Decision Trees and Decisionmaking

J. R. QUINLAN

***Abstract*—Various practical systems capable of extracting descriptive decisionmaking knowledge from data have been developed and evaluated. Techniques that represent knowledge about classification tasks in the form of decision trees are focused on. A sample of techniques is sketched, ranging from basic methods of constructing decision trees to ways of using them noncategorically. Some characteristics that suggest whether a particular classification task is likely to be amenable or otherwise to tree-based methods are discussed.**

I. Introduction

DECISIONMAKERS need to make predictions—whether the U.S. dollar will rise in the short term, whether a patient will benefit from a surgical procedure, whether it will rain tomorrow. One sound basis for such predictions is an extrapolation of past, known cases. The science of statistics provides a range of tools for this purpose, usually based on the idea of fitting a particular class of models to the data and then hypothesising that future events will conform to the fitted model. Researchers in artificial intelligence (AI) have long been interested in the same task, usually from a less model-driven standpoint. The basic scenario for this branch of learning is one in which an intelligent agent, shown a collection of case studies of some activity, employs inductive inference to derive useful information about that activity. As with their statistical counterparts, many early learning programs were concerned with finding appropriate values for numeric parameters, a line of research well summarized in [23]. Samuel's work with checkers used a mixture of symbolic and numerical learning that allowed his program to achieve expert status [41]. EPAM [9] and CLS [12], [13] were pioneering systems for acquiring purely symbolic classificatory knowledge from a study of training objects of known class.

In their 1983 article, Carbonell, Michalski, and Mitchell suggest *underlying learning strategy* as a fruitful dimension along which learning methods can be ranked, with the metric being the amount of inference that is carried out by the learner. Rather than forming a continuum, though, research seems to have polarized into two clumps on this scale. At one end the objects given to the learner are complex and support deep inference, so the learning programs tend to extract a great deal of knowledge from only a few of them. This approach is typified by Winston's program, which learns the concept of an arch from analyzing structured descriptions of scenes [48] and by Marvin's ability to induce Prolog programs from ground instances and dialog with a tutor [40]. On the other hand, if the objects are simple and unstructured, it is possible to examine large numbers of them in a search for knowledge. In fact, if the training objects contain errors and omissions so that no theory can correctly account for all of them, a search for patterns across many objects may be the only feasible way to proceed.

This paper is concerned with a subclass of programs in the latter clump. Learning takes place over fixed-format descriptions of objects, each of which belongs to one class. Programs attempt to construct some classification procedure that accurately models the classes of the known objects so that, when an object of unknown class is encountered, a plausible class label can be affixed to it. This classification know-how can be represented in many ways. AQ11 [18] and the more recent AQ15 [19] express it as *complexes* or conjunctions of conditions, a class being described by one or more such complexes. PLS1 [38] uses a similar representation in which each class is denoted by one or more rectangles in the description space of the objects. CN2 [8] employs the familiar *production rule* formalism [49] to represent classification knowledge. STAGGER [42] is an incremental learning system that uses *weighted characterizations*, logical expressions with likelihood ratios that capture the sufficiency and necessity of the expression for predicting that an object belongs to a designated class.

We focus here on systems that discover classification rules in the form of *decision trees*. Systems of this type employ a top-down, divide-and-conquer strategy that partitions the given set of objects into smaller and smaller subsets in step with the growth of the tree. After sketching Hunt's basic technique, the paper discusses some of the extensions and amplifications that are necessary if real-world data is to be employed. In Section V we show why some tasks are ill-suited to this approach and, conversely, that it is especially effective for others. The paper concludes with some comments on current research initiatives.

Manuscript received January 16, 1989; revised September 1, 1989. This work was supported in part by a grant from the Australian Research Council. This paper is a revised and expanded version of one presented in June 1988 at the Illinois Interdisciplinary Workshop on Decision Making, Ubana-Champaign, IL.

The author is with the Basser Department of Computer Science, University of Sydney, Sydney, NSW2006 Australia.

IEEE Log Number 8933031.

0018-9472/90/0300-0339$01.00 ©1990 IEEE

II. Terminology

We start with a universe of *objects*, each belonging to one of a disjoint set of *classes*. The properties of objects are known only through their values of a collection of *attributes*, so that two objects can be distinguished only if they have different values for some attribute. Attributes may be either *discrete*, having values drawn from a known set of possible values, or they may be *continuous* with values that are real numbers. For example, the weather on a particular day could be (loosely) described by the attributes:

- the *outlook*, whose possible values are *rain*, *overcast* or *sunny*
- the *temperature*, with continuous values
- the *humidity*, again with continuous values
- whether it is *windy*, with values *true* or *false*.

A particular object might be described as

$$\text{outlook} = \text{overcast, temperature} = 78°, \\ \text{humidity} = 80\%, \text{windy} = \text{false}$$

A *training set* is a collection of objects whose classes are known. The assumption underlying all inductive learning is that, if patterns can be found that account for the class membership of the known objects, these same patterns will predict the class of a new, unseen object. A *classification task* is the problem of finding a general classification rule that works well on the objects in a given training set. By the above assumption, this rule codifies regularities in the training set that will be useful when trying to classify an unseen object.

