Machine Learning 1: 81–106, 1986
© 1986 Kluwer Academic Publishers, Boston – Manufactured in The Netherlands

Induction of Decision Trees

J.R. QUINLAN (munnari!nswitgould.oz!quinlan@seismo.css.gov)
Centre for Advanced Computing Sciences, New South Wales Institute of Technology, Sydney 2007, Australia

(Received August 1, 1985)

Key words: classification, induction, decision trees, information theory, knowledge acquisition, expert systems

Abstract. The technology for building knowledge-based systems by inductive inference from examples has been demonstrated successfully in several practical applications. This paper summarizes an approach to synthesizing decision trees that has been used in a variety of systems, and it describes one such system, ID3, in detail. Results from recent studies show ways in which the methodology can be modified to deal with information that is noisy and/or incomplete. A reported shortcoming of the basic algorithm is discussed and two means of overcoming it are compared. The paper concludes with illustrations of current research directions.

1. Introduction

Since artificial intelligence first achieved recognition as a discipline in the mid 1950's, machine learning has been a central research area. Two reasons can be given for this prominence. The ability to learn is a hallmark of intelligent behavior, so any attempt to understand intelligence as a phenomenon must include an understanding of learning. More concretely, learning provides a potential methodology for building high-performance systems.

Research on learning is made up of diverse subfields. At one extreme there are adaptive systems that monitor their own performance and attempt to improve it by adjusting internal parameters. This approach, characteristic of a large proportion of the early learning work, produced self-improving programs for playing games (Samuel, 1967), balancing poles (Michie, 1982), solving problems (Quinlan, 1969) and many other domains. A quite different approach sees learning as the acquisition of structured knowledge in the form of concepts (Hunt, 1962; Winston, 1975), discrimination nets (Feigenbaum and Simon, 1963), or production rules (Buchanan, 1978).

The practical importance of machine learning of this latter kind has been underlin-

ed by the advent of knowledge-based expert systems. As their name suggests, these systems are powered by knowledge that is represented explicitly rather than being implicit in algorithms. The knowledge needed to drive the pioneering expert systems was codified through protracted interaction between a domain specialist and a knowledge engineer. While the typical rate of knowledge elucidation by this method is a few rules per man day, an expert system for a complex task may require hundreds or even thousands of such rules. It is obvious that the interview approach to knowledge acquisition cannot keep pace with the burgeoning demand for expert systems; Feigenbaum (1981) terms this the 'bottleneck' problem. This perception has stimulated the investigation of machine learning methods as a means of explicating knowledge (Michie, 1983).

This paper focusses on one microcosm of machine learning and on a family of learning systems that have been used to build knowledge-based systems of a simple kind. Section 2 outlines the features of this family and introduces its members. All these systems address the same task of inducing decision trees from examples. After a more complete specification of this task, one system (ID3) is described in detail in Section 4. Sections 5 and 6 present extensions to ID3 that enable it to cope with noisy and incomplete information. A review of a central facet of the induction algorithm reveals possible improvements that are set out in Section 7. The paper concludes with two novel initiatives that give some idea of the directions in which the family may grow.

2. The TDIDT family of learning systems

Carbonell, Michalski and Mitchell (1983) identify three principal dimensions along which machine learning systems can be classified:

- the underlying learning strategies used;
- the representation of knowledge acquired by the system; and
- the application domain of the system.

This paper is concerned with a family of learning systems that have strong common bonds in these dimensions.

Taking these features in reverse order, the *application domain* of these systems is not limited to any particular area of intellectual activity such as Chemistry or Chess; they can be applied to any such area. While they are thus general-purpose systems, the applications that they address all involve *classification*. The product of learning is a piece of procedural knowledge that can assign a hitherto-unseen object to one of a specified number of disjoint classes. Examples of classification tasks are:

1. the diagnosis of a medical condition from symptoms, in which the classes could be either the various disease states or the possible therapies;
2. determining the game-theoretic value of a chess position, with the classes *won for white*, *lost for white*, and *drawn*; and
3. deciding from atmospheric observations whether a severe thunderstorm is unlikely, possible or probable.

It might appear that classification tasks are only a minuscule subset of procedural tasks, but even activities such as robot planning can be recast as classification problems (Dechter and Michie, 1985).

The members of this family are sharply characterized by their *representation of acquired knowledge* as decision trees. This is a relatively simple knowledge formalism that lacks the expressive power of semantic networks or other first-order representations. As a consequence of this simplicity, the learning methodologies used in the TDIDT family are considerably less complex than those employed in systems that can express the results of their learning in a more powerful language. Nevertheless, it is still possible to generate knowledge in the form of decision trees that is capable of solving difficult problems of practical significance.

The *underlying strategy* is non-incremental learning from examples. The systems are presented with a set of cases relevant to a classification task and develop a decision tree from the top down, guided by frequency information in the examples but not by the particular order in which the examples are given. This contrasts with incremental methods such as that employed in MARVIN (Sammut, 1985), in which a dialog is carried on with an instructor to 'debug' partially correct concepts, and that used by Winston (1975), in which examples are analyzed one at a time, each producing a small change in the developing concept; in both of these systems, the order in which examples are presented is most important. The systems described here search for patterns in the given examples and so must be able to examine and re-examine all of them at many stages during learning. Other well-known programs that share this data-driven approach include BACON (Langley, Bradshaw and Simon, 1983) and INDUCE (Michalski, 1980).

In summary, then, the systems described here develop decision trees for classification tasks. These trees are constructed beginning with the root of the tree and proceeding down to its leaves. The family's palindromic name emphasizes that its members carry out the *T*op-*D*own *I*nduction of *D*ecision *T*rees.

The example objects from which a classification rule is developed are known only through their values of a set of properties or attributes, and the decision trees in turn are expressed in terms of these same attributes. The examples themselves can be assembled in two ways. They might come from an existing database that forms a history of observations, such as patient records in some area of medicine that have accumulated at a diagnosis center. Objects of this kind give a reliable statistical picture but, since they are not organized in any way, they may be redundant or omit

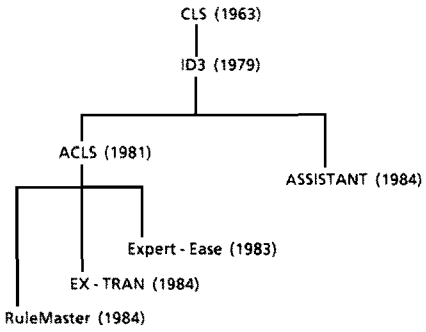

Figure 1. The TDIDT family tree.

uncommon cases that have not been encountered during the period of record-keeping. On the other hand, the objects might be a carefully culled set of tutorial examples prepared by a domain expert, each with some particular relevance to a complete and correct classification rule. The expert might take pains to avoid redundancy and to include examples of rare cases. While the family of systems will deal with collections of either kind in a satisfactory way, it should be mentioned that earlier TDIDT systems were designed with the 'historical record' approach in mind, but all systems described here are now often used with tutorial sets (Michie, 1985).

