Generating Production Rules From Decision Trees

J. R. Quinlan*
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
545 Technology Square
Cambridge MA 02139 USA

Abstract

Many inductive knowledge acquisition algorithms generate classifiers in the form of decision trees. This paper describes a technique for transforming such trees to small sets of production rules, a common formalism for expressing knowledge in expert systems. The method makes use of the training set of cases from which the decision tree was generated, first to generalize and assess the reliability of individual rules extracted from the tree, and subsequently to refine the collection of rules as a whole. The final set of production rules is usually both simpler than the decision tree from which it was obtained, and more accurate when classifying unseen cases. Transformation to production rules also provides a way of combining different decision trees for the same classification domain.

Introduction

A *decision tree* is a simple recursive structure for expressing a sequential classification process in which a *case,* described by a set of *attributes,* is assigned to one of a disjoint set of *classes.* Each leaf of the tree denotes a class. An interior node denotes a *test* on one or more of the attributes with a subsidiary decision tree for each possible outcome of the test. To classify a case we start at the root of the tree. If this is a leaf, the case is assigned to the nominated class; if it is a test, the outcome for this case is determined and the process continued with the subsidiary tree appropriate to that outcome.

Figure 1 shows a non-trivial decision tree for one aspect of the diagnosis of thyroid disease (Quinlan, Compton, Horn and Lazarus, 1986). To simplify printing, the tree has been turned on its side. Leaves are shown in bold font, and the possible outcomes at an interior node are represented by logical expressions with equal indentation. The interpretation of the attributes and decision classes is not important here, but notice that the root of this tree is a test on attribute T3. When classifying a case, we will be directed to the subtree starting with *FTI* < 49.5 or that headed *FTI* < 172.5 depending on whether the value of *TZ* is less than, or greater than or equal to, 1.15.

Research that commenced in the late 1950s with Hunt's *Concept Learning System* (Hunt, Marin and Stone, 1966) has led to several reliable methods for developing decision trees from *training sets* of cases with known classes. Modern systems of this type, such as those described in (Breiman, Friedman, Olshen and Stone, 1984; Kononenko, Bratko and Roskar, 1984; Quinlan, 1986) can deal effectively with large training sets affected by noise and incompleteness, and can classify new cases even when the outcome of crucial tests is unknown.

The starting point for this paper is a decision tree developed by some means from a training set of cases. We examine methods for re-expressing the decision tree as a succinct collection of *production rules* of the form

If left-hand side then class (certainty factor)

There are three reasons for such a transformation. First, production rules are a widely-used and well-understood vehicle for representing knowledge in expert systems (Winston, 1984). Secondly, a decision tree such as that in Figure 1 can be difficult for a human expert to understand and modify, whereas the extreme modularity of production rules makes them relatively transparent. Finally, and most importantly, this transformation can improve classification performance by eliminating tests in the decision tree attributable to peculiarities of the training set, and by making it possible to combine different decision trees for the same task.

The transformation takes place in two stages addressed in the following sections. Individual rules are first developed from the decision tree, and the collection of rules so derived is then processed as an entity to yield the final ruleset.

Extracting Individual Rules

Recall that classifying a case using a decision tree is effected by following a path through the tree to one of the leaves. This path from the root of the tree to a leaf establishes *conditions,* in terms of specified outcomes for the tests along the path, that must be

* Permanent address: School of Computing Sciences, New South Wales Institute of Technology, Sydney 2007, Australia.