A *decision tree* is a recursive structure for expressing classification rules. Such a tree may be a *leaf* associated with one class. Alternatively, the tree may consist of a *test* that has a set of mutually exclusive possible outcomes together with a subsidiary decision tree for each such outcome. To classify an object, we start with the root of the tree. If this is a leaf, the object is assigned the class associated with that leaf. Alternatively, if the root of the decision tree is a test, the outcome of this test for the given object is determined and the process continues using the subsidiary decision tree associated with that outcome. An object is thus classified by tracing out a path from the root of the decision tree to one of its leaves.

To continue the example, a decision tree that classifies days as either *Play* or *Don't Play* might be represented as shown in Fig. 1, and such a decision tree would assign the day described above to class *Play*.

As this tree illustrates, each test is commonly restricted to being a function of a single attribute. While any function of the attributes could feasibly be used for a test, this would tend to have two detrimental outcomes. As will be seen below, decision trees are constructed by examining all possible tests at each stage, so increasing the forms to be explored would have a computational cost. More importantly, though, a decision tree in which each test was a complex function might be considerably more inscrutable. Tree-based methods are thus particularly sensitive to the way attributes are formulated. The implicit assumption here is that the person who defined the attributes has provided us with the relevant ammunition for constructing decision trees. If some more complex test were appropriate, such as testing whether $A_i > A_j$ for distinct continuous attributes A_i and A_j (a function of two attributes), there is nothing to prevent the attribute definer from including a redundant attribute designed to flag this condition.[1]

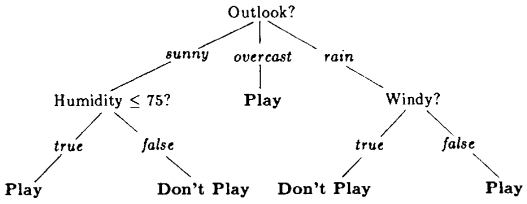


Fig. 1. Decision tree.

No matter what form a test takes, the fact that it has a discrete set of outcomes allows the definition of a simple algorithm for discovering decision trees from training sets.

III. Algorithm for Constructing Decision Trees

Hunt's procedure for generating a decision tree from a set S of objects, each belonging to one of the classes $C_1, C_2, \cdots, C_k$, is as follows.

1) If all the objects in S belong to the same class, C_i say, the decision tree for S consists of a leaf labelled with this class.
2) Otherwise, let T be some test with possible outcomes $O_1, O_2, \cdots, O_n$. Each object in S has one outcome for T so the test partitions S into subsets $S_1, S_2, \cdots, S_n$ where each object in S_i has outcome O_i for T. T becomes the root of the decision tree and for each outcome O_i we build a subsidiary decision tree by invoking the same procedure recursively on the set S_i.

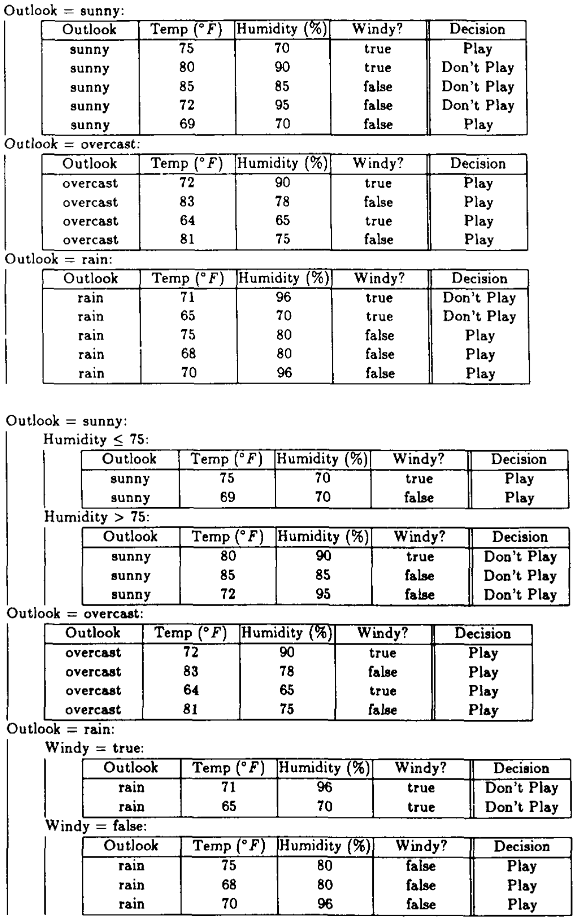
The attributes are *adequate* for the classification task unless there are two objects in the training set that have the same value for every attribute but belong to different classes. If the attributes are adequate it is always possible to construct a test at step 2 that results in a non-trivial partition of S, thereby assuring that the algorithm terminates.

The process is illustrated with the small training set of objects in Table I. Each object is described in terms of the attributes given earlier and belongs to one of the decision classes *Play*, *Don't Play*. When this set is partitioned by testing on *outlook*, the resulting three subsets are shown at the top of Fig. 2. Further division of the first and third subsets leads to the situation at the bottom of Fig. 2, which is equivalent to the decision tree of Fig. 1.

[1] We will return to the issue of more complex tests in Section 6.