Figure 1 shows a family tree of the TDIDT systems. The patriarch of this family is Hunt's Concept Learning System framework (Hunt, Marin and Stone, 1966). CLS constructs a decision tree that attempts to minimize the cost of classifying an object. This cost has components of two types: the measurement cost of determining the value of property A exhibited by the object, and the misclassification cost of deciding that the object belongs to class J when its real class is K. CLS uses a lookahead strategy similar to minimax. At each stage, CLS explores the space of possible decision trees to a fixed depth, chooses an action to minimize cost in this limited space, then moves one level down in the tree. Depending on the depth of lookahead chosen, CLS can require a substantial amount of computation, but has been able to unearth subtle patterns in the objects shown to it.

ID3 (Quinlan, 1979, 1983a) is one of a series of programs developed from CLS in response to a challenging induction task posed by Donald Michie, viz. to decide from pattern-based features alone whether a particular chess position in the King-Rook vs King-Knight endgame is lost for the Knight's side in a fixed number of ply. A full description of ID3 appears in Section 4, so it is sufficient to note here that it embeds a tree-building method in an iterative outer shell, and abandons the cost-driven lookahead of CLS with an information-driven evaluation function.

ACLS (Paterson and Niblett, 1983) is a generalization of ID3. CLS and ID3 both require that each property used to describe objects has only values from a specified set. In addition to properties of this type, ACLS permits properties that have

unrestricted integer values. The capacity to deal with attributes of this kind has allowed ACLS to be applied to difficult tasks such as image recognition (Shepherd, 1983).

ASSISTANT (Kononenko, Bratko and Roskar, 1984) also acknowledges ID3 as its direct ancestor. It differs from ID3 in many ways, some of which are discussed in detail in later sections. ASSISTANT further generalizes on the integer-valued attributes of ACLS by permitting attributes with continuous (real) values. Rather than insisting that the classes be disjoint, ASSISTANT allows them to form a hierarchy, so that one class may be a finer division of another. ASSISTANT does not form a decision tree iteratively in the manner of ID3, but does include algorithms for choosing a 'good' training set from the objects available. ASSISTANT has been used in several medical domains with promising results.

The bottom-most three systems in the figure are commercial derivatives of ACLS. While they do not significantly advance the underlying theory, they incorporate many user-friendly innovations and utilities that expedite the task of generating and using decision trees. They all have industrial successes to their credit. Westinghouse Electric's Water Reactor Division, for example, points to a fuel-enrichment application in which the company was able to boost revenue by 'more than ten million dollars per annum' through the use of one of them.[1]

3. The induction task

We now give a more precise statement of the induction task. The basis is a universe of *objects* that are described in terms of a collection of *attributes*. Each attribute measures some important feature of an object and will be limited here to taking a (usually small) set of discrete, mutually exclusive values. For example, if the objects were Saturday mornings and the classification task involved the weather, attributes might be

outlook, with values {sunny, overcast, rain}
temperature, with values {cool, mild, hot}
humidity, with values {high, normal}
windy, with values {true, false}

Taken together, the attributes provide a zeroth-order language for characterizing objects in the universe. A particular Saturday morning might be described as

outlook: overcast
temperature: cool
humidity: normal
windy: false

[1] Letter cited in the journal *Expert Systems* (January, 1985), p. 20.

Each object in the universe belongs to one of a set of mutually exclusive *classes*. To simplify the following treatment, we will assume that there are only two such classes denoted P and N, although the extension to any number of classes is not difficult. In two-class induction tasks, objects of class P and N are sometimes referred to as *positive instances* and *negative instances*, respectively, of the concept being learned.

The other major ingredient is a *training set* of objects whose class is known. The induction task is to develop a *classification rule* that can determine the class of any object from its values of the attributes. The immediate question is whether or not the attributes provide sufficient information to do this. In particular, if the training set contains two objects that have identical values for each attribute and yet belong to different classes, it is clearly impossible to differentiate between these objects with reference only to the given attributes. In such a case attributes will be termed *inadequate* for the training set and hence for the induction task.

As mentioned above, a classification rule will be expressed as a decision tree. Table 1 shows a small training set that uses the 'Saturday morning' attributes. Each object's value of each attribute is shown, together with the class of the object (here, class P mornings are suitable for some unspecified activity). A decision tree that correctly classifies each object in the training set is given in Figure 2. Leaves of a decision tree are class names, other nodes represent attribute-based tests with a branch for each possible outcome. In order to classify an object, we start at the root of the tree, evaluate the test, and take the branch appropriate to the outcome. The process continues until a leaf is encountered, at which time the object is asserted to belong to

Table 1. A small training set

No.	Attributes				Class
	Outlook	Temperature	Humidity	Windy	
1	sunny	hot	high	false	N
2	sunny	hot	high	true	N
3	overcast	hot	high	false	P
4	rain	mild	high	false	P
5	rain	cool	normal	false	P
6	rain	cool	normal	true	N
7	overcast	cool	normal	true	P
8	sunny	mild	high	false	N
9	sunny	cool	normal	false	P
10	rain	mild	normal	false	P
11	sunny	mild	normal	true	P
12	overcast	mild	high	true	P
13	overcast	hot	normal	false	P
14	rain	mild	high	true	N

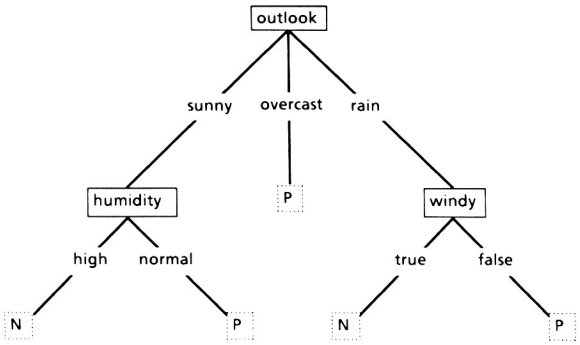

Figure 2. A simple decision tree

the class named by the leaf. Taking the decision tree of Figure 2, this process concludes that the object which appeared as an example at the start of this section, and which is not a member of the training set, should belong to class P. Notice that only a subset of the attributes may be encountered on a particular path from the root of the decision tree to a leaf; in this case, only the outlook attribute is tested before determining the class.

If the attributes are adequate, it is always possible to construct a decision tree that correctly classifies each object in the training set, and usually there are many such correct decision trees. The essence of induction is to move beyond the training set, i.e. to construct a decision tree that correctly classifies not only objects from the training set but other (unseen) objects as well. In order to do this, the decision tree must capture some meaningful relationship between an object's class and its values of the attributes. Given a choice between two decision trees, each of which is correct over the training set, it seems sensible to prefer the simpler one on the grounds that it is more likely to capture structure inherent in the problem. The simpler tree would therefore be expected to classify correctly more objects outside the training set. The decision tree of Figure 3, for instance, is also correct for the training set of Table 1, but its greater complexity makes it suspect as an 'explanation' of the training set.[2]

4. ID3

One approach to the induction task above would be to generate all possible decision trees that correctly classify the training set and to select the simplest of them. The

[2] The preference for simpler trees, presented here as a commonsense application of Occam's Razor, is also supported by analysis. Pearl (1978b) and Quinlan (1983a) have derived upper bounds on the expected error using different formalisms for generalizing from a set of known cases. For a training set of predetermined size, these bounds increase with the complexity of the induced generalization.