** Much of the research described here was carried out at the Artificial Intelligence Laboratory, Massachusetts Institute of Technology, whose artificial intelligence research is supported in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-85-K-0124. This research has also received support from the Australian Research Grants Scheme and the Westinghouse Electric Corporation.

```
T3 < 1.15:
|   FTI < 49.5: negative
|   FTI ≥ 49.5:
|   |   TT4 < 52:
|   |   |   TSH < 4.1:
|   |   |   |   age ≥ 55: negative
|   |   |   |   age < 55:
|   |   |   |   |   sex = M: negative
|   |   |   |   |   sex = F: sick
|   |   |   TSH ≥ 4.1:
|   |   |   |   age ≥ 74: sick
|   |   |   |   age < 74:
|   |   |   |   |   on thyroxine: sick
|   |   |   |   |   not on thyroxine: negative
|   |   TT4 ≥ 52:
|   |   |   FTI ≥ 157: negative
|   |   |   FTI < 157:
|   |   |   |   FTI < 61.5: negative
|   |   |   |   FTI ≥ 61.5:
|   |   |   |   |   referral source ∈ {SVHC,SVI,SVHD}: sick
|   |   |   |   |   other referral source:
|   |   |   |   |   |   age ≥ 71.5: sick
|   |   |   |   |   |   age < 71.5:
|   |   |   |   |   |   |   illness noted: sick
|   |   |   |   |   |   |   no illness noted:
|   |   |   |   |   |   |   |   TSH < 8.85: negative
|   |   |   |   |   |   |   |   TSH ≥ 8.85: sick
T3 ≥ 1.15:
|   FTI < 172.5:
|   |   TT4 < 56.5:
|   |   |   TSH ≥ 5.75: negative *
|   |   |   TSH < 5.75:
|   |   |   |   referral source = SVI: sick
|   |   |   |   other referral source: negative
|   |   TT4 ≥ 56.5:
|   |   |   FTI < 163.5: negative
|   |   |   FTI ≥ 163.5:
|   |   |   |   FTI ≥ 164.5: negative
|   |   |   |   FTI < 164.5:
|   |   |   |   |   on thyroxine: negative
|   |   |   |   |   not on thyroxine:
|   |   |   |   |   |   suspected hyperthyroid: negative
|   |   |   |   |   |   not suspected hyperthyroid: sick
|   FTI ≥ 172.5:
|   |   T3 ≥ 2.75: negative
|   |   T3 < 2.75:
|   |   |   on thyroxine: negative
|   |   |   not on thyroxine:
|   |   |   |   TT4 ≥ 195: negative
|   |   |   |   TT4 < 195:
|   |   |   |   |   TSH < 0.31: sick
|   |   |   |   |   TSH ≥ 0.31: negative
```

Figure .1: Sample Decision Tree

satisfied by any case classified by that leaf. For example, any case that is classified as *negative* by the asterisked leaf near the middle of Figure 1 must satisfy all the conditions

$$T3 \geq 1.15$$
$$FTI < 172.5$$
$$TT4 < 56.5$$
$$TSH \geq 5.75$$

Every leaf of a decision tree thus corresponds to a primitive production rule of the form

$$\text{if } X_1 \wedge X_2 \wedge \ldots \wedge X_n \text{ then class c}$$

where the X_i'S are conditions and c is the class of the leaf.

At this point we make use of the training set *T* of cases from which the decision tree was generated in order to improve this prototype rule. Let X_i be one of these conditions and let $S \subset T$ be the set of cases that satisfy all the other conditions in the left-hand side of our rule. With respect only to 5, the relevance of X_i to determining whether a case belongs to class *c* (given that the other conditions are satisfied) can be summarized by the 2 x 2 contingency table

	class c	not class c
satisfies X_i	sc	$s\bar{c}$
does not satisfy X_i	$\bar{s}c$	$\bar{s}\bar{c}$

where se is the number of these cases that satisfy X_t and belong to class e, se is the number that satisfy X_i but belong to some class other than c, and so on.

Note that sc-f *sc* is the number of cases in the training set *T* that satisfy the entire left-hand side of the rule and that *ac* of them belong to the class nominated by the rule. These two numbers provide a means of estimating the accuracy or *certainty factor* of the rule. The obvious choice of setting

$$C(sc, s\bar{c}) = \frac{sc}{sc + s\bar{c}}$$

can be rather optimistic, especially when the numbers are small. Since for any reasonable rule *sc* will be larger than *ac,* the use of Yates' correction for continuity (Snedecor and Cochran, 1980, p118) gives a more reasonable estimate as

$$C(sc, s\bar{c}) = \frac{sc - \frac{1}{2}}{sc + s\bar{c}}$$

There are at least two sets of circumstances under which this condition X_i should be deleted from the left-hand side of the rule. The first typically arises with *disjunctive concepts* (Bundy, Silver and Plummer, 1985) in which a case belongs to a particular class whenever a disjunctive logical expression of the form $Y \vee Z$ is satisfied. A decision tree for such a classification task might commence with a test that is relevant to *Y* but not to Z, so the leaves associated with the disjunct *Z* will generate prototype rules that contain irrelevant conditions. If X_i is such a condition, eliminating it will produce a more general rule without any decrease in accuracy, i.e. $C(sc + \bar{s}c, s\bar{c} + \bar{s}\bar{c}) \geq C(sc, s\bar{c})$.

Secondly, the presence of X_i in the left-hand side of the rule may give greater apparent accuracy, but this accuracy may derive from chance characteristics of the training set that cannot be expected to hold for unseen cases. The algorithm used to construct the decision tree from the training set *T* has probably attempted to 'fit' the data, even when it is noisy or inconclusive. Under these circumstances, retaining X_i can be dangerous because the seeming reliability of the rule can lend false confidence to a classification. There are several statistical tests that can be used to signal this state of affairs. Following a suggestion of Donald Michie, I use Fisher's exact test (Finney, Latscha, Bennett and Hsu, 1963) to determine the significance level at which we can reject the hypothesis that X_i is irrelevant to whether a case satisfying all the other conditions belongs to class *c.* If this level is not very small, the condition X_i is deleted.