TABLE I
SMALL TRAINING SET

Outlook	Temp (°F)	Humidity (%)	Windy?	Decision
rain	71	96	true	Don't Play
rain	65	70	true	Don't Play
overcast	72	90	true	Play
overcast	83	78	false	Play
rain	75	80	false	Play
overcast	64	65	true	Play
sunny	75	70	true	Play
sunny	80	90	true	Don't Play
sunny	85	85	false	Don't Play
overcast	81	75	false	Play
rain	68	80	false	Play
rain	70	96	false	Play
sunny	72	95	false	Don't Play
sunny	69	70	false	Play



Outlook = sunny:

Outlook	Temp (°F)	Humidity (%)	Windy?	Decision
sunny	75	70	true	Play
sunny	80	90	true	Don't Play
sunny	85	85	false	Don't Play
sunny	72	95	false	Don't Play
sunny	69	70	false	Play

Outlook = overcast:

Outlook	Temp (°F)	Humidity (%)	Windy?	Decision
overcast	72	90	true	Play
overcast	83	78	false	Play
overcast	64	65	true	Play
overcast	81	75	false	Play

Outlook = rain:

Outlook	Temp (°F)	Humidity (%)	Windy?	Decision
rain	71	96	true	Don't Play
rain	65	70	true	Don't Play
rain	75	80	false	Play
rain	68	80	false	Play
rain	70	96	false	Play

Outlook = sunny:

Humidity ≤ 75:

Outlook	Temp (°F)	Humidity (%)	Windy?	Decision
sunny	75	70	true	Play
sunny	69	70	false	Play

Humidity > 75:

Outlook	Temp (°F)	Humidity (%)	Windy?	Decision
sunny	80	90	true	Don't Play
sunny	85	85	false	Don't Play
sunny	72	95	false	Don't Play

Outlook = overcast:

Outlook	Temp (°F)	Humidity (%)	Windy?	Decision
overcast	72	90	true	Play
overcast	83	78	false	Play
overcast	64	65	true	Play
overcast	81	75	false	Play

Outlook = rain:

Windy = true:

Outlook	Temp (°F)	Humidity (%)	Windy?	Decision
rain	71	96	true	Don't Play
rain	65	70	true	Don't Play

Windy = false:

Outlook	Temp (°F)	Humidity (%)	Windy?	Decision
rain	75	80	false	Play
rain	68	80	false	Play
rain	70	96	false	Play

Fig. 2. Successive subdivisions of training set.

This top-down divide-and-conquer approach can be used to advantage even where the final product is not a decision tree. PLS1 [38] considers training objects to be enclosed in a rectangle in the description space. A concept is refined by splitting one of the rectangles using a hyperplane parallel to one of the attribute axes, a process equivalent to subdividing the training set by some test on a single attribute. These methods can all be described under the general banner *induction by specialization* [39].

The algorithm for constructing a decision tree from a training set is nondeterministic since any test that gives a non-trivial partition of S may be chosen in step 2. The decision tree that it generates will correctly classify all objects in the training set (providing that the attributes are adequate) but this does not mean that all possible trees are of equal worth. The reason for generating a decision tree in the first place is to generalize over the objects in the training set and so provide a handle on classifying unseen objects. If two trees D_1 and D_2 both classify correctly all objects in the training set, but D_1 is considerably simpler than D_2, it seems plausible that D_1 represents a greater level of generalization and so better captures some underlying regularity in the domain. This can be stated as the *first strategic heuristic* for constructing decision trees: simpler trees are preferable.

This heuristic can be supported by more theoretical arguments. Pearl [26] shows that the credibility of a theory expressed in any of three formal languages is adversely affected by its complexity. Under certain assumptions, the expected error rate when a decision tree derived from a given training set is used to classify unseen objects increases with complexity as measured by the number of leaves on the tree [28].

Complexity might be defined here in various ways. Common measures include the above number of leaves (or nodes) in the decision tree, or the average number of tests required to clasify objects in the training set. Other measures are related to the structure of the decision tree. Arbab and Michie [1], for example, equate simplicity with the *linearity* of a decision tree, defined as the proportion of tests for which at most one outcome is associated with a non-leaf.

One possible approach to constructing a decision tree would be to generate all trees that correctly classify the training set and then pick the simplest under whatever criterion is being employed. Unfortunately, Hyafil and Rivest [14] have shown that the problem of finding a decision tree with the minimum expected number of tests is NP-complete, and it seem likely that any optimization process over decision trees is also computationally infeasible. A simple example should illustrate the difficulty. Consider the case in which there are two classes and only four discrete attributes, two with two possible values and two with three. Any classification task using this framework would rightly be regarded as tiny, but the number of sensible decision trees that it supports exceeds 2.2×10^{14}. The size of the search space for any realistic classification task would appear to preclude any form of exhaustive optimization.

As mentioned above, the structure of the decision tree generated by the algorithm is determined by the choice of test in step 2. Clearly, we wish to select a test that results in a "good" partition of S, one which allows the subsidiary decision trees to be as simple as possible. The *second strategic heuristic* governing the learning of decision trees is that the complexity of a tree for a set S of objects can be predicted by looking at the way the objects in S are distributed among the various classes. For instance, a set consisting largely of objects from a single class seems to be pretty close to a leaf, while a set which contains equal numbers of objects from all classes is likely to require a complex subtree to sort out.