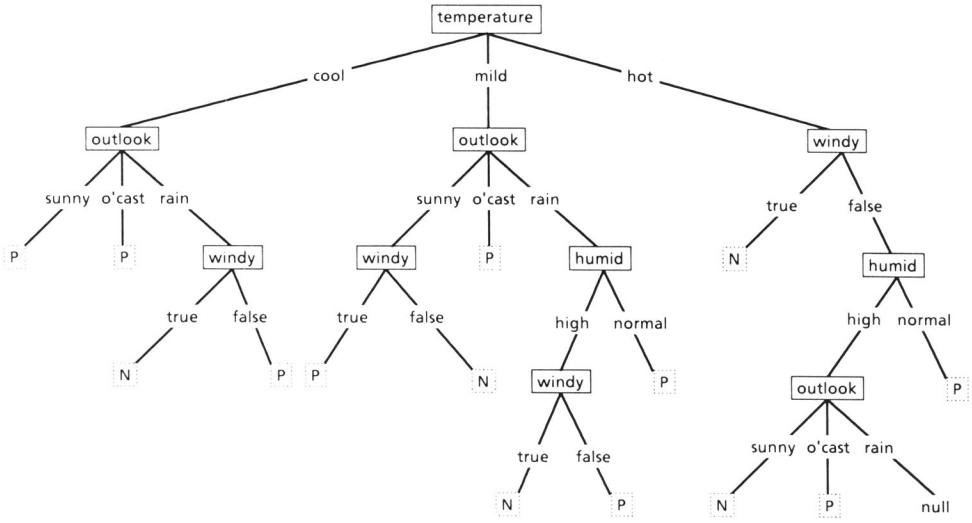

Figure 3. A complex decision tree.

number of such trees is finite but very large, so this approach would only be feasible for small induction tasks. ID3 was designed for the other end of the spectrum, where there are many attributes and the training set contains many objects, but where a reasonably good decision tree is required without much computation. It has generally been found to construct simple decision trees, but the approach it uses cannot guarantee that better trees have not been overlooked.

The basic structure of ID3 is iterative. A subset of the training set called the *window* is chosen at random and a decision tree formed from it; this tree correctly classifies all objects in the window. All other objects in the training set are then classified using the tree. If the tree gives the correct answer for all these objects then it is correct for the entire training set and the process terminates. If not, a selection of the incorrectly classified objects is added to the window and the process continues. In this way, correct decision trees have been found after only a few iterations for training sets of up to thirty thousand objects described in terms of up to 50 attributes. Empirical evidence suggests that a correct decision tree is usually found more quickly by this iterative method than by forming a tree directly from the entire training set. However, O'Keefe (1983) has noted that the iterative framework cannot be guaranteed to converge on a final tree unless the window can grow to include the entire training set. This potential limitation has not yet arisen in practice.

The crux of the problem is how to form a decision tree for an arbitrary collection C of objects. If C is empty or contains only objects of one class, the simplest decision tree is just a leaf labelled with the class. Otherwise, let T be any test on an object with possible outcomes $O_1, O_2, \ldots O_w$. Each object in C will give one of these outcomes for T, so T produces a partition $\{C_1, C_2, \ldots C_w\}$ of C with C_i containing those ob-

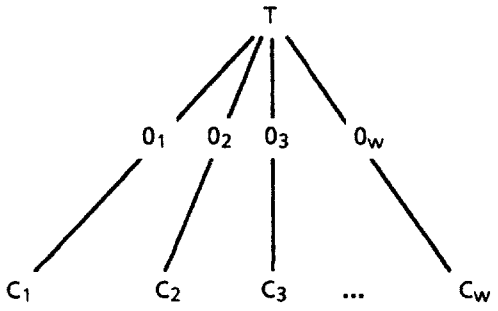

Figure 4. A tree structuring of the objects in C.

jects having outcome O_i. This is represented graphically by the tree form of Figure 4. If each subset C_i in this figure could be replaced by a decision tree for C_i, the result would be a decision tree for all of C. Moreover, so long as two or more C_i's are non-empty, each C_i is smaller than C. In the worst case, this divide-and-conquer strategy will yield single-object subsets that satisfy the one-class requirement for a leaf. Thus, provided that a test can always be found that gives a non-trivial partition of any set of objects, this procedure will always produce a decision tree that correctly classifies each object in C.

The choice of test is crucial if the decision tree is to be simple. For the moment, a test will be restricted to branching on the values of an attribute, so choosing a test comes down to selecting an attribute for the root of the tree. The first induction programs in the ID series used a seat-of-the-pants evaluation function that worked reasonably well. Following a suggestion of Peter Gacs, ID3 adopted an information-based method that depends on two assumptions. Let C contain p objects of class P and n of class N. The assumptions are:

(1) Any correct decision tree for C will classify objects in the same proportion as their representation in C. An arbitrary object will be determined to belong to class P with probability $p/(p+n)$ and to class N with probability $n/(p+n)$.
(2) When a decision tree is used to classify an object, it returns a class. A decision tree can thus be regarded as a source of a message 'P' or 'N', with the expected information needed to generate this message given by

$$I(p, n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

If attribute A with values $\{A_1, A_2, \ldots A_v\}$ is used for the root of the decision tree, it will partition C into $\{C_1, C_2, \ldots C_v\}$ where C_i contains those objects in C that have value A_i of A. Let C_i contain p_i objects of class P and n_i of class N. The expected

information required for the subtree for C_i is $I(p_i, n_i)$. The expected information required for the tree with A as root is then obtained as the weighted average

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

where the weight for the ith branch is the proportion of the objects in C that belong to C_i. The information gained by branching on A is therefore

$$\text{gain}(A) = I(p, n) - E(A)$$

A good rule of thumb would seem to be to choose that attribute to branch on which gains the most information.[3] ID3 examines all candidate attributes and chooses A to maximize gain(A), forms the tree as above, and then uses the same process recursively to form decision trees for the residual subsets $C_1, C_2, \ldots C_v$.

To illustrate the idea, let C be the set of objects in Table 1. Of the 14 objects, 9 are of class P and 5 are of class N, so the information required for classification is

$$I(p, n) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.940 \text{ bits}$$

Now consider the outlook attribute with values {sunny, overcast, rain}. Five of the 14 objects in C have the first value (sunny), two of them from class P and three from class N, so

$$p_1 = 2 \quad n_1 = 3 \quad I(p_1, n_1) = 0.971$$

and similarly

$$p_2 = 4 \quad n_2 = 0 \quad I(p_2, n_2) = 0$$
$$p_3 = 3 \quad n_3 = 2 \quad I(p_3, n_3) = 0.971$$

The expected information requirement after testing this attribute is therefore

$$E(\text{outlook}) = \frac{5}{14} I(p_1, n_1) + \frac{4}{14} I(p_2, n_2) + \frac{5}{14} I(p_3, n_3)$$
$$= 0.694 \text{ bits}$$

[3] Since I(p,n) is constant for all attributes, maximizing the gain is equivalent to minimizing E(A), which is the mutual information of the attribute A and the class. Pearl (1978a) contains an excellent account of the rationale of information-based heuristics.