The algorithm for dropping conditions from the left-hand tide of a rule can now be stated succinctly. Condition X_i is a candidate for elimination either if its removal will not decrease the certainty factor of the rule, or if the hypothesis that X_i is irrelevant cannot be rejected at the 1% level or better. So long as there are candidates for elimination, we discard the one whose removal has the least detrimental effect on the accuracy of the rule, and continue. Of course, after any X_i has been removed, the contingency tables for the remaining conditions must be recalculated.

As an illustration of the process, consider the rule above, extracted from Figure 1 which was in turn generated from a training set of 2800 cases. We focus first on the condition $TT4 < 56.5$. The contingency table over all cases satisfying the remaining conditions is

	class *negative*	not class *negative*
$TSH < 56.5$	12	0
$TSH \geq 56.5$	141	0

so that removing this condition will increase the value of the certainty factor. The same holds for the condition $FTI < 172.5$. In the reduced rule the contingency table for the condition $TSH > 5.75$ is

	class *negative*	not class *negative*
$TSH < 5.75$	168	0
$TSH \geq 5.75$	1809	20

Even though the rule without this condition is apparently less accurate, the condition is removed because the hypothesis that it is irrelevant to whether a case in 5 is class *negative* can only be rejected at the 17% level. The remaining condition is significant at better than the 0.1% level, so the final rule from this path becomes

if T3 > 1.15 then class negative (99.0%)

The number of rules generated in this way is almost always smaller than the number of leaves in the decision tree. Some paths generate no rules, either because all conditions are eliminated or because the rule replicates another from a different path. In this example, although the decision tree of Figure 1 has 27 leaves, the process above produces just 13 rules.

[Aside: The reader may wonder why we use the decision tree at all, instead of developing rules directly from the training set of cases. Working from the tree has two major advantages. Most interesting classification tasks involve attributes with continuous values which must be formed into tests by the development of appropriate threshholds (e.g. $TZ < 1.15$ from before). The divide-and-conquer approach commonly employed by algorithms for constructing decision trees provides a powerful and context-sensitive means of coping with this otherwise complex problem. Secondly, even a long path in a decision tree typically involves only a small proportion of the possible attributes. The training set of Figure 1 uses 23 attributes to describe each case, but no path in the decision tree uses more than nine tests; the space of potential rules is thus shrunk from $0(2^{23})$ to $0(2^9)$ with a corresponding reduction in computational load.]

Processing Collections of Rules

Having reduced the given decision tree to a set of plausible rules, we might judge the transformation task to have been accomplished. It seems relevant to wonder, though, how well the rules classify unseen cases, and whether some subset of the rules might be as useful as the whole set. These questions presume that there is some target production rule interpreter in the wings. The following uses an extremely simple interpreter:

> To classify a case, find a rule that applies to it. If there is more than one, choose the rule with the higher certainty factor. If no rule applies, take the class by default to be the most frequent class in the training set.

Alternative and equally sensible interpreters (e.g. those that choose the most specialized applicable rule) should produce similar results.

Let R be the set of production rules and T the training set of cases from which the decision tree was generated. We would like to find that subset of R which misclassified the fewest cases in T but, by analogy with the set-covering task, this is an NP problem. Instead, a heuristic algorithm is used to find a "good" subset by successively discarding single rules.

For any case in T and any single rule r, we look at the class to which this case would be assigned by the entire set R and the reduced set R - {r}. The *advantage* of r is the number of cases in T for which the correct class is given by R but not by R - {r}> less the number of cases vice versa. If the advantage of r is negative or zero, removing r from the set of production rules will not increase the number of cases in T that are misclassified. This suggests a straightforward procedure: at each step, delete from R the rule with least advantage, so long as this advantage is less than or equal to zero. The set of rules remaining at the end of this process is locally optimal to the extent that deleting any further rule will increase the number of misclassifications over T. (This may overlook, however, situations in which deleting a subset of the rules would improve performance.) The procedure usually finds a good subset of R, but is weak when the initial set of rules contains many pairs of very similar rules: in this situation, most rules will have advantage 0 and so advantage is a poor basis on which to choose the rule to delete.

We saw previously that the decision tree of Figure 1 with 27 leaves gave rise to 13 production rules. The winnowing process described above reduces this set to just four rules with an average of 3.75 conditions per rule.

Accuracy

We now turn to the classification accuracy of the reduced set of rules. Since each rule was formed by eliminating conditions from a path in the tree, it tends to be over-generalized with respect to the training set. However, the relevant test of any classification mechanism is its performance on unseen cases.