The numerous heuristic criteria that have been proposed for making this choice can be grouped into three major streams:

- Heuristics based on *information* or *entropy* [2], [6], [27], [34], [36]. These heuristics rank prospective tests on the apparent information that they provide regarding class membership of objects in the training set. Some heuristics attempt to take account of varying numbers of outcomes for the tests being compared.
- Heuristics based on *error* [2], [13]. Suppose that, after making a test, we had to "freeze" the tree in some form. This group of heuristics assesses the worth of the test by the reduction in errors that we would make classifying the training objects. The Gini Diversity Index [2] belongs in this category since it is equivalent to measuring the expected error when the subsets arising from the test are replaced by probabilistic classifiers.
- Heuristics based on *statistical significance* [11], [20], [21]. Potential tests are ranked by the confidence with which we can reject the hypothesis that they are irrelevant to the class of objects in the training set.

Although they all seem reasonable, the use of different heuristic criteria can subtly but significantly affect the trees produced, especially when the training set is small and/or possible tests have very different numbers of outcomes.

IV. Coping With Mother Nature

The basic algorithm for constructing decision trees ignores complexities that arise in real-world classification tasks. The attributes may *not* be adequate; it may be necessary to assign a leaf to a set S containing objects from more than one class, either by deciding not to partition such a set or by subsequently replacing the generated decision tree with a leaf. The divide-and-conquer approach can produce empty subsets when there are three or more outcomes of a test and, while the choice of which class to associate with such a subset does not matter as far as the training set is concerned, it may be important when the decision tree is used to classify unseen objects. Worse still, we may have to deal with objects whose values for some of the attributes are unknown, leading to problems when building the tree (to which S_i does the object belong?) or when classifying an object (which outcome should be followed?). A full-blown algorithm for constructing decision trees will incorporate methods for dealing with the following problems, among others.

A. Noisy Data

In real-world classification tasks it is frequently the case that the class of objects in the training set cannot be expressed as a function of the attribute values. This can arise either because the attribute values contain errors, or because the attributes collectively provide insufficient information to classify an object. For example, whether it will rain this afternoon is (presumably) a deterministic function of the physical environment of the region, but an attempt to predict whether it will rain on the basis of approximate descriptions of this environment will not be uniformly successful; days with apparently identical descriptions will have different outcomes.

In these circumstances, continuing to divide the training set until all subsets contain members of a single class may be impossible. Even when such perfect division is possible it may be inadvisable, as the lower levels of the tree may model idiosyncrasies of the training set rather than structure that is useful for classifying unseen objects; we may be attempting to fit the noise in the training data.

There are two approaches to dealing with this problem. In one of them a heuristic, sometimes known as a *stopping criterion* [2], determines whether or not a multiclass set of training objects should be divided further, using either features such as its size or statistical significance tests. An example of the former might be, "Do not split any set containing fewer than eight objects," and of the latter, "Do not permit a test on an attribute unless the hypothesis that the attribute is independent of the objects' class can be rejected with high confidence." Further discussion may be found in [6], [11], [20], [30].

The other approach is to allow the tree to grow without constraint, then to remove unimportant or unsubstantiated portions by *pruning* it. Pruning involves replacing a subtree by one of its branches or, more commonly, by a leaf. While this involves more computation than the first approach, it allows the effect of subsequent tests to be evaluated before deciding whether a particular test is significant. Several different methods have been explored, examples being

- Cost Complexity Pruning [2]. The desirability of a tree is measured as a function of its accuracy on the training set and its size. A sequence of pruned variants of the original tree is formed and one of them selected on the basis of performance on a test set of unseen objects that were not used to construct the tree.
- Minimum Description Length Pruning [36]. Working from Rissanen's MDL Principle, that variant of the tree is found which minimizes the total information required to specify the class of all training objects via a general rule (the pruned tree) and exceptions to the rule.
- Pessimistic Pruning [32]. This method increases the estimated error rates of subtrees to reflect the size

and composition of the training subsets, then replaces every subtree whose predicted error rate is not significantly lower than that of a leaf.

Most pruning methods usually result in a smaller tree that has the additional advantage of increased accuracy when classifying unseen objects. A comparison of some techniques appears in [32].

A closely-related problem to that of noise occurs when the attributes are not sufficiently informative to allow any classifier to perform without error on the training set. Spangler, Fayyad and Uthurusamy [45], who refer to such data as *inconclusive*, have designed a specialized stopping criterion and heuristic for test selection under the assumption that the final classifier will be imperfect.

B. Unknown Attribute Values

Another bugbear of data in the wild is its incompleteness; it is frequently the case that some objects in the training set have unknown values for one or more attributes. This can come about either because the information is unavailable or missing, or because that value is irrelevant in the light of values for the other attributes. For example, a patient's temperature might not be recorded, either because it was inadvertently not taken, or because temperature had no bearing on the patient's condition.

This problem affects both the discovery of a decision tree and its subsequent use—the objects in the training set may have missing attribute values and, when an object is classified, it may be impossible to determine the outcome of a test. Once again there are two rather different ways around the problem.