The gain of this attribute is then

$$\text{gain(outlook)} = 0.940 - \text{E(outlook)} = 0.246 \text{ bits}$$

Similar analysis gives

$$\text{gain(temperature)} = 0.029 \text{ bits}$$
$$\text{gain(humidity)} = 0.151 \text{ bits}$$
$$\text{gain(windy)} = 0.048 \text{ bits}$$

so the tree-forming method used in ID3 would choose outlook as the attribute for the root of the decision tree. The objects would then be divided into subsets according to their values of the outlook attribute and a decision tree for each subset would be induced in a similar fashion. In fact, Figure 2 shows the actual decision tree generated by ID3 from this training set.

A special case arises if C contains no objects with some particular value A_j of A, giving an empty C_j. ID3 labels such a leaf as 'null' so that it fails to classify any object arriving at that leaf. A better solution would generalize from the set C from which C_j came, and assign this leaf the more frequent class in C.

The worth of ID3's attribute-selecting heuristic can be assessed by the simplicity of the resulting decision trees, or, more to the point, by how well those trees express real relationships between class and attributes as demonstrated by the accuracy with which they classify objects other than those in the training set (their *predictive accuracy*). A straightforward method of assessing this predictive accuracy is to use only part of the given set of objects as a training set, and to check the resulting decision tree on the remainder.

Several experiments of this kind have been carried out. In one domain, 1.4 million chess positions described in terms of 49 binary-valued attributes gave rise to 715 distinct objects divided 65%:35% between the classes. This domain is relatively complex since a correct decision tree for all 715 objects contains about 150 nodes. When training sets containing 20% of these 715 objects were chosen at random, they produced decision trees that correctly classified over 84% of the unseen objects. In another version of the same domain, 39 attributes gave 551 distinct objects with a correct decision tree of similar size; training sets of 20% of these 551 objects gave decision trees of almost identical accuracy. In a simpler domain (1,987 objects with a correct decision tree of 48 nodes), randomly-selected training sets containing 20% of the objects gave decision trees that correctly classified 98% of the unseen objects. In all three cases, it is clear that the decision trees reflect useful (as opposed to random) relationships present in the data.

This discussion of ID3 is rounded off by looking at the computational requirements of the procedure. At each non-leaf node of the decision tree, the gain of each untested attribute A must be determined. This gain in turn depends on the values p_i

and n_i for each value A_i of A, so every object in C must be examined to determine its class and its value of A. Consequently, the computational complexity of the procedure at each such node is $O(|C| \cdot |A|)$, where $|A|$ is the number of attributes above. ID3's total computational requirement per iteration is thus proportional to the product of the size of the training set, the number of attributes and the number of non-leaf nodes in the decision tree. The same relationship appears to extend to the entire induction process, even when several iterations are performed. No exponential growth in time or space has been observed as the dimensions of the induction task increase, so the technique can be applied to large tasks.

5. Noise

So far, the information supplied in the training set has been assumed to be entirely accurate. Sadly, induction tasks based on real-world data are unlikely to find this assumption to be tenable. The description of objects may include attributes based on measurements or subjective judgements, both of which may give rise to errors in the values of attributes. Some of the objects in the training set may even have been misclassified. To illustrate the idea, consider the task of developing a classification rule for medical diagnosis from a collection of patient histories. An attribute might test for the presence of some substance in the blood and will almost inevitably give false positive or negative readings some of the time. Another attribute might assess the patient's build as slight, medium, or heavy, and different assessors may apply different criteria. Finally, the collection of case histories will probably include some patients for whom an incorrect diagnosis was made, with consequent errors in the class information provided in the training set.

What problems might errors of these kinds pose for the tree-building procedure described earlier? Consider again the small training set in Table 1, and suppose now that attribute outlook of object 1 is incorrectly recorded as overcast. Objects 1 and 3 will then have identical descriptions but belong to different classes, so the attributes become inadequate for this training set. The attributes will also become inadequate if attribute windy of object 4 is corrupted to true, because that object will then conflict with object 14. Finally, the initial training set can be accounted for by the simple decision tree of Figure 2 containing 8 nodes. Suppose that the class of object 3 were corrupted to N. A correct decision tree for this corrupted training set would now have to explain the apparent special case of object 3. The smallest such tree contains twelve nodes, half again as complex as the 'real' tree. These illustrations highlight two problems: errors in the training set may cause the attributes to become inadequate, or may lead to decision trees of spurious complexity.

Non-systematic errors of this kind in either the values of attributes or class information are usually referred to as *noise*. Two modifications are required if the tree-building algorithm is to be able to operate with a noise-affected training set.

(1) The algorithm must be able to work with inadequate attributes, because noise can cause even the most comprehensive set of attributes to appear inadequate.
(2) The algorithm must be able to decide that testing further attributes will not improve the predictive accuracy of the decision tree. In the last example above, it should refrain from increasing the complexity of the decision tree to accommodate a single noise-generated special case.

We start with the second requirement of deciding when an attribute is really relevant to classification. Let C be a collection of objects containing representatives of both classes, and let A be an attribute with random values that produces subsets $\{C_1, C_2, \ldots C_v\}$. Unless the proportion of class P objects in each of the C_i is exactly the same as the proportion of class P objects in C itself, branching on attribute A will give an apparent information gain. It will therefore appear that testing attribute A is a sensible step, even though the values of A are random and so cannot help to classify the objects in C.

One solution to this dilemma might be to require that the information gain of any tested attribute exceeds some absolute or percentage threshold. Experiments with this approach suggest that a threshold large enough to screen out irrelevant attributes also excludes attributes that are relevant, and the performance of the tree-building procedure is degraded in the noise-free case.

An alternative method based on the chi-square test for stochastic independence has been found to be more useful. In the previous notation, suppose attribute A produces subsets $\{C_1, C_2, \ldots C_v\}$ of C, where C_i contains p_i and n_i objects of class P and N, respectively. If the value of A is irrelevant to the class of an object in C, the expected value p'_i of p_i should be

$$p'_i = p \cdot \frac{p_i + n_i}{p + n}$$

If n'_i is the corresponding expected value of n_i, the statistic

$$\sum_{i=1}^{v} \frac{(p_i - p'_i)^2}{p'_i} + \frac{(n_i - n'_i)^2}{n'_i}$$

is approximately chi-square with v-1 degrees of freedom. Provided that none of the values p'_i or n'_i are very small, this statistic can be used to determine the confidence with which one can reject the hypothesis that A is independent of the class of objects in C (Hogg and Craig, 1970). The tree-building procedure can then be modified to prevent testing any attribute whose irrelevance cannot be rejected with a very high (e.g. 99%) confidence level. This has been found effective in preventing over-

complex trees that attempt to 'fit the noise' without affecting performance of the procedure in the noise-free case.[4]

Turning now to the first requirement, we see that the following situation can arise: a collection of C objects may contain representatives of both classes, yet further testing of C may be ruled out, either because the attributes are inadequate and unable to distinguish among the objects in C, or because each attribute has been judged to be irrelevant to the class of objects in C. In this situation it is necessary to produce a leaf labelled with class information, but the objects in C are not all of the same class.

