*Int. J. Man–Machine Studies* (1987) **27**, 349–370

# PRISM: An algorithm for inducing modular rules

JADZIA CENDROWSKA

*c/o The Faculty of Mathematics, The Open University, Walton Hall, Milton Keynes, MK7 6AA, U.K.*

*(Received 29 May 1987)*

The decision tree output of Quinlan's ID3 algorithm is one of its major weaknesses. Not only can it be incomprehensible and difficult to manipulate, but its use in expert systems frequently demands irrelevant information to be supplied. This report argues that the problem lies in the induction algorithm itself and can only be remedied by radically altering the underlying strategy. It describes a new algorithm, PRISM which, although based on ID3, uses a different induction strategy to induce rules which are modular, thus avoiding many of the problems associated with decision trees.

## 1. Introduction

Considerable effort has recently been devoted to the development of efficient knowledge acquisition techniques for expert systems, with rule induction algorithms coming under the scrutiny of a substantial number of researchers. Particular attention has been paid to Ross Quinlan's ID3 algorithm (Quinlan, 1979*a*, 1979*b*, 1983*a*) which, having performed well in the domain of chess end-games, was soon adopted for use in a number of commercial applications. However, despite this apparent success, some major limitations to the ID3 algorithm have been identified (Bundy, Silver & Plummer, 1984; Cendrowska, 1984; Hart, 1985; O'Rorke, 1982), which makes its use unsuitable for many domains. The algorithm's inability to deal with noisy input data is an area for much current research and new improved variants of ID3 are constantly being reported in the technical press (A-Razzak, Hassan & Pettipher, 1985; Hart, 1985; Lavrac *et al.* 1986; Michie, 1983; Quinlan, 1983*b*), but concern has been shown about the way in which the results of the induction process are expressed.

This report discusses the second of these two limitations. ID3 produces its output in the form of a decision tree, which can be incomprehensible (to humans), difficult to manipulate (by humans and computers) and complicates the provision of explanations (by computers for humans). In addressing this subject, it is argued that current research aimed at modifying the decision tree output of ID3 is misplaced, that the decision tree output is an inherent weakness in the algorithm itself and that this can only be remedied by radically altering the underlying induction strategy.

The first part of this report explains the problem in more detail, highlighting it by means of a simple example which is introduced in Section 2. Section 3 describes how ID3 tackles the induction task using an information theoretic approach, and the inherent weaknesses of this approach are discussed in Section 4. The subsequent sections describe how the induction strategy can be changed to avoid some of these problems and outline a proposal for a new algorithm, PRISM which, although based



0020-7373/87/100349 + 22$03.00/0 

on techniques employed by ID3, produces its output as modular rules. The report concludes with an assessment of the performance of PRISM on a large training set.

## 2. The domain

The following example, taken from the world of ophthalmic optics, will be used throughout this report to illustrate the procedures involved in rule induction.

An adult spectacle wearer enters an ophthalmic practice with a view to purchasing her first pair of contact lenses. She has had her eyes examined recently elsewhere and has brought her prescription with her. She understands that there are different types of contact lenses available, and that it is the optician's decision as to whether or not she is suitable for contact lens wear, and if so, which type she should be fitted with.

From the optician's point of view, this is a three-category† classification problem. His decision will be one of:

$\delta_1$: the patient should be fitted with hard contact lenses,
$\delta_2$: the patient should be fitted with soft contact lenses,
$\delta_3$: the patient should not be fitted with contact lenses.

In reaching his decision he must consider one or more of four† factors:

*a*: the age of the patient
1. young,
2. pre-presbyopic, or
3. presbyopic

*b*: her spectacle prescription
1. myope, or
2. hypermetrope

*c*: whether she is astigmatic
1. no, or
2. yes

*d*: her tear production rate
1. reduced, or
2. normal

Table 1 shows the optician's decision for each combination of the four factors. However, the optician does not carry such a table around with him, either on his person or in his head. Instead, through his training and experience, he has learned to exercise his professional judgement in each individual case, and will make his decision almost instinctively. If questioned as to how he arrived at a particular decision, his answer is likely to be of the form:

> This patient is not suitable for contact lens wear because her tear production rate is reduced.

or

> This patient can only be fitted with hard contact lenses because she is astigmatic. As she is young and has a normal tear production rate, hard lenses are not contraindicted.

† It should be noted that this is a highly simplified example. In real life there are many types of contact lenses and many more factors affecting the decision as to which type, if any, to fit.

TABLE 1
*Decision table for fitting contact lenses*

| | Value of attribute | | | | Decision† | | Value of attribute | | | | Decision† | | Value of attribute | | | | Decision† |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *a* | *b* | *c* | *d* | $\delta$ | | *a* | *b* | *c* | *d* | $\delta$ | | *a* | *b* | *c* | *d* | $\delta$ |
| 1 | 1 | 1 | 1 | 1 | 3 | 9 | 2 | 1 | 1 | 1 | 3 | 17 | 3 | 1 | 1 | 1 | 3 |
| 2 | 1 | 1 | 1 | 2 | 2 | 10 | 2 | 1 | 1 | 2 | 2 | 18 | 3 | 1 | 1 | 2 | 3 |
| 3 | 1 | 1 | 2 | 1 | 3 | 11 | 2 | 1 | 2 | 1 | 3 | 19 | 3 | 1 | 2 | 1 | 3 |
| 4 | 1 | 1 | 2 | 2 | 1 | 12 | 2 | 1 | 2 | 2 | 1 | 20 | 3 | 1 | 2 | 2 | 1 |
| 5 | 1 | 2 | 1 | 1 | 3 | 13 | 2 | 2 | 1 | 1 | 3 | 21 | 3 | 2 | 1 | 1 | 3 |
| 6 | 1 | 2 | 1 | 2 | 2 | 14 | 2 | 2 | 1 | 2 | 2 | 22 | 3 | 2 | 1 | 2 | 2 |
| 7 | 1 | 2 | 2 | 1 | 3 | 15 | 2 | 2 | 2 | 1 | 3 | 23 | 3 | 2 | 2 | 1 | 3 |
| 8 | 1 | 2 | 2 | 2 | 1 | 16 | 2 | 2 | 2 | 2 | 3 | 24 | 3 | 2 | 2 | 2 | 3 |

† The reader is asked not to be tempted to use this decision table to determine whether or not (s)he is suitable for contact lenses as there are many factors, not mentioned here, which may radically influence the decision.