In the tree construction phase, Breiman *et al.* [2] rank possible tests by ignoring training objects that have unknown values of the attribute in question. Having decided on a test T, they introduce the notion of a *surrogate split*, a test on a different attribute which produces a division of the current training objects that most resembles the division produced by the test T. When dividing these objects, the outcome for objects whose outcome for T is unknown is taken as the corresponding outcome of the surrogate test. Again, when an unseen object is being classified and the outcome of a test is unknown, the outcome of the surrogate test is used instead. In fact, there is a ranked list of surrogate splits for each test so that, if the first surrogate employs an attribute whose value is also unknown, the second surrogate is used, and so on.

This approach is based on the idea of using other information (a test on the surrogate attribute) to determine an outcome for the test on an unknown value, and to treat this value thereafter as known. An alternative approach is to treat the outcome as "fuzzy" or probabilistic. When C4[2] builds a tree, it assesses the merits of each possible test by notionally distributing objects with unknown attribute values among the outcomes [29]. Once a test is chosen, objects with unknown values of the tested attribute are discarded. When classifying an unseen case that has an unknown value of the tested attribute, all outcomes of the test are explored and the results combined probabilistically, the weights associated with the various outcomes being taken as their relative frequencies in the training set at that point [31].[3]

Other methods of determining unknown values have been tried. CN2 [8], simply replaces an unknown attribute value with the most commonly-occurring value of that attribute. ASSISTANT [6] uses a Bayesian formalism to compute the most likely value given the values of other attributes. Shapiro (private communication) suggests building a decision tree for each attribute to predict that attribute's value from the values of the other attributes. Experiments with these approaches are reported in [29].

C. Less-Than-Certain Classification

Whether from application of a stopping criterion or from subsequent pruning, it may now happen that the training objects associated with a leaf are no longer members of a single class. Consider two leaves, one containing 100 objects, all from class C, and another containing three objects, two of which belong to class C. In either case C is the best choice of class for unseen objects that find their way to these leaves, but there the similarity ends. In the first case statistical evidence allows us to be reasonably confident that the unseen object really does belong to class C, while in the second we ought to be much less confident.

Rather than give a bald classification, it is a simple step to provide additional information on its reliability. PLS1 can provide probabilistic membership for objects falling into a rectangle in the description space [38]. Carter and Catlett [5] introduce the idea of *class probability trees*. In the same vein, C4 provides a menu of possible classes for an object based on the class membership of a leaf and uncertainties in determining the outcome of tests on the path to that leaf [31].

V. When are Tree-Based Methods Appropriate?

In a decisionmaking context, the principal advantages of tree-based methods are:

- *Clarity and Conciseness*—The classification knowledge is presented in a form that human decisionmakers can understand and scrutinize.
- *Context Sensitivity*—Attributes may not be uniformly helpful in different decisionmaking contexts. Decision trees, and in fact all products of divide-and-

[2] Over the years many modifications and refinements have been made to ID3 in areas such as the criterion for selecting tests, pruning, probabilistic classification and generation of rules from decision trees. As a means of differentiating the current systems from its simpler ancestor, it has been renamed C4.

[3] Since the conjunction of conditions associated with different leaves are necessarily mutually exclusive, probabilities can be combined across leaves without introducing assumptions of the independence or otherwise of the attributes.

conquer techniques, allow the relevance of different attributes to be conditional on the outcomes of earlier tests.

- *Flexibility*—These methods cope successfully with both continuous and discrete attributes.

On the downside, their main disadvantage is

- *Restricted Formalism*—Each test is limited to a straightforward division based on a single attribute.

The following paragraphs examine these points in more detail.

The success of the expert systems field has demonstrated the validity of the "knowledge is power" thesis that underpins this branch of AI [10]. Experience in this area has also brought home the need to be able to justify the way decisions are arrived at. From this perspective, it is not sufficient just to reach correct decisions; human decision-makers must be able to understand the basis for such decisions. In a medical application, for instance, a practitioner may prefer a decisionmaking aid that is correct 85% of the time, but whose decisions can be related to his own medical knowledge, rather than an inscrutable "black box" that is correct 90% of the time. Decision trees represent classification knowledge symbolically, different conjunctions of conditions being associated with nominated classes. In contrast, many statistical classifiers and all connectionist networks represent classification information as a collection of numbers. It is not surprising that the former can often be more intelligible to a human user.

The process of classifying a case by a decision tree amounts to tracing out a path from the root of the tree to a leaf. The tests along each such path are dependent on the outcomes of previous tests. In the decision tree given earlier, for example, the attribute *humidity* is relevant only when it has been established that *outlook* is *sunny*. The ability to change contexts increases not only the descriptive power of the formalism, but its accuracy as well. This can be demonstrated with reference to the training set of Table I. When the discrete attributes are suitably encoded (for example, *overcast* becomes 0, *sunny* becomes 1, and so on) the training set can be input to a least-squares regression program. The resulting linear model misclassifies three of the objects in the training set; no linear combination of the attributes satisfactorily captures their different relevances in different contexts.