Two possibilities suggest themselves. The notion of class could be generalized to allow the value $p/(p+n)$ in the interval (0,1), a class of 0.8 (say) being interpreted as 'belonging to class P with probability 0.8'. An alternative approach would be to opt for the more numerous class, i.e. to assign the leaf to class P if $p > n$, to class N if $p < n$, and to either if $p = n$. The first approach minimizes the sum of the squares of the error over objects in C, while the second minimizes the sum of the absolute errors over objects in C. If the aim is to minimize expected error, the second approach might be anticipated to be superior, and indeed this has been found to be the case.

Several studies have been carried out to see how this modified procedure holds up under varying levels of noise (Quinlan 1983b, 1985a). One such study is outlined here, based on the earlier-mentioned task with 551 objects and 39 binary-valued attributes. In each experiment, the whole set of objects was artificially corrupted as described below and used as a training set to produce a decision tree. Each object was then corrupted anew, classified by this tree and the error rate determined. This process was repeated twenty times to give more reliable averages.

In this study, values were corrupted as follows. A noise level of n percent applied to a value meant that, with probability n percent, the true value was replaced by a value chosen at random from among the values that could have appeared.[5] Table 2 shows the results when noise levels varying from 5% to 100% were applied to the values of the most noise-sensitive attribute, to the values of all attributes simultaneously, and to the class information. This table demonstrates the quite different forms of degradation observed. Destroying class information produces a linear increase in error so that, when all class information is noise, the resulting decision tree classifies objects entirely randomly. Noise in a single attribute does not have a dramatic effect. Noise in all attributes together, however, leads to a relatively rapid increase in error which reaches a peak and declines. The peak is somewhat inter-

[4] ASSISTANT uses an information-based measure to perform much the same function, but no comparative results are available to date.

[5] It might seem that the value should be replaced by an incorrect value. Consider, however, the case of a two-valued attribute corrupted with 100% noise. If the value of each object were replaced by the (only) incorrect value, the initial attribute will have been merely inverted with no loss of information.

Table 2. Error rates produced by noise in a single attribute, all attributes, and class information

Noise level	Single attribute	All attributes	Class information
5%	1.3%	11.9%	2.6%
10%	2.5%	18.9%	5.5%
15%	3.3%	24.6%	8.3%
20%	4.6%	27.8%	9.9%
30%	6.1%	29.5%	14.8%
40%	7.6%	30.3%	18.1%
50%	8.8%	29.2%	21.8%
60%	9.4%	27.5%	26.4%
70%	9.9%	25.9%	27.2%
80%	10.4%	26.0%	29.5%
90%	10.8%	25.6%	34.1%
100%	10.8%	25.9%	49.6%

esting, and can be explained as follows. Let C be a collection of objects containing p from class P and n from class N, respectively. At noise levels around 50%, the algorithm for constructing decision trees may still find relevant attributes to branch on, even though the performance of this tree on unseen but equally noisy objects will be essentially random. Suppose the tree for C classifies objects as class P with probability $p/(n+p)$. The expected error if objects with a similar class distribution to those in C were classified by this tree is given by

$$\frac{p}{p+n} \cdot \left(1 - \frac{p}{p+n}\right) + \frac{n}{p+n} \cdot \left(1 - \frac{n}{p+n}\right) = \frac{2pn}{(p+n)^2}$$

At very high levels of noise, however, the algorithm will find all attributes irrelevant and classify everything as the more frequent class; assume without loss of generality that this class is P. The expected error in this case is

$$\frac{p}{p+n} \cdot 0 + \frac{n}{p+n} \cdot 1 = \frac{n}{p+n}$$

which is less than the above expression since we have assumed that p is greater than n. The decline in error is thus a consequence of the chi-square cutoff coming into play as noise becomes more intense.

The table brings out the point that low levels of noise do not cause the tree-building machinery to fall over a cliff. For this task, a 5% noise level in a single attribute produces a degradation in performance of less than 2%; a 5% noise level in all attributes together produces a 12% degradation in classification performance; while a similar

noise level in class information results in a 3% degradation. Comparable figures have been obtained for other induction tasks.

One interesting point emerged from other experiments in which a correct decision tree formed from an uncorrupted training set was used to classify objects whose descriptions were corrupted. This scenario corresponds to forming a classification rule under controlled and sanitized laboratory conditions, then using it to classify objects in the field. For higher noise levels, the performance of the correct decision tree on corrupted data was found to be inferior to that of an imperfect decision tree formed from data corrupted to a similar level! (This phenomenon has an explanation similar to that given above for the peak in Table 2.) The moral seems to be that it is counter-productive to eliminate noise from the attribute information in the training set if these same attributes will be subject to high noise levels when the induced decision tree is put to use.

6. Unknown attribute values

The previous section examined modifications to the tree-building process that enabled it to deal with noisy or corrupted values. This section is concerned with an allied problem that also arises in practice: unknown attribute values. To continue the previous medical diagnosis example, what should be done when the patient case histories that are to form the training set are incomplete?

One way around the problem attempts to fill in an unknown value by utilizing information provided by context. Using the previous notation, let us suppose that a collection C of objects contains one whose value of attribute A is unknown. ASSISTANT (Kononenko *et al*, 1984) uses a Bayesian formalism to determine the probability that the object has value A_i of A by examining the distribution of values of A in C as a function of their class. Suppose that the object in question belongs to class P. The probability that the object has value A_i for attribute A can be expressed as

$$\text{prob}(A = A_i \mid \text{class} = P) = \frac{\text{prob}(A = A_i \;\&\; \text{class} = P)}{\text{prob}(\text{class} = P)} = \frac{p_i}{p}$$

where the calculation of p_i and p is restricted to those members of C whose value of A is known. Having determined the probability distribution of the unknown value over the possible values of A, this method could either choose the most likely value or divide the object into fractional objects, each with one possible value of A, weighted according to the probabilities above.

Alen Shapiro (private communication) has suggested using a decision-tree approach to determine the unknown values of an attribute. Let C′ be the subset of C consisting of those objects whose value of attribute A is defined. In C′, the original

Table 3. Proportion of times that an unknown attribute value is replaced by an incorrect value

Replacement method	Attribute		
	1	2	3
Bayesian	28%	27%	38%
Decision tree	19%	22%	19%
Most common value	28%	27%	40%

class (P or N) is regarded as another attribute while the value of attribute A becomes the 'class' to be determined. That is, C′ is used to construct a decision tree for determining the value of attribute A from the other attributes and the class. When constructed, this decision tree can be used to 'classify' each object in C − C′ and the result assigned as the unknown value of A.

Although these methods for determining unknown attribute values look good on paper, they give unconvincing results even when only a single value of one attribute is unknown; as might be expected, their performance is much worse when several values of several attributes are unknown. Consider again the 551-object 39-attribute task. We may ask how well the methods perform when asked to fill in a single unknown attribute value. Table 3 shows, for each of the three most important attributes, the proportion of times each method fails to replace an unknown value by its correct value. For comparison, the table also shows the same figure for the simple strategy: always replace an unknown value of an attribute with its most common value. The Bayesian method gives results that are scarcely better than those given by the simple strategy and, while the decision-tree method uses more context and is thereby more accurate, it still gives disappointing results.