Each explanation is a justification of a decision in terms of the values of relevant attributes, and is based on one or more 'rules of thumb':

**if** tear production rate is reduced
**then** do not fit contact lenses,

**if** the patient is astigmatic, **and**
the patient is young, **and**
the tear production rate is normal
**then** fit hard contact lenses.

Although the optician is able to easily justify each individual decision, he would find it quite difficult to formalize his knowledge as a complete set of rules. ID3 seeks to establish this underlying set of rules, in the form of a decision tree, from examples of the optician's decisions. The algorithm is described in detail in Section 3. Table 1 is used as the training set of instances; $\delta_1$, $\delta_2$ and $\delta_3$ are the decisions or classifications; *a*, *b*, *c* and *d* are the attributes. Attribute *a* has three possible values (1, 2 and 3) and attributes *b*, *c* and *d* each have two possible values (1 and 2). Each instance is a description of a classification in terms of values of the four attributes. The following assumptions have been made about the training set:

- the classifications are mutually exclusive
- there is no noise, i.e. each instance is complete and correct
- each instance can be classified uniquely
- no instance is duplicated
- the values of the attributes are discrete
- the training set is complete, i.e. all possible combinations of attribute–value pairs are represented

## 3. An information theoretic approach I

### 3.1. ENTROPY

The training set can be thought of as a discrete information system, i.e. it contains a number of discrete messages (values of attributes) which impart some information

about an event (classification). The entropy of a set of events has been defined as a measure of the 'freedom of choice' involved in the selection of the event, or the 'uncertainty' associated with this selection (Edwards, 1964, Goldman, 1968, Shannon & Weaver, 1949). Given a training set, $S$, if the above assumptions hold, then each instance is classified correctly and uniquely, i.e. there is no uncertainty about the classification. The entropy of $S$ is 0. The entropy of a decision tree or rule set, which fully describes $S$ is also 0, but in most cases the decision tree is a generalization of $S$, which implies that some information offered by the training set is redundant. ID3 tries to reduce this redundant information as much as possible (and thus find the least complex decision tree which fully describes the training set) by partitioning $S$ into the smallest possible number of subsets, each of which can be described by a set of features (attribute–value pairs) whose entropy is 0.

If all that is known about the classifications is their probabilities of occurrence, $p(\delta_i; i = 1, 2, 3)$, then the entropy of the set of classifications,

$$H = -\sum_i p(\delta_i) \log_2 p(\delta_i) \text{ bits.} \tag{1}$$

For the contact lens classification problem,

$$H = -p(\delta_1) \log_2 p(\delta_1) - p(\delta_2) \log_2 p(\delta_2) - p(\delta_3) \log_2 p(\delta_3) \text{ bits.}$$

The probabilities of occurrence of each of the classifications are

$$p(\delta_1) = 4/24,$$
$$p(\delta_2) = 5/24,$$
$$p(\delta_3) = 15/24.$$

Thus,

$$\begin{aligned} H &= -\frac{4}{24}\log_2\left(\frac{4}{24}\right) - \frac{5}{24}\log_2\left(\frac{5}{24}\right) - \frac{15}{24}\log_2\left(\frac{15}{24}\right) \\ &= 0{\cdot}4308 + 0{\cdot}4715 + 0{\cdot}4238 \\ &= 1{\cdot}3261 \text{ bits.} \end{aligned} \tag{2}$$

The induction algorithm partitions the training set into subsets in such a way as to reduce this entropy by the maximum amount, and continues doing so recursively until the entropy is 0.

### 3.2. REDUCING ENTROPY

If the training set, $S$, is divided according to the values of some attribute, $\alpha$, then unless the classification, $\delta$, is completely independent of $\alpha$, the values will contain some information about $\delta$. The total entropy of the subsets is known as the conditional entropy of $S$ with known $\alpha$, $H(S \mid \alpha)$. Let $p(\alpha_x)$ be the probability that attribute $\alpha$ has value $x$, and let $p(\delta_n \cap \alpha_x)$ be the probability that the classification is $\delta_n$ and the value of $\alpha$ is $x$. Then

$$H(S \mid \alpha) = H(S \cap \alpha) - H(\alpha), \tag{3}$$

where

$$H(S \cap \alpha) = -\sum_x \sum_n p(\delta_n \cap \alpha_x) \log_2 p(\delta_n \cap \alpha_x) \tag{4}$$

and

$$H(\alpha) = -\sum_x p(\alpha_x) \log_2 p(\alpha_x). \tag{5}$$

By performing this calculation for each attribute, it is possible to minimize the entropy of $S$ by dividing it into subsets according to the values of that attribute for which $H(S \mid \alpha)$ is minimum.

The calculation can be simplified by using a frequency table, for example for attribute $a$:

| No. of instances referencing | $a_1$ | $a_2$ | $a_3$ | Total |
|---|---|---|---|---|
| $\delta_1$ | 2 | 1 | 1 | 4 |
| $\delta_2$ | 2 | 2 | 1 | 5 |
| $\delta_3$ | 4 | 5 | 6 | 15 |
| Total | 8 | 8 | 8 | 24 |

$$\begin{aligned} H(S \mid a) &= H(S \cap a) - H(a) \\ &= -\sum_x \sum_n p(\delta_n \cap a_x) \log_2 p(\delta_n \cap a_x) + \sum_x p(a_x) \log_2 p(a_x) \\ &= -3 \times \frac{2}{24} \log_2 \left(\frac{2}{24}\right) - 3 \times \frac{1}{24} \log_2 \left(\frac{1}{24}\right) - \frac{4}{24} \log_2 \left(\frac{4}{24}\right) \\ &\quad - \frac{5}{24} \log_2 \left(\frac{5}{24}\right) - \frac{6}{24} \log_2 \left(\frac{6}{24}\right) + 3 \times \frac{8}{24} \log_2 \left(\frac{8}{24}\right) \\ &= \frac{1}{24} (3 \times 8 \log_2 8 - 3 \times 2 \log_2 2 - 2 \times \log_2 1 - 4 \log_2 4 \\ &\quad - 5 \log_2 5 - 6 \log_2 6) \\ &= 1{\cdot}2867 \text{ bits.} \end{aligned} \tag{6}$$