Table 1: Average Results In Six Domains

Domain	Decision Trees		Production Rules	
	Average Size (Nodes)	Average Error Rate	Average Size (Rules)	Average Error Rate
Endocrinology:				
hypothyroidism	23.6	0.6%	3.0	0.7%
discordant assay	52.4	1.9%	1.8	1.3%
Faulty LED digits	92.2	29.0%	15.8	29.8%
Credit assessment	248.0	20.6%	7.8	16.5%
Chess endgame	88.0	11.2%	11.6	9.2%
Disjunctive concept	190.0	17.7%	4.2	10.0%

When presented with 972 unseen cases, the decision tree of Figure 1 misclassifies 14 of them as compared to 13 errors from the final set of four production rules above. Results from other experiments reported in detail in (Quinlan, 1987) are summarized in Table 1. In each of six domains, ten decision trees were generated from a training set and their performance measured on unseen cases. Each decision tree was then transformed to a set of production rules whose accuracy was assessed on the same unseen cases. The average sizes and error rates for each domain shown in Table 1 bring out the point that the production rules are generally much simpler and sometimes more accurate than the decision tree from which they were generated.

Another advantage of transforming decision trees to production rules is their resulting modularity. There is no obvious way to combine two decision trees for the same classification task so as to generate a super-tree that is more accurate than either of its parents. If each decision tree is converted to a set of rules, though, a composite reduced set can be produced simply by merging rules from all trees before applying the final winnowing process outlined above. This approach has been found to give encouraging results. For example, ten decision trees derived from the same training set as the tree of Figure 1, when combined in this way, yield a set of five production rules that correctly classify all but 8 of the 972 unseen cases, even though the best of the trees gives 11 errors on these same cases.

Conclusion

The conclusion of this work is that it is possible to re-express complex decision trees as small sets of production rules that outperform the original trees when asked to classify unseen cases. The methods outlined here also provide a way to merge different decision trees for the same task, thereby obtaining another increase in accuracy.

This method for reducing the number of rules can be contrasted with the TRUNC algorithm employed in AQ15 (Michalski, Mozetic Hong and Lavrac, 1986). The analog of a rule in that system is a *complex* or conjunction of conditions associated with a class. Unlike rules, complexes are exact in the sense that any case in the training set satisfying all the conditions is guaranteed to belong to the designated class. At each iteration, TRUNC discards the complex satisfied by the fewest cases in the training set until some stopping criterion is met. AQ15 uses a powerful form of partial or *analogical* matching to allow a case which satisfies no complex to be deemed to match the most similar complex. As a result, even though deleting a complex cannot decrease the number of misclassified training cases, it may not necessarily cause an increase in this number. Interestingly, Michalski e*t al* also report that removal of little-used complexes has been found to lead to improved classification performance on unseen cases.

Although the algorithms presented here work well, they should be capable of further improvement. Both the condition-dropping and rule-dropping processes use a hill-climbing approach which can often get stuck on a local optimum. More sophisticated search strategies should generate better individual rules and better rule sets at the cost of some increase in computation.

Acknowledgements

I thank many colleagues for comments and suggestions, particularly Wray Buntine, Jason Catlett, Bill Leech, John McDermott, Donald Michie and Ron Rivest. I am grateful to the Garvan Institute of Medical Research, Sydney, for providing access to the thyroid data.

References

Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (1984), *Classification and Regression Trees,* Belmont: Wadsworth.

Bundy, A., Silver, B. and Plummer, D. (1985), An analytical comparison of some rule-learning programs, *Artificial Intelligence 27,* pp 137-181.

Finney, D.J., Latscha, R., Bennett, B.M. and Hsu, P. (1963), *Tables for Testing Significance in a* 2 x 2 *Contingency Table,* Cambridge University Press.

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

Kononenko, I., Bratko, I. and Roskar, E. (1984), Experiments in automatic learning of medical diagnostic rules, Technical Report, Jozef Stefan Institute, Ljubljana, Yugoslavia.

Michalski, R.S., Mozetic, I., Hong, J. and Lavrac, N. (1986), The multi-purpose incremental learning system AQ15 and its testing application to three medical domains, Proc. Fifth National Conference on Artificial Intelligence, Philadelphia.

Quinlan, J.R. (1986), Induction of decision trees, *Machine Learning 1,* 1.

Quinlan, J.R., Compton, P.J., Horn, K.A. and Lazarus, L. (1986), Inductive knowledge acquisition: a case study, *Proc. Second Australian Conference on Applications of Expert Systems,* Sydney.

Quinlan, J.R. (1987), Simplifying decision trees, *Int. Journal Man-Machine Studies,* to appear.

Snedecor, G.W. and Cochran, W.G. (1980), *Statistical Methods* (7th edition), Iowa State University Press.

Winston, P.H. (1984), *Artificial Intelligence* (2nd edition), Addison-Wesley.