Rather than trying to guess unknown attribute values, we could treat 'unknown' as a new possible value for each attribute and deal with it in the same way as other values. This can lead to an anomalous situation, as shown by the following example. Suppose A is an attribute with values $\{A_1, A_2\}$ and let C be a collection of objects such that

$$p_1 = 2 \quad p_2 = 2$$
$$n_1 = 2 \quad n_2 = 2$$

giving a value of 1 bit for E(A). Now let A′ be an identical attribute except that one of the objects with value A_1 of A has an unknown value of A′. A′ has the values $\{A'_1, A'_2, A'_3 = \text{unknown}\}$, so the corresponding values might be

$$p'_1 = 1 \quad p'_2 = 2 \quad p'_3 = 1$$
$$n'_1 = 2 \quad n'_2 = 2 \quad n'_3 = 0$$

resulting in a value of 0.84 bits for E(A′). In terms of the selection criterion developed earlier, A′ now seems to give a higher information gain than A. Thus, having unknown values may apparently increase the desirability of an attribute, a result entirely opposed to common sense. The conclusion is that treating 'unknown' as a separate value is not a solution to the problem.

One strategy which has been found to work well is as follows. Let A be an attribute with values $\{A_1, A_2, \ldots A_v\}$. For some collection C of objects, let the numbers of objects with value A_i of A be p_i and n_i, and let p_u and n_u denote the numbers of objects of class P and N respectively that have unknown values of A. When the information gain of attribute A is assessed, these objects with unknown values are distributed across the values of A in proportion to the relative frequency of these values in C. Thus the gain is assessed as if the true value of p_i were given by

$$p_i + p_u \cdot \text{ratio}_i$$

where

$$\text{ratio}_i = \frac{p_i + n_i}{\sum_i (p_i + n_i)}$$

and similarly for n_i. (This expression has the property that unknown values can only decrease the information gain of an attribute.) When an attribute has been chosen by the selection criterion, objects with unknown values of that attribute are discarded before forming decision trees for the subsets $\{C_i\}$.

The other half of the story is how unknown attribute values are dealt with during classification. Suppose that an object is being classified using a decision tree that wishes to branch on attribute A, but the object's value of attribute A is unknown. The correct procedure would take the branch corresponding to the real value A_i but, since this value is unknown, the only alternative is to explore all branches without forgetting that some are more probable than others.

Conceptually, suppose that, along with the object to be classified, we have been passed a *token* with some value T. In the situation above, each branch of A_i is then explored in turn, using a token of value

$$T \cdot \text{ratio}_i$$

i.e. the given token value is distributed across all possible values in proportion to the ratios above. The value passed to a branch may be distributed further by subsequent tests on other attributes for which this object has unknown values. Instead of a single path to a leaf, there may now be many, each qualified by its token value. These token values at the leaves are summed for each class, the result of the classification being

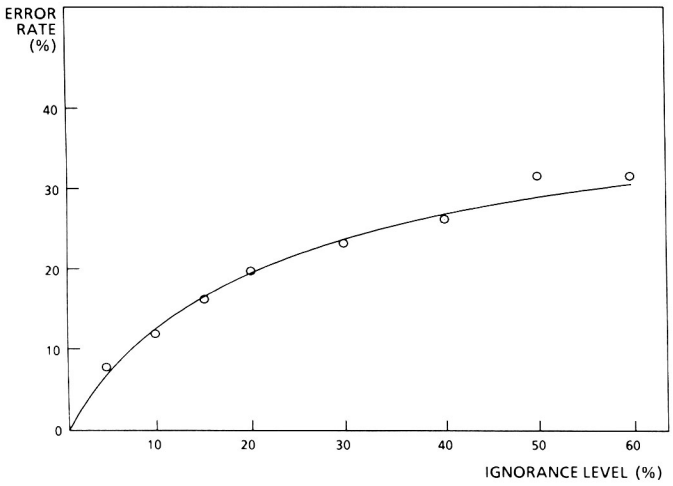

Figure 5. Error produced by unknown attribute values.

that class with the higher value. The distribution of values over the possible classes might also be used to compute a confidence level for the classification.

Straightforward though it may be, this procedure has been found to give a very graceful degradation as the incidence of unknown values increases. Figure 5 summarizes the results of an experiment on the now-familiar task with 551 objects and 39 attributes. Various 'ignorance levels' analogous to the earlier noise levels were explored, with twenty repititions at each level. For each run at an ignorance level of m percent, a copy of the 551 objects was made, replacing each value of every attribute by 'unknown' with m percent probability. A decision tree for these (incomplete) objects was formed as above, and then used to classify a new copy of each object corrupted in the same way. The figure shows that the degradation of performance with ignorance level is gradual. In practice, of course, an ignorance level even as high as 10% is unlikely – this would correspond to an average of one value in every ten of the object's description being unknown. Even so, the decision tree produced from such a patchy training set correctly classifies nearly ninety percent of objects that also have unknown values. A much lower level of degradation is observed when an object with unknown values is classified using a correct decision tree.

This treatment has assumed that no information whatsoever is available regarding an unknown attribute. Catlett (1985) has taken this approach a stage further by allowing partial knowledge of an attribute value to be stated in Shafer notation (Garvey, Lowrance and Fischler, 1981). This notation permits probabilistic assertions to be made about any subset or subsets of the possible values of an attribute that an object might have.

7. The selection criterion

Attention has recently been refocussed on the evaluation function for selecting the best attribute-based test to form the root of a decision tree. Recall that the criterion described earlier chooses the attribute that gains most information. In the course of their experiments, Bratko's group encountered a medical induction problem in which the attribute selected by the gain criterion ('age of patient', with nine value ranges) was judged by specialists to be less relevant than other attributes. This situation was also noted on other tasks, prompting Kononenko *et al* (1984) to suggest that the gain criterion tends to favor attributes with many values.

Analysis supports this finding. Let A be an attribute with values A_1, A_2, ... A_v and let A' be an attribute formed from A by splitting one of the values into two. If the values of A were sufficiently fine for the induction task at hand, we would not expect this refinement to increase the usefulness of A. Rather, it might be anticipated that excessive fineness would tend to obscure structure in the training set so that A' was in fact less useful than A. However, it can be proved that gain(A') is greater than or equal to gain(A), being equal to it only when the proportions of the classes are the same for both subdivisions of the original value. In general, then, gain(A') will exceed gain(A) with the result that the evaluation function of Section 4 will prefer A' to A. By analogy, attributes with more values will tend to be preferred to attributes with fewer.

As another way of looking at the problem, let A be an attribute with random values and suppose that the set of possible values of A is large enough to make it unlikely that two objects in the training set have the same value for A. Such an attribute would have maximum information gain, so the gain criterion would select it as the root of the decision tree. This would be a singularly poor choice since the value of A, being random, contains no information pertinent to the class of objects in the training set.