Similarly,

$$H(S \mid b) = 1{\cdot}2867 \text{ bits,} \tag{7}$$

$$H(S \mid c) = 0{\cdot}9491 \text{ bits,} \tag{8}$$

$$H(S \mid d) = 0{\cdot}7773 \text{ bits.} \tag{9}$$

Therefore, the entropy of $S$ can be reduced by the greatest amount by dividing $S$ according to the values of attribute $d$. Two subsets are formed, each of which is then further subdivided in the same way until the entropy of each subset is 0, i.e. all instances in the subset belong to the same classification. The final decision tree is

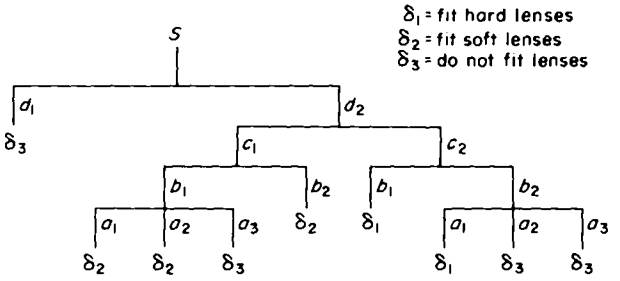


FIG. 1. Decision tree produced by ID3.

shown in Fig. 1. For convenience, this can be written as a set of individual rules:

1. $d_1 \rightarrow \delta_3$
2. $d_2 \wedge c_1 \wedge b_1 \wedge a_1 \rightarrow \delta_2$
3. $d_2 \wedge c_1 \wedge b_1 \wedge a_2 \rightarrow \delta_2$
4. $d_2 \wedge c_1 \wedge b_1 \wedge a_3 \rightarrow \delta_3$
5. $d_2 \wedge c_1 \wedge b_2 \rightarrow \delta_2$
6. $d_2 \wedge c_2 \wedge b_1 \rightarrow \delta_1$
7. $d_2 \wedge c_2 \wedge b_2 \wedge a_1 \rightarrow \delta_1$
8. $d_2 \wedge c_2 \wedge b_2 \wedge a_2 \rightarrow \delta_3$
9. $d_2 \wedge c_2 \wedge b_2 \wedge a_3 \rightarrow \delta_3$

## 4. Rule representation

One of the principal features of rule-based expert systems is that the modularity of the rules typically enables a knowledge base to be easily updated or modified. It also provides a means for explanation. There is a requirement, therefore, that rules should be both modular and comprehensible, whether they are elicited from experts or automatically induced from examples.

Although ID3 has been proved to be computationally efficient (Carbonell, Michalski & Mitchell, 1983; Michie, 1983; O'Rorke, 1982), it produces its output in the form of a decision tree (e.g. Fig. 1). This decision tree representation of rules has a number of disadvantages. Firstly, decision trees are extremely difficult to manipulate—to extract information about any single classification it is necessary to examine the complete tree. This problem is only partially resolved by trivially converting the tree into a set of individual rules, as the amount of information contained in some of these will often be more than an easily be assimilated. More importantly, there are rules that cannot easily be represented by trees.

Consider, for example, the following rule set:

$$\text{Rule 1: } a_1 \wedge b_1 \rightarrow \delta_1,$$

$$\text{Rule 2: } c_1 \wedge d_1 \rightarrow \delta_1.$$

Suppose that Rules 1 and 2 cover all instances of class $\delta_1$ and all other instances are of class $\delta_2$. These two rules cannot be represented by a single decision tree as the

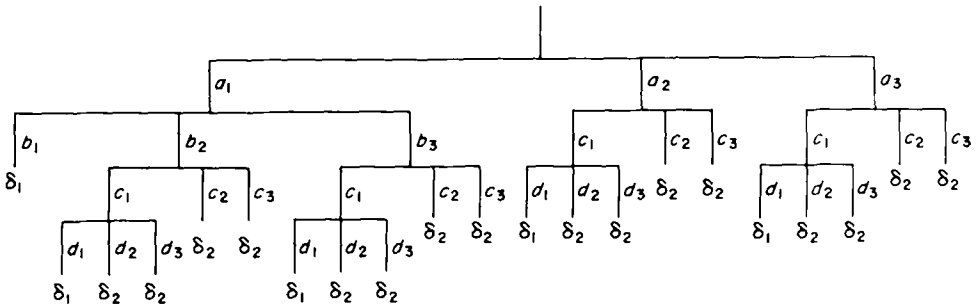

FIG. 2. Decision tree representation of Rules 1 and 2 (Section 4).

root node of the tree must split on a single attribute, and there is no attribute which is common to both rules. The simplest decision tree representation of the set of instances covered by these rules would necessarily add an extra term to one of the rules, which in turn would require at least one extra rule to cover instances excluded by the addition of that extra term. The complexity of the tree would depend on the number of possible values of the attributes selected for partitioning. For example, let the four attributes, $a$, $b$, $c$ and $d$ each have three possible values, 1, 2 and 3, and let attribute $a$ be selected for partitioning at the root node. Then the simplest decision tree representation of Rules 1 and 2 above is shown in Fig. 2. The paths relating to class $\delta_1$ can be listed as follows:

1. $a_1 \wedge b_1 \rightarrow \delta_1$,
2. $a_1 \wedge b_2 \wedge c_1 \wedge d_1 \rightarrow \delta_1$,
3. $a_1 \wedge b_3 \wedge c_1 \wedge d_1 \rightarrow \delta_1$,
4. $a_2 \wedge c_1 \wedge d_1 \rightarrow \delta_1$,
5. $a_3 \wedge c_1 \wedge d_1 \rightarrow \delta_1$.