Many common statistical classification techniques deal effectively with either continuous or discrete attributes, but not both. Model-fitting methods such as regression usually require discrete attributes to be encoded as numbers on a continuous scale. Any such transformation introduces spatial connotations that may affect the outcome. The previous encoding, for instance, makes *overcast* closer to *sunny* than to *rain*. Bayesian methods, on the other hand, require that all attributes have discrete values, so continuous attributes must be divided into subranges. Decision-tree methods, however, cope with attributes of both types by finding suitable (context-dependent) thresholds for continuous attributes where needed.[4]

On the negative side, tree-based methods are totally dependent on being provided with an appropriate collection of attributes. "Appropriate" does not just mean "sufficient". Consider a task with two continuous attributes, X and Y, where the class of an object is determined by whether or not its value of X is greater than its value of Y. These attributes are clearly sufficient to determine an object's class, but trees constructed from training sets of such objects will attempt, cumbersomely, to approximate the line $X = Y$ by horizontal and vertical segments.[5] The restriction of tests to functions of a single attribute means that, in cases like this, the task formalizer should include redundant attributes relevant to the classification. (In this example, such a redundant attribute might be $X - Y$.) Notice, however, that inclusion of redundant attributes does not hamper tree-based methods (except for a linear increase in computation), whereas it can pose serious problems for least-square or Bayesian methods that assume some form of independence of the attributes.

Tree-based methods may be inappropriate, therefore, where sensible tests are likely to depend on combinations of attributes that cannot be specified in advance. Such situations include tasks where a number of factors relevant to classification must be weighed against each other. As an illustration, we examined a rain-prediction task in which objects are described in terms of 35 numerical attributes. Regression was used to produce a linear model that accurately classified 76% of several thousand unseen cases. From the same training set, C4 produced a messy tree that was correct for 71% of the same unseen cases, a significantly worse result.

Several studies have been published comparing tree-based methods, other AI learning techniques and common statistical approaches. As a few examples,

- Cestnik *et al.* [6] and Clark and Niblett [8] evaluate programs that grow trees and generate rules against simple Bayesian classifiers in several medical domains. No technique was consistently better, all performing at about the level of a human expert.
- Another trial conducted by Kinney and Murphy [15] compared discriminant analysis with a simple implementation of ID3 in a cardiology application. Both methods were weak (suggesting insufficient or poor-quality data), but the tree-based method actually degraded as more data was made available.[6]

[4] In fact, the flexibility to develop different thresholds at different parts of the tree can be a mixed blessing. Once a tree has been constructed, it is often necessary to check whether similar but different thresholds of a continuous attribute can be collapsed into a single threshold.

[5] Shepherd, Piper, and Rutovitz [44] give a demonstration of the clumsy trees that can result from trying to approximate such decision surfaces by hyperplanes.

[6] The implementation of ID3 that was used in these experiments did not incorporate pruning.

- In a medical domain for which large volumes of training data were available, trees developed by C4 rivalled the accuracy of a mature expert system whose rules were derived by interviewing human endocrinologists [33].
- Carter and Catlett [5] compared sophisticated decision-tree methods with the discriminant analysis methods employed by a bank for assessing credit applications. The trees were significantly more accurate but, in fairness, the bank was attempting to maximize utility rather than accuracy.
- Many studies have found that decision-tree and other divide-and-conquer methods rate favorably against other AI learning methods such as connectionist networks [47], specific-to-general algorithms [24], [39] and genetic algorithms [35]. This should not be interpreted as a claim that decision-tree methods are in any sense optimal. In fact, Weiss and Kapouleas [47] show that search-intensive learning methods can sometimes produce simpler and more accurate classifiers, but at the expense of exponentially greater computation.

These studies are most valuable in helping to characterize more exactly the sorts of problems for which decision-tree methods are suitable.

VI. Conclusion

Decision trees provide a powerful formalism for representing comprehensible, accurate classifiers. The top-down method of constructing them is computationally undemanding. When they are used, information regarding attribute values is sought only as required, making them attractive in a diagnostic context. They have also been found useful as components of intelligent systems, for instance:

- Leech [16] describes a process control application in which the use of decision tree methods, together with statistical and OR techniques, led to savings of several million dollars per year.
- Buntine and Stirling [3] have reported a planning application that employs decision trees at choice points to generate a plan from a grammar of possible plans.

The current maturity of the technology should not be taken to imply that it is static. Four examples of recent initiatives follow.

Incremental Tree Construction Methods: In a real-world situation it is quite likely that, after a tree has been constructed, additional training objects will become available. We could simply discard the current decision tree and construct a replacement using the enlarged training set, but it is aesthetically and economically appealing to try to modify the existing tree. Utgoff [46] has developed an incremental algorithm that produces the same tree as it would have found if all the examples had been available at once.

Multi level Induction: One weakness of many classification methods, including decision trees, is that they proceed directly from given data to conclusions without forming the intermediate hypotheses that often characterize human classification. Shapiro [43] shows that the definition of a subconcept hierarchy can lead to much smaller, more comprehensible trees. Muggleton [22] describes exciting work aimed at automating the development of such hierarchies.

Less Restrictive Formalisms: In the previous section it was noted that the expressive power of decision trees is limited by the restriction that each test involve a single attribute. There is much interest in seeing how this restriction can be relaxed without losing the concomitant benefits of computational economy and conciseness. Pagallo and Haussler [25] explore the issue of learning new attributes that are conjunctions of single-attribute tests, and Chan [7] tries logical combinations of Boolean attributes. The effect of both these extensions is to reduce the complexity of trees without compromising their intelligibility and, as a by-product, to generate more accurate trees. Manago [17] goes further; he transforms decision trees to first-order constructs by allowing the introduction of variables in tests and the description of objects by frames.