ASSISTANT (Kononenko *et al*, 1984) solves this problem by requiring that all tests have only two outcomes. If we have an attribute A as before with v values A_1, A_2, ... A_v, the decision tree no longer has a branch for each possible value. Instead, a subset S of the values is chosen and the tree has two branches, one for all values in the set and one for the remainder. The information gain is then computed as if all values in S were amalgamated into one single attribute value and all remaining values into another. Using this selection criterion (the *subset* criterion), the test chosen for the root of the decision tree uses the attribute and subset of its values that maximizes the information gain. Kononenko *et al* report that this modification led to smaller decision trees with an improved classification performance. However, the trees were judged to be less intelligible to human beings, in agreement with a similar finding of Shepherd (1983).

Limiting decision trees to a binary format harks back to CLS, in which each test was of the form 'attribute A has value A_i', with two branches corresponding to true and false. This is clearly a special case of the test implemented in ASSISTANT, which

permits a set of values, rather than a single value, to be distinguished from the others.

It is also worth noting that the method of dealing with attributes having continuous values follows the same binary approach. Let A be such an attribute and suppose that the distinct values of A that occur in C are sorted to give the sequence $V_1, V_2, \ldots, V_k$. Each pair of values V_i, V_{i+1} suggests a possible threshold

$$\frac{V_i + V_{i+1}}{2}$$

that divides the objects of C into two subsets, those with a value of A above and below the threshold respectively. The information gain of this division can then be investigated as above.

If all tests must be binary, there can be no bias in favor of attributes with large numbers of values. It could be argued, however, that ASSISTANT's remedy has undesirable side-effects that have to be taken into account. First, it can lead to decision trees that are even more unintelligible to human experts than is ordinarily the case, with unrelated attribute values being grouped together and with multiple tests on the same attribute.

More importantly, the subset criterion can require a large increase in computation. An attribute A with v values has 2^v value subsets and, when trivial and symmetric subsets are removed, there are still $2^{v-1}-1$ different ways of specifying the distinguished subset of attribute values. The information gain realized with each of these must be investigated, so a single attribute with v values has a computational requirement similar to $2^{v-1}-1$ binary attributes. This is not of particular consequence if v is small, but the approach would appear infeasible for an attribute with 20 values.

Another method of overcoming the bias is as follows. Consider again our training set containing p and n objects of class P and N respectively. As before, let attribute A have values $A_1, A_2, \ldots A_v$ and let the numbers of objects with value A_i of attribute A be p_i and n_i respectively. Enquiring about the value of attribute A itself gives rise to information, which can be expressed as

$$IV(A) = -\sum_{i=1}^{v} \frac{p_i+n_i}{p+n} \log_2 \frac{p_i+n_i}{p+n}$$

IV(A) thus measures the information content of the answer to the question, 'What is the value of attribute A?' As discussed earlier, gain(A) measures the reduction in the information requirement for a classification rule if the decision tree uses attribute A as a root. Ideally, as much as possible of the information provided by determining the value of an attribute should be useful for classification purposes or, equivalently, as little as possible should be 'wasted'. A good choice of attribute would then be one for which the ratio

gain(A) / IV(A)

is as large as possible. This ratio, however, may not always be defined – IV(A) may be zero – or it may tend to favor attributes for which IV(A) is very small. The *gain ratio* criterion selects, from among those attributes with an average-or-better gain, the attribute that maximizes the above ratio.

This can be illustrated by returning to the example based on the training set of Table 1. The information gain of the four attributes is given in Section 4 as

$$\begin{aligned} \text{gain(outlook)} &= 0.246 \text{ bits} \\ \text{gain(temperature)} &= 0.029 \text{ bits} \\ \text{gain(humidity)} &= 0.151 \text{ bits} \\ \text{gain(windy)} &= 0.048 \text{ bits} \end{aligned}$$

Of these, only outlook and humidity have above-average gain. For the outlook attribute, five objects in the training set have the value sunny, four have overcast and five have rain. The information obtained by determining the value of the outlook attribute is therefore

$$\begin{aligned} \text{IV(outlook)} &= -\frac{5}{14}\log_2\frac{5}{14} - \frac{4}{14}\log_2\frac{4}{14} - \frac{5}{14}\log_2\frac{5}{14} \\ &= 1.578 \text{ bits} \end{aligned}$$

Similarly,

$$\text{IV(humidity)} = -\frac{7}{14}\log_2\frac{7}{14} - \frac{7}{14}\log_2\frac{7}{14} = 1 \text{ bit}$$

So,

$$\begin{aligned} \text{gain ratio(outlook)} &= 0.246 / 1.578 = 0.156 \\ \text{gain ratio(humidity)} &= 0.151 / 1.000 = 0.151 \end{aligned}$$

The gain ratio criterion would therefore still select the outlook attribute for the root of the decision tree, although its superiority over the humidity attribute is now much reduced.

The various selection criteria have been compared empirically in a series of experiments (Quinlan, 1985b). When all attributes are binary, the gain ratio criterion has been found to give considerably smaller decision trees: for the 551-object task, it produces a tree of 143 nodes compared to the smallest previously-known tree of 175 nodes. When the task includes attributes with large numbers of values, the subset criterion gives smaller decision trees that also have better predictive performance, but can require much more computation. However, when these many-valued attributes are augmented by redundant attributes which contain the same information at a

lower level of detail, the gain ratio criterion gives decision trees with the greatest predictive accuracy. All in all, these experiments suggest that the gain ratio criterion does pick a good attribute for the root of the tree. Testing an attribute with many values, however, will fragment the training set C into very small subsets $\{C_i\}$ and the decision trees for these subsets may then have poor predictive accuracy. In such cases, some mechanism such as value subsets or redundant attributes is needed to prevent excessive fragmentation.

The three criteria discussed here are all information-based, but there is no reason to suspect that this is the only possible basis for such criteria. Recall that the modifications to deal with noise barred an attribute from being used in the decision tree unless it could be shown to be relevant to the class of objects in the training set. For any attribute A, the value of the statistic presented in Section 5, together with the number v of possible values of A, determines the confidence with which we can reject the null hypothesis that an object's value of A is irrelevant to its class. Hart (1985) has proposed that this same test could function directly as a selection criterion: simply pick the attribute for which this confidence level is highest. This measure takes explicit account of the number of values of an attribute and so may not exhibit bias. Hart notes, however, that the chi-square test is valid only when the expected values of p'_i and n'_i are uniformly larger than four. This condition could be violated by a set C of objects either when C is small or when few objects in C have a particular value of some attribute, and it is not clear how such sets would be handled. No empirical results with this approach are yet available.

8. Conclusion

The aim of this paper has been to demonstrate that the technology for building decision trees from examples is fairly robust. Current commercial systems are powerful tools that have achieved noteworthy successes. The groundwork has been done for advances that will permit such tools to deal even with noisy, incomplete data typical of advanced real-world applications. Work is continuing at several centers to improve the performance of the underlying algorithms.