Clearly, the consequence of forcing a simple rule set into a decision tree representation is that the individual rules, when extracted from the tree, are often too specific (i.e. they reference attributes which are irrelevant). This makes them highly unsuitable for use in many domains, as is illustrated by the following example.

Suppose the decision tree in Fig. 1 was used as the knowledge base for an expert system advising on contact lens suitability, and suppose the patient requiring contact lenses was a presbyope with high hypermetropia and astigmatism (attributes $a_3$ & $b_2$ & $c_2$). The optician would know immediately from the age of the patient and her prescription that she was not a suitable candidate for contact lens wear (a decision taking about 30 seconds to make and costing the patient nothing). The expert system, however, would be unable to make a decision without the result of a tear production rate test (attribute $d$). This test is normally carried out as part of a contact lens consultation requiring a lot of time and payment of a fee. Having spent all this time and money, it would be quite understandable if the patient became upset or angry on finding out that the consultation had been, after all, unnecessary. The consequences could be even more serious if the expert system was a medical one and attribute $d$ involved surgery.

Clearly, a decision tree in its unmodified form is most unsuitable for some domains, not only because it an be incomprehensible, but because in many cases its use would demand irrelevant information to be supplied, information that could be costly to obtain. Attempts have been made at modifying the algorithm to avoid this problem by assigning a 'cost' to each attribute. Attempts have also been made at converting decision trees into simple rule sets by identifying and removing redundant nodes, or by incorporating extra information which enables the user to focus on only relevant parts of the tree, but the problem is not an easy one to solve, particularly for very large and complex decision trees.

Although simplification of the trees is possible by identifying common branches or parts of branches, the combinatorial explosion in the number of comparisons that have to be made as the complexity increases makes this method only feasible for small trees. Also, parts of a branch may be matched in different ways, and the question then arises as to which is the better generalization to make. This would involve either asking the expert, or using another rule induction program to induce new rules from the old ones.

## 5. An information theoretic approach II

### 5.1. ENTROPY VS. INFORMATION GAIN

The main cause of the problem described in the preceding section is either that an attribute is highly relevant to only one classification and irrelevant to the others, or that only one value of the attribute is relevant. For example, the attribute $d$ in the contact lens problem is highly relevant to the classification $\delta_3$, *if its value is* 1, and because of this, it is selected for partitioning the training set, for which all its values are used.

Figure 3 shows the decision tree after $S$ has been partitioned according to the

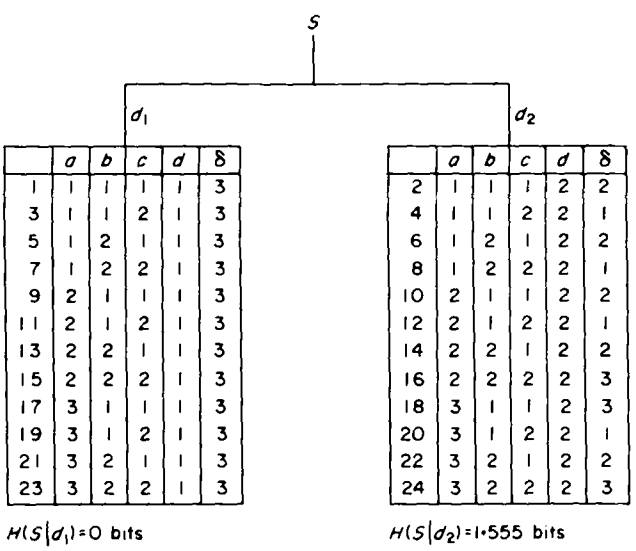


FIG. 3. $S$ partitioned according to $d$.

values of attribute $d$. It can be seen that although the entropy of the branch $d_1$ has been reduced to 0, the entropy of the branch $d_2$ has actually increased to 1·555 bits. Attribute $d$ was chosen because ID3 minimizes the *average entropy* of the training set, or alternatively, it maximizes the *average* amount of information contributed by an attribute to the determination of *any* classification.

In order to eliminate the use of irrelevant values of attributes and attributes which are irrelevant to a classification, the algorithm needs to maximize the *actual* amount of information contributed by knowing the value of the attribute to the determination of a *specific* classification.

### 5.2. INFORMATION CONTENT

As stated at the beginning of Section 3, the values of attributes can be thought of as discrete messages in a discrete information system. Now, the amount of information about an event in a message $i$,

$$I(i) = \log_2 \left( \frac{\text{probability of event after the message is received}}{\text{probability of event before the message is received}} \right) \text{bits.}$$

The training set, $S$, contains 4 instances belonging to class $\delta_1$, 5 belonging to class $\delta_2$ and 15 to class $\delta_3$. Therefore, the probability of an instance belonging to class $\delta_1$, $p(\delta_1)$ is 4/24 and thus if the message $i$ was $\delta_1$ (i.e. the class is $\delta_1$) then the amount of information received in this message,

$$I(\delta_1) = \log_2 \left( \frac{1}{p(\delta_1)} \right) = -\log_2 \left( \frac{4}{24} \right) = 2{\cdot}585 \text{ bits.} \tag{10}$$

Similarly, the amount of information received in the message $\delta_2$,

$$I(\delta_2) = \log_2 \left( \frac{1}{p(\delta_2)} \right) = -\log_2 \left( \frac{5}{24} \right) = 2{\cdot}263 \text{ bits.} \tag{11}$$

and in the message $\delta_3$,

$$I(\delta_3) = \log_2 \left( \frac{1}{p(\delta_3)} \right) = -\log_2 \left( \frac{15}{24} \right) = 0{\cdot}678 \text{ bits.} \tag{12}$$

Thus the lower the probability of occurrence of an event, the more information we receive if we are told that the event has occurred.