Tree-Structured Models: There is no reason why tree-based methods need be limited to classification. Brieman *et al.* [2] note that storing values rather than classes at leaves results in a tree-structured function. I have carried out preliminary experiments in which the leaves hold regression models rather than simple values. The intention here is to synthesize a function by dividing up the description space into subregions, in each of which a separate linear model provides a reasonable approximation to the target function.

Acknowledgment

I am grateful to the organizers and participants of the Illinois Interdisciplinary Workshop on Decision Making for many comments that helped to improve this paper.

References

[1] B. Arbab and D. Michie, Generating rules from examples, *Proc. 9th Int. Joint Conf. Artificial Intell.*, (Los Altos: Morgan Kaufmann), 1985.

[2] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees.* Belmont, CA: Wadsworth, 1984.

[3] W. Buntine and D. Stirling, "Interactive induction," *Proc. IEEE Conf. AI Appl.*, San Diego, CA, 1988.

[4] J. G. Carbonell, R. S. Michalski, and T. M. Mitchell, "An overview of machine learning," in *Machine Learning: An AI Approach*, R. S. Michalski, J. G. Carbonell and T. M. Mitchell, Eds. Palo Alto, CA: Tioga, 1983.

[5] C. Carter and J. Catlett, Assessing credit card applications using machine learning, *IEEE Expert*, vol. 2, no. 3, pp. 71–79, 1987.

[6] B. Cestnik, I. Kononenko, and I. Bratko, "ASSISTANT 86: A knowledge-elicitation tool for sophisticated users," in *Progress in Machine Learning*, I. Bratko and N. Lavrač, Eds. Wilmslow, UK: Sigma Press, 1987.

[7] P. K. Chan, "Inductive learning with BCT," in *Proc. Sixth Int. Workshop on Machine learning*, A. M. Segre, Ed. Los Altos, CA: Morgan Kaufmann, 1989.

[8] P. Clark, and T. Niblett, "The CN2 induction algorithm," *Machine Learning*, vol. 3, no. 4, pp. 261–284, 1989.

[9] E. A. Feigenbaum and H. A. Simon, "Performance of a reading task by an elementary perceiving and memorizing program," *Behavioral Sci.*, vol. 8, no. 3, 1963.

[10] E. A. Feigenbaum, "Expert systems in the 1980s," in *Infotech State of the Art Report on Machine Intelligence*, A. Bond, Ed. Maidenhead, UK: Pergamon-Infotech, 1981.

[11] A. E. Hart, "Experience in the use of an inductive system in knowledge engineering," in *Research and Development in Expert Systems*, M. A. Bramer, Ed. Cambridge: Cambridge Univ. Press, 1985.

[12] E. B. Hunt, *Concept Learning: An Information Processing Problem*. New York: Wiley, 1962.

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

[14] L. Hyafil, and R. L. Rivest, "Constructing optimal binary decision trees is NP-complete," *Information Processing Letters*, vol. 5, no. 1, 1976.

[15] E. L. Kinney and D. D. Murphy, Comparison of the ID3 algorithm versus discriminant analysis for performing feature selection, *Computers and Biomedical Research*, vol. 20, pp. 467–476, 1987.

[16] W. J. Leech, "A rule based process control method with feedback," *Proc. Instrument Soc. of Amer. Conf.*, Houston, TX, pp. 169–175, 1986.

[17] M. Manago, "Knowledge intensive induction," in *Proc. Sixth Int. Workshop on Machine Learning*, A. M. Segre, Ed. Los Altos, CA: Morgan Kaufmann, 1989.

[18] R. S. Michalski, "Pattern recognition as rule-guided inductive inference," *IEEE Trans. Pattern Anal. Machine Intell.* vol. 2, no. 4, 1980.

[19] R. S. Michalski, I. Mozetič, J. Hong, and N. Lavrač, "The multipurpose incremental learning system AQ15 and its testing application to three medical domains," in *Proc. Fifth Nat. Conf. Artificial Intell.*, Los Altos, CA: Morgan Kaufmann, 1986.

[20] J. Mingers, "Expert systems—rule induction with statistical data," *J. Oper. Res. Soc.*, vol. 38, no. 1, pp. 39–47, 1987.

[21] ____, "An empirical comparison of selection methods for decision-tree induction," *Machine Learning*, vol. 3, no. 4, pp. 261–284, 1989.

[22] S. Muggleton, "Structuring knowledge by asking questions," in *Proc. 10th Int. Joint Conf. Artificial Intell.*, Los Altos, CA: Morgan Kaufmann, 1987.

[23] N. J. Nilsson, *Learning Machines*. New York: McGraw-Hill, 1965.

[24] P. O'Rorke, "A comparative study of inductive learning systems AQ11P and ID3 using a chess endgame test problem," Tech. Rep. UIUCDCS-F-82-899, Dept. Comput. Sci., Univ. Illinois at Urbana-Champaign, 1982.

[25] G. Pagallo and D. Haussler, "Two algorithms that learn DNF by discovering relevant features," in *Proc. Sixth Int. Workshop on Machine Learning*, A. M. Segre, Ed. Los Altos, CA: Morgan Kaufmann, 1989.