Two examples of contemporary research give some pointers to the directions in which the field is moving. While decision trees generated by the above systems are fast to execute and can be very accurate, they leave much to be desired as representations of knowledge. Experts who are shown such trees for classification tasks in their own domain often can identify little familiar material. It is this lack of familiarity (and perhaps an underlying lack of modularity) that is the chief obstacle to the use of induction for building large expert systems. Recent work by Shapiro (1983) offers a possible solution to this problem. In his approach, called *Structured Induction*, a rule-formation task is tackled in the same style as structured programming. The task is solved in terms of a collection of notional super-attributes, after which the subtasks

of inducing classification rules to find the values of the super-attributes are approached in the same top-down fashion. In one classification problem studied, this method reduced a totally opaque, large decision tree to a hierarchy of nine small decision trees, each of which 'made sense' to an expert.

ID3 allows only two classes for any induction task, although this restriction has been removed in most later systems. Consider, however, the task of developing a rule from a given set of examples for classifying an animal as a monkey, giraffe, elephant, horse, etc. A single decision tree could be produced in which these various classes appeared as leaves. An alternative approach taken by systems such as INDUCE (Michalski, 1980) would produce a collection of classification rules, one to discriminate monkeys from non-monkeys, another to discriminate giraffes from non-giraffes, and so on. Which approach is better? In a private communication, Marcel Shoppers has set out an argument showing that the latter can be expected to give more accurate classification of objects that were not in the training set. The multi-tree approach has some associated problems – the separate decision trees may classify an animal as both a monkey and a giraffe, or fail to classify it as anything, for example – but if these can be sorted out, this approach may lead to techniques for building more reliable decision trees.

Acknowledgements

It is a pleasure to acknowledge the stimulus and suggestions provided over many years by Donald Michie, who continues to play a central role in the development of this methodology. Ivan Bratko, Igor Kononenko, Igor Mosetic and other members of Bratko's group have also been responsible for many insights and constructive criticisms. I have benefited from numerous discussions with Ryszard Michalski, Alen Shapiro, Jason Catlett and other colleagues. I am particularly grateful to Pat Langley for his careful reading of the paper in draft form and for the many improvements he recommended.

References

Buchanan, B.G., & Mitchell, T.M. (1978). Model-directed learning of production rules. In D.A. Waterman, F. Hayes-Roth (Eds.), *Pattern directed inference systems*. Academic Press.

Carbonell, J.G., Michalski, R.S., & Mitchell, T.M. (1983). An overview of machine learning, In R.S. Michalski, J.G. Carbonell and T.M. Mitchell, (Eds.), *Machine learning: An artificial intelligence approach*. Palo Alto: Tioga Publishing Company.

Catlett, J. (1985). *Induction using the shafer representation* (Technical report). Basser Department of Computer Science, University of Sydney, Australia.

Dechter, R., & Michie, D. (1985). *Structured induction of plans and programs* (Technical report). IBM Scientific Center, Los Angeles, CA.

Feigenbaum, E.A., & Simon, H.A. (1963). Performance of a reading task by an elementary perceiving and memorizing program, *Behavioral Science, 8.*

Feigenbaum, E.A. (1981). Expert systems in the 1980s. In A. Bond (Ed.), *State of the art report on machine intelligence.* Maidenhead: Pergamon-Infotech.

Garvey, T.D., Lowrance, J.D., & Fischler, M.A. (1981). An inference technique for integrating knowledge from disparate sources. *Proceedings of the Seventh International Joint Conference on Artificial Intelligence.* Vancouver, B.C., Canada: Morgan Kaufmann.

Hart, A.E. (1985). Experience in the use of an inductive system in knowledge engineering. In M.A. Bramer (Ed.), *Research and development in expert systems.* Cambridge University Press.

Hogg, R.V., & Craig, A.T. (1970). *Introduction to mathematical statistics.* London: Collier-Macmillan.

Hunt, E.B. (1962). *Concept learning: An information processing problem.* New York: Wiley.

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

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

Langley, P., Bradshaw, G.L., & Simon, H.A. (1983). Rediscovering chemistry with the BACON system. In R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* Palo Alto: Tioga Publishing Company.

Michalski, R.S (1980). Pattern recognition as rule-guided inductive inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence 2.*

Michalski, R.S., & Stepp, R.E. (1983). Learning from observation: conceptual clustering. In R.S. Michalski, J.G. Carbonell & T.M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* Palo Alto: Tioga Publishing Company.

Michie, D. (1982). Experiments on the mechanisation of game-learning 2 – Rule-based learning and the human window. *Computer Journal 25.*

Michie, D. (1983). Inductive rule generation in the context of the Fifth Generation. *Proceedings of the Second International Machine Learning Workshop.* University of Illinois at Urbana-Champaign.

Michie, D. (1985). Current developments in Artificial Intelligence and Expert Systems. In *International Handbook of Information Technology and Automated Office Systems.* Elsevier.

Nilsson, N.J. (1965). *Learning machines,* New York: McGraw-Hill.

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

Patterson, A., & Niblett, T. (1983). *ACLS user manual.* Glasgow: Intelligent Terminals Ltd.

Pearl, J. (1978a). Entropy, information and rational decisions (Technical report). Cognitive Systems Laboratory, University of California, Los Angeles.

Pearl, J. (1978b). On the connection between the complexity and credibility of inferred models. *International Journal of General Systems, 4.*

Quinlan, J.R. (1969). A task-independent experience gathering scheme for a problem solver. *Proceedings of the First International Joint Conference on Artificial Intelligence.* Washington, D.C.: Morgan Kaufmann.

Quinlan, J.R. (1979). Discovering rules by induction from large collections of examples. In D. Michie (Ed.), *Expert systems in the micro electronic age.* Edinburgh University Press.

Quinlan, J.R. (1982). Semi-autonomous acquisition of pattern-based knowledge. In J.E. Hayes, D. Michie & Y-H. Pao (Eds.), *Machine intelligence 10.* Chichester: Ellis Horwood.

Quinlan, J.R. (1983a). Learning efficient classification procedures and their application to chess endgames. In R.S. Michalski, J.G. Carbonell & T.M. Mitchell, (Eds.), *Machine learning: An artificial intelligence approach.* Palo Alto: Tioga Publishing Company.

Quinlan, J.R. (1983b). Learning from noisy data, *Proceedings of the Second International Machine Learning Workshop.* University of Illinois at Urbana-Champaign.

Quinlan, J.R. (1985a). The effect of noise on concept learning. In R.S. Michalski, J.G. Carbonell & T.M.

Mitchell (Eds.), *Machine learning*. Los Altos: Morgan Kaufmann (in press).

Quinlan, J.R. (1985b). Decision trees and multi-valued attributes. In J.E. Hayes & D. Michie (Eds.), *Machine intelligence 11*. Oxford University Press (in press).

Sammut, C.A. (1985). Concept development for expert system knowledge bases. *Australian Computer Journal 17*.

Samuel, A. (1967). Some studies in machine learning using the game of checkers II: Recent progress. *IBM J. Research and Development 11*.

Shapiro, A. (1983). *The role of structured induction in expert systems*. Ph.D. Thesis, University of Edinburgh.

Shepherd, B.A. (1983). An appraisal of a decision-tree approach to image classification. *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*. Karlsruhe, West Germany: Morgan Kaufmann.

Winston, P.H. (1975). Learning structural descriptions from examples. In P.H. Winston (Ed.), *The psychology of computer vision*. McGraw-Hill.