Now, if the message received was that attribute $d$ has value 1, then the amount of information received in this message about $\delta_3$,

$$I(\delta_3 \mid d_1) = \log_2 \left( \frac{p(\delta_3 \mid d_1)}{p(\delta_3)} \right) \text{bits.} \tag{13}$$

where $p(\delta_3 \mid d_1)$ is the probability of $\delta_3$ given that the value of $d$ is 1.

For $S$, $p(\delta_3 \mid d_1) = 1$, therefore

$$I(\delta_3 \mid d_1) = \log_2 \left( \frac{1}{p(\delta_3)} \right) = 0{\cdot}678 \text{ bits.} \tag{14}$$

Thus knowing that attribute $d$ has value 1 contributes 0·678 bits of information to the belief that an instance belongs to class $\delta_3$.

If, on the other hand, the message was that attribute $d$ has value 2, then the amount of information received about $\delta_3$,

$$I(\delta_3 \mid d_2) = \log_2\left(\frac{p(\delta_3 \mid d_2)}{p(\delta_3)}\right) = \log_2\left(\frac{3/12}{15/24}\right) = -1{\cdot}322 \text{ bits.} \tag{15}$$

The minus sign indicates that knowing that the value of $d$ is 2 makes it less certain that an instance belongs to $\delta_3$ than if the value of $d$ was unknown. $d_2$ is therefore not a good choice for describing $\delta_3$.

If an attribute-value pair, $\alpha_x$, and a classification, $\delta_n$, are completely independent, then $p(\delta_n \mid \alpha_x) = p(\delta_n)$ and $I(\delta_n \mid \alpha_x) = \log_2 1 = 0$, i.e. the fact $\alpha_x$ contributes no information to the belief that the class is $\delta_n$.

### 5.3. MAXIMIZING INFORMATION GAIN

The task of an induction algorithm must be to find the attribute–value pair, $\alpha_x$, which contributes the most information about a specified classification, $\delta_n$, i.e. for which $I(\delta_n \mid \alpha_x)$ is maximum. Now,

$$I(\delta_n \mid \alpha_x) = \log_2\left(\frac{p(\delta_n \mid \alpha_x)}{p(\delta_n)}\right) \text{ bits.} \tag{16}$$

but $p(\delta_n)$ is the same for all $\alpha_x$, and thus it is only necessary to find the $\alpha_x$ for which $p(\delta_n \mid \alpha_x)$ is maximum.

The values of $p(\delta_n \mid \alpha_x)$ for all $\alpha_x$ and $n = 1$ are listed in Table 2a. There are two candidates for 'best' $\alpha_x$. These are $c_2$ and $d_2$. For $c_2$, chosen arbitrarily, the information gain,

$$I(\delta_1 \mid c_2) = \log_2\left(\frac{p(\delta_1 \mid c_2)}{p(\delta_1)}\right) = \log_2\left(\frac{4/12}{4/24}\right) = 1 \text{ bit.} \tag{17}$$

Had $d_2$ been chosen, the information gain would also have been 1 bit. Repeating the process now on a subset of $S$ which contains only those instances which have value 2 for attribute $c$, it can be seen from Table 2b that $p(\delta_1 \mid \alpha_x)$ has the highest value for $d_2$. The information gain (for this subset),

$$I(\delta_1 \mid d_2) = \log_2\left(\frac{p(\delta_1 \mid d_2)}{p(\delta_1)}\right) = \log_2\left(\frac{4/6}{4/12}\right) = 1 \text{ bit.} \tag{18}$$

If the process is now repeated on the subset which contains only those instances which have value 2 for attribute $c$ and value 2 for attribute $d$ (Table 2c), there is again a choice for 'best' $\alpha_x$. Suppose the second of these, $b_1$, is selected.† Then

$$I(\delta_1 \mid b_1) = \log_2\left(\frac{p(\delta_1 \mid b_1)}{p(\delta_1)}\right) = \log_2\left(\frac{1}{4/6}\right) = 0{\cdot}585 \text{ bits.} \tag{19}$$

From equation 10, the information provided by the message $\delta_1$ before any attributes are known = 2·585 bits.

The information provided by $c_2 = 1$ bit.

† The reason for this choice is explained in Section 7.2.1.

TABLE 2a
*Selecting the first term*

| $\alpha_x$ | $p(\delta_1 \mid \alpha_x)$ |
|---|---|
| $a_1$ | 2/8 = 0·25 |
| $a_2$ | 1/8 = 0·125 |
| $a_3$ | 1/8 = 0·125 |
| $b_1$ | 3/12 = 0·25 |
| $b_2$ | 1/12 = 0·083 |
| $c_1$ | 0 = 0 |
| $c_2$ | 4/12 = 0·333 |
| $d_1$ | 0 = 0 |
| $d_2$ | 4/12 = 0·333 |

TABLE 2b
*Selecting the second term*

| $\alpha_x$ | $p(\delta_1 \mid \alpha_x)$ |
|---|---|
| $a_1$ | 2/4 = 0·5 |
| $a_2$ | 1/4 = 0·25 |
| $a_3$ | 1/4 = 0·25 |
| $b_1$ | 3/6 = 0·5 |
| $b_2$ | 1/6 = 0·167 |
| $d_1$ | 0 = 0 |
| $d_2$ | 4/6 = 0·667 |

TABLE 2c
*Selecting the third term*

| $\alpha_x$ | $p(\delta_1 \mid \alpha_x)$ |
|---|---|
| $a_1$ | 2/2 = 1 |
| $a_2$ | 1/2 = 0·5 |
| $a_3$ | 1/2 = 0·5 |
| $b_1$ | 3/3 = 1 |
| $b_2$ | 1/3 = 0·333 |

The information provided by $d_2$ when $c_2$ is known = 1 bit.

The information provided by $b_1$ when $d_2$ and $c_2$ are known = 0·585 bits.

Therefore, the information provided by $c_2 \wedge d_2 \wedge b_1 = 1 + 1 + 0{\cdot}585 = 2{\cdot}585$ bits.

i.e. the message $c_2 \wedge d_2 \wedge b_1$ provides the same amount of information as the message $\delta_1$.