[26] J. Pearl, On the connection between the complexity and credibility of inferred models," *Int. J. Gen. Syst.*, vol. 4, 1978.

[27] J. R. Quinlan, "Discovering rules by induction from large collections of examples," in *Expert Systems in the Micro Electronic Age*, D. Michie, Ed. Edinburgh Univ. Press, 1979.

[28] ____, "Learning efficient classification procedures and their application to chess endgames," in *Machine Learning: an AI Approach*, R. S. Michalski, J. G. Carbonell and T. M. Mitchell, Eds. Palo Alto, CA: Tioga, 1983.

[29] ____, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[30] ____, "The effect of noise on concept learning," in *Machine Learning: an AI Approach*, vol. 2, R. S. Michalski, J. G. Carbonell and T. M. Mitchell, Eds. Los Altos CA,: Morgan Kaufmann.

[31] ____, "Probabilistic decision trees," in *Proc. Fourth Int. Workshop on Machine Learning*, P. Langley, Ed. Los Altos, CA: Morgan Kaufmann, 1987.

[32] ____, "Simplifying decision trees," *Int. J. Man–Machine Studies*, vol. 27, pp. 221–234, 1987.

[33] ____, "Induction, knowledge, and expert systems," in *Artificial Intelligence Developments and Applications*, J. S. Gero and R. Stanton, Eds. Amsterdam: North Holland, 1987.

[34] ____, "Decision trees and multi-valued attributes," in *Machine Intelligence* vol. 11, J. E. Hayes, D. Michie and J. Richards, Eds. Oxford, UK: Oxford Univ. Press, 1988.

[35] ____, "An empirical comparison of genetic and decision tree classifiers," *Proc. Fifth Int. Machine Learning Conf.*, J. Laird, Ed. Los Altos, CA: Morgan Kaufmann, 1988.

[36] J. R. Quinlan and R. L. Rivest, "Inferring decision trees using the minimum description length principle," *Information and Computation*, vol. 80, no. 3, pp. 227–248, 1989.

[37] J. R. Quinlan, "Unknown attribute values in induction," in *Proc. Sixth Int. Workshop on Machine Learning*, A. M. Segre, Ed. Los Altos, CA: Morgan Kaufmann, 1989.

[38] L. A. Rendell, "A general framework for induction and a study of selective induction," *Machine Learning* vol. 1, no. 2, pp. 177–226, 1986.

[39] L. A. Rendell, H. H. Cho, and R. Seshu, "Improving the design of similarity-based rule-learning systems," *Int. J. Expert Syst.*, vol. 2, no. 1, pp. 97–133, 1989.

[40] C. A. Sammut, Concept development for expert system knowledge bases, *Austral. Comput. J.*, vol. 17, no. 1, 1985.

[41] A. Samuel "Some studies in machine learning using the game of checkers II: Recent progress, *IBM J. Res. Development* vol. 11, no. 6, 1967.

[42] J. C. Schlimmer and R. H. Granger, "Beyond incremental processing: tracking concept drift," *Proc. Fifth Nat. Conf. Artificial Intell.* (Los Altos, CA: Morgan Kaufmann), 1986.

[43] A. D. Shapiro, *Structured Induction in Expert Systems*. Reading, MA: Addison-Wesley, 1987.

[44] B. Shepherd, J. Piper, and D. Rutovitz, "Comparison of ACLS and classical linear methods in a biological application," in *Machine Intelligence* vol. 11, J. E. Hayes, D. Michie and J. Richards, Eds. Oxford, UK: Oxford Univ. Press, 1988.

[45] S. Spangler, U. M. Fayyad, and R. Uthurusamy, (1989), "Induction of decision trees from inconclusive data," in *Proc. Sixth Int. Workshop on Machine Learning*, A. M. Segre, Ed., (Los Altos, CA: Morgan Kaufmann), 1989.

[46] P. E. Utgoff, "Incremental induction of decision trees," *Machine Learning*, vol. 4, no. 2, 1989.

[47] S. M. Weiss, and I. Kapouleas, "An empirical comparison of pattern recognition, neural nets, and machine learning classification methods," *Proc. 11th Int. Joint Conf. Artificial Intell.*, (Los Altos, CA: Morgan Kaufmann), 1989.

[48] P. H. Winston, "Learning structural descriptions from examples," in *The Psychology of Computer Vision*, P. H. Winston, Ed. New York: McGraw-Hill, 1975.

[49] ____, *Artificial Intelligence*, (second ed.). Reading, MA: Addison-Wesley, 1984.

J. Ross Quinlan obtained his B.Sc. from the University of Sydney, Sydney, Australia, in 1965 and his Ph.D. from the newly formed Computer Science Group at the University of Washington, Seattle, WA, in 1968.

He is Professor of Computer Science and Head of the Basser Department of Computer Science in the University of Sydney. He has held appointments at Carnegie-Mellon University, the Rand Corporation, the New South Wales Institute of Technology and the Massachusetts Institute of Technology. He established the Sydney Expert Systems Group in 1983 and initiated the series of Australian Conferences on Applications of Expert Systems in 1985; selected papers from these conferences have appeared in *Applications of Expert Systems, vols. 1 & 2.* Addison-Wesley, 1988, 1989. His research focuses on inductive techniques to assist the knowledge acquisition process.