Specialization of (i.e. adding more attribute-value pairs to) $c_2 \wedge d_2 \wedge b_1$ does not increase the information gain. All other attributes are irrelevant in this description as all instances containing $c_2 \,\&\, d_2 \,\&\, b_1$ belong to class $\delta_1$ ($p(\delta_1 \mid c_2 \wedge d_2 \wedge b_1) = 1$). The induced rule is therefore

$$c_2 \wedge d_2 \wedge b_1 \rightarrow \delta_1$$

and is known to be correct for $S$.

### 5.4. TRIMMING THE TREE

The decision tree at this stage of the induction process is shown in Fig. 4. The algorithm has concentrated on building the shortest branch possible for the class $\delta_1$. The remaining branches are not yet labelled, and the next step in the induction process is to identify the best rule for the set of instances which are not examples of the first rule. This is done by removing from $S$ all instances containing $c_2 \,\&\, d_2 \,\&\, b_1$

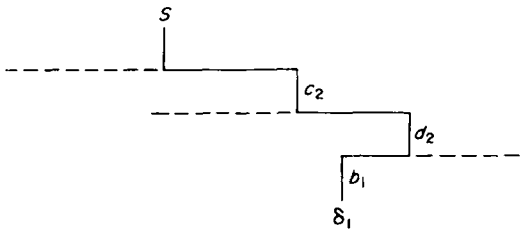


FIG. 4. 'Decision tree' after induction of the first rule.

and applying the algorithm to the remaining instances. If this is repeated until there are no instances of class $\delta_1$ left in $S$, the result is not a decision tree but a collection of branches. The whole process can then be repeated for each classification in turn, starting with the complete training set, $S$, each time.

The final output is an unordered collection of modular rules, each rule being as general as possible (but see Section 7.2), thus ensuring that there are no redundant terms. The rule set for the optician's contact lens classification problem is as follows:

1. $c_2 \wedge d_2 \wedge b_1 \rightarrow \delta_1$,
2. $a_1 \wedge c_2 \wedge d_2 \rightarrow \delta_1$,
3. $c_1 \wedge d_2 \wedge b_2 \rightarrow \delta_2$,
4. $c_1 \wedge d_2 \wedge a_1 \rightarrow \delta_2$,
5. $c_1 \wedge d_2 \wedge a_2 \rightarrow \delta_2$,
6. $d_1 \rightarrow \delta_3$,
7. $a_3 \wedge b_1 \wedge c_1 \rightarrow \delta_3$,
8. $b_2 \wedge c_2 \wedge a_2 \rightarrow \delta_3$,
9. $b_2 \wedge c_2 \wedge a_3 \rightarrow \delta_3$.

Although the number of rules in this set is the same as the number of leaf nodes in the decision tree (Fig. 1), six of the rules have had redundant terms removed. The presbyopic patent with high hypermetropia and astigmatism no longer needs to undergo an examination to be told that she is not suitable for contact lens wear (Rule 9).

## 6. The 'correctness' of rules and predictability

Given that the assumptions listed at the end of Section 2 hold, the above algorithm produces a complete set of correct rules.† This section is devoted to explaining first the meaning, and then the importance of this statement.

### 6.1. A COMPLETE SET...

A set of rules is complete if for every possible example of a classification there is at least one rule which explains it. It is assumed that all examples can be adequately

† This statement applies to most training sets. For the remainder, the algorithm must first be modified as explained in Section 7.2.

described in terms of the attributes used for the training set. Such a set of rules can be used for predicting the classification of any instance, which is a basic requirement for any rule induction program. A set of rules *must* be complete if it is induced from a complete training set. Otherwise, a rule set can be either complete or incomplete.

### 6.2. ... OF CORRECT RULES

On the other hand, a rule which is not incorrect is not necessarily correct. There are different levels of 'correctness'. An incorrect rule is one which misclassifies instances. For example, the rule Rule 1: $a_1 \wedge b_1 \rightarrow \delta_1$ is incorrect if it is too general, because there will be some instances which have value 1 for attribute $a$ and value 1 for attribute $b$, but which are of a class of other than $\delta_1$. These instances will be misclassified as $\delta_1$ by Rule 1. It is possible for a rule to be both too general and too specific; for example, if Rule 1 should have been $a_1 \wedge c_1 \rightarrow \delta_1$, then it is too general with respect to attribute $c$ but too specific with respect to attribute $b$. However, this does not alter the fact that the rule is incorrect because it still misclassifies some instances. An incorrect rule is, therefore, one which does not reference all the relevant attributes.

A rule which is not too general is correct in the sense that it will not misclassify any instances. If it is too specific, however, it will fail to classify some instances which it should classify, although there may be other rules in the set which will cover these instances. A rule which is too specific is incorrect in the sense that it will not fire unless the value of an irrelevant attribute has been determined. The undesirability of this was discussed in Section 4.

A 'correct' rule, therefore, is one which references all the relevant attributes and no irrelevant ones. A complete set of correct rules classifies all possible instances correctly.

### 6.3. PREDICTABILITY

The algorithm described in Section 5 induces a complete set of correct rules, on the condition that the assumptions listed in Section 2 hold. However, these assumptions are extremely restrictive and unlikely to be applicable to 'real-life' classification problems. In particular, the last assumption—that the training set be complete—is most unrealistic. Relaxing any of the restrictions, even slightly, introduces into the set of induced rules the possibility of errors or uncertainty, thus reducing their predictability value. If the rule set cannot be guaranteed to be complete and correct (in the strict sense) when the training set does meet the assumptions then any errors or uncertainty introduced by relaxing the restrictions will be greatly increased. The importance of knowing that the rule set is complete and correct for a complete and noiseless training set cannot be over-emphasized.

## 7. Prism

The theory outlined in Section 5 has been embodied in a new rule induction program, PRISM. PRISM takes as input a training set entered as a file of ordered sets of attribute values, each set being terminated by a classification. Information about the attributes and classifications (e.g. name, number of possible values, list of

possible values, etc.) is input from a separate file at the start of the program, and the results are output as individual rules for each of the classifications listed in terms of the described attributes.

### 7.1. THE BASIC ALGORITHM

The basic induction algorithm is essentially as described above, namely:

If the training set contains instances of more than one classification, then for each classification, $\delta_n$, in turn:

Step 1: calculate the probability of occurrence, $p(\delta_n \mid \alpha_x)$, of the classification $\delta_n$ for each attribute–value pair $\alpha_x$,

Step 2: select the $\alpha_x$ for which $p(\delta_n \mid \alpha_x)$ is a maximum and create a subset of the training set comprising all the instances which contain the selected $\alpha_x$,

Step 3: repeat Steps 1 and 2 for this subset until it contains only instances of class $\delta_n$. The induced rule is a conjunction of all the attribute–value pairs used in creating the homogeneous subset.

Step 4: remove all instances covered by this rule from the training set,

Step 5: repeat Steps 1–4 until all instances of class $\delta_n$ have been removed.

When the rules for one classification have been induced, the training set is restored to its initial state and the algorithm is applied again to induce a set of rules covering the next classification. As the classifications are considered separately, their order of presentation is immaterial. If all instances are of the same classification then that classification is returned as the rule, and the algorithm terminates.

Although the basic induction algorithm used by PRISM is based on techniques employed by ID3, it is quite unlike ID3 in many respects. The major difference is that PRISM concentrates on finding only relevant values of attributes, while ID3 is concerned with finding the attribute which is most relevant overall, even though some values of that attribute may be irrelevant. All other differences between the two algorithms stem from this. ID3 divides a training set into homogeneous subsets without reference to the class of this subset, whereas PRISM must identify subsets of a specific class. This has the disadvantage of slightly increased computational effort, but the advantage of an output in the form of modular rules rather than a decision tree.

### 7.2. THE USE OF HEURISTICS

The two algorithms are similar in that they both employ an information theoretic approach to discovering disjunctive rules by grouping together sets of instances with similar features. Consequently, they both encounter similar difficulties in certain circumstances. In particular, there is the problem of which attribute or attribute–value pair to choose when the results of the respective calculations indicate that there are two or more which are equal. In ID3, however, the choice is immaterial because the objective is to reduce entropy at the maximal rate and this is achieved equally well whichever attribute is chosen. On the other hand, if the wrong choice is made in PRISM, then the result is that an irrelevant attribute–value pair may be

chosen. Fortunately, this most unwelcome feature can be avoided by incorporating some heuristics in the basic algorithm.

*7.2.1. Opting for generality I*

If there are two or more rules describing a classification, PRISM tries to induce the most general rule first. The rationale behind this is that the more general a rule is then the less likely it is to reference an irrelevant attribute. Thus where there is a choice of attribute–value pairs, PRISM selects that attribute–value pair which has the highest frequency of occurrence in the set of instances being considered. Referring back to Table 2c in Section 5 (selection of a third term for the first rule for class $\delta_1$), it can be seen that the attribute–value pairs $a_1$ and $b_1$ both offer an equal information gain. PRISM selects $b_1$ because the resulting rule covers three instances, whereas the rule resulting from the selection of $a_1$ would only cover two instances. Thus the rule $c_2 \wedge d_2 \wedge b_1 \rightarrow \delta_1$ is more general than $c_2 \wedge d_2 \wedge a_1 \rightarrow \delta_1$. In this particular case, both rules are in fact equally correct, and so the order in which they are induced does not really matter, but opting for generality in this way has the advantage of reducing computational effort when there is a significant difference in the number of instances covered by each of the rules. Its true value, however, is realized when the training set is an incomplete one and there is a possibility that one potential rule is a specialization of another. In this situation PRISM *must* select the more general.

*7.2.2. Opting for generality II*

When both the information gain offered by two or more attribute–value pairs is the same and the numbers of instances referencing them is the same, PRISM selects the first. This is the only time that the order of input of the attributes affects the induction process, but in these cases it is still possible for an irrelevant attribute–value pair to be selected. To illustrate how PRISM copes with this situation, suppose there are four attributes, $a$, $b$, $c$ and $d$, each having three possible values, 1, 2 and 3, and the rules to be induced for class $\delta_1$ are:

$$\text{Rule 1: } c_1 \wedge d_1 \rightarrow \delta_1,$$
$$\text{Rule 2: } c_2 \wedge d_2 \rightarrow \delta_1,$$
$$\text{Rule 3: } c_3 \wedge d_3 \rightarrow \delta_1.$$

Thus, attributes $a$ and $b$ are irrelevant to $\delta_1$, whereas all values of attributes $c$ and $d$ are equally relevant. If the training set is complete, then $p(\delta_1 \mid \alpha_x)$ is the same for all $\alpha_x$ and PRISM selects $a_1$. The subset containing only instances which have value 1 for attribute $a$ also presents the same problem—$p(\delta_1 \mid \alpha_x)$ is equal for all $\alpha_x$, so $b_1$ is selected, and so on. The result is the following set of rules:

$$\text{Rule 1: } a_1 \wedge b_1 \wedge c_1 \wedge d_1 \rightarrow \delta_1,$$
$$\text{Rule 2: } a_2 \wedge b_1 \wedge c_1 \wedge d_1 \rightarrow \delta_1,$$
$$\text{Rule 3: } a_3 \wedge b_1 \wedge c_1 \wedge d_1 \rightarrow \delta_1,$$
$$\text{Rule 4: } b_2 \wedge a_1 \wedge c_1 \wedge d_1 \rightarrow \delta_1,$$
$$\text{Rule 5: } b_3 \wedge a_1 \wedge c_1 \wedge d_1 \rightarrow \delta_1.$$

At this stage $p(\delta_1 \mid \alpha_x)$ is greater for $c_2, c_3, d_2$ and $d_3$ than for any other attribute–value pair, so the next two rules are induced correctly:

$$\text{Rule 6: } c_2 \wedge d_2 \rightarrow \delta_1,$$

$$\text{Rule 7: } c_3 \wedge d_3 \rightarrow \delta_1.$$

The remaining instances all have value 1 for attribute $c$ and value 1 for attribute $d$, so the final rule is

$$\text{Rule 8: } c_1 \wedge d_1 \rightarrow \delta_1.$$

Rules 1–5 are all specializations of Rule 8. To avoid this happening, PRISM first induces all rules for a classification and then selects the most general of these on the basis of (i) the rule which covers the maximum number of instances, and (ii) the rule which references the fewest attributes. The instances covered by this rule are removed from the training set, and PRISM goes on to induce the remaining rules in the same way. For the above example, the result is that Rules 6 and 7 are induced first, and then Rule 8. These three rules account for all instances of class $\delta_1$, so Rules 1–5 are discarded.

Although this iterative procedure is quite costly in terms of computational effort, it ensures (at least for a complete training set) that the induced rules are maximally general.

## 8. Induction from incomplete training sets

When PRISM is applied to a complete training set, the resulting set of rules can confidently be expected to be complete and correct. When the training set is incomplete, this confidence is reduced. The smaller the relative number of instances in the training set, the more likely it is that the rule set will contain errors. Errors in the induction process arise for a number of reasons and can be best explained using an (artificial) example. For this purpose, suppose there are four attributes, $a, b, c$ and $d$. Attribute $a$ has five possible values $(1, 2, 3, 4, 5)$, attributes $b$ and $c$ each have four possible values $(1, 2, 3, 4)$ and attribute $d$ has three possible values $(1, 2, 3)$. Thus a complete training set would consist of $5 \times 4 \times 4 \times 3 = 240$ instances. Suppose that the rule set governing class $\delta_1$ is

$$\text{Rule 1: } a_4 \wedge d_2 \rightarrow \delta_1,$$

$$\text{Rule 2: } c_1 \wedge d_1 \rightarrow \delta_1,$$

$$\text{Rule 3: } a_2 \wedge c_4 \wedge d_2 \rightarrow \delta_1,$$

$$\text{Rule 4: } a_5 \wedge c_4 \wedge d_2 \rightarrow \delta_1,$$

and that the 40 instances listed in Table 3 are the only ones available to the induction program.

The set of rules induced by PRISM for the class $\delta_1$ is

$$\text{Rule A: } a_4 \wedge d_2 \rightarrow \delta_1,$$

$$\text{Rule B: } a_3 \wedge c_1 \wedge d_1 \rightarrow \delta_1,$$

$$\text{Rule C: } a_2 \wedge c_4 \rightarrow \delta_1,$$

$$\text{Rule D: } b_1 \wedge d_1 \wedge c_1 \rightarrow \delta_1.$$

TABLE 3
*Example of incomplete training set*

| *a* | *b* | *c* | *d* | δ | *a* | *b* | *c* | *d* | δ | *a* | *b* | *c* | *d* | δ | *a* | *b* | *c* | *d* | δ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 3 | 3 | 2 | 2 | 1 | 2 | 2 | 2 | 3 | 2 | 1 | 1 | 1 | 4 | 3 | 2 | 2 | 1 |
| 1 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 3 | 2 | 4 | 1 | 2 | 4 | 4 | 1 | 3 | 2 |
| 1 | 2 | 3 | 1 | 2 | 2 | 2 | 4 | 2 | 1 | 3 | 2 | 4 | 2 | 2 | 4 | 4 | 3 | 1 | 2 |
| 1 | 3 | 1 | 3 | 2 | 2 | 3 | 2 | 1 | 2 | 3 | 3 | 1 | 1 | 1 | 5 | 1 | 1 | 2 | 2 |
| 1 | 3 | 3 | 2 | 2 | 2 | 3 | 3 | 1 | 2 | 3 | 3 | 1 | 2 | 2 | 5 | 1 | 3 | 2 | 2 |
| 1 | 4 | 1 | 3 | 2 | 2 | 3 | 3 | 3 | 2 | 3 | 3 | 2 | 2 | 2 | 5 | 2 | 2 | 2 | 2 |
| 1 | 4 | 4 | 1 | 2 | 2 | 4 | 1 | 3 | 2 | 3 | 4 | 2 | 1 | 2 | 5 | 3 | 1 | 2 | 2 |
| 2 | 1 | 1 | 1 | 1 | 2 | 4 | 2 | 1 | 2 | 4 | 1 | 3 | 2 | 1 | 5 | 3 | 2 | 3 | 2 |
| 2 | 1 | 1 | 3 | 2 | 3 | 1 | 1 | 1 | 1 | 4 | 1 | 4 | 2 | 1 | 5 | 4 | 1 | 3 | 2 |
| 2 | 1 | 2 | 1 | 2 | 3 | 1 | 4 | 3 | 2 | 4 | 2 | 1 | 3 | 2 | 5 | 4 | 4 | 3 | 2 |

It can be seen that Rule 1 is induced correctly (Rule A), Rule 2 has been specialized in two ways (Rules B and D), Rule 3 has been generalized (Rule C) and Rule 4 has not been induced at all. The decision tree induced by ID3 from the same training set is shown in Fig. 5. The bold lines depict the branches for class $\delta_1$.

### 8.1. FAILURE TO INDUCE A RULE

A rule will not be induced if there are no examples of it in the training set (e.g. Rule 4 above). This applies to all induction programs. Even human beings cannot be expected to induce rules from non-existent information.

### 8.2. OVER-GENERALIZATION

An induced rule may be too general if there are no counter-examples to it in the training set. For example, Rule C above ($a_2 \wedge c_4 \rightarrow \delta_1$) is a generalization of the correct rule, Rule 3 ($a_2 \wedge c_4 \wedge d_2 \rightarrow \delta_1$). As there are no instances containing $a_2 \,\&\, c_4 \,\&\, d_1$ or $a_2 \,\&\, c_4 \,\&\, d_3$ in the training set, then there are no counter-examples to $a_2 \wedge c_4 \rightarrow \delta_1$ and no reason to specialize. Any attempts to specialize automatically would have unwanted side-effects on rules which were not too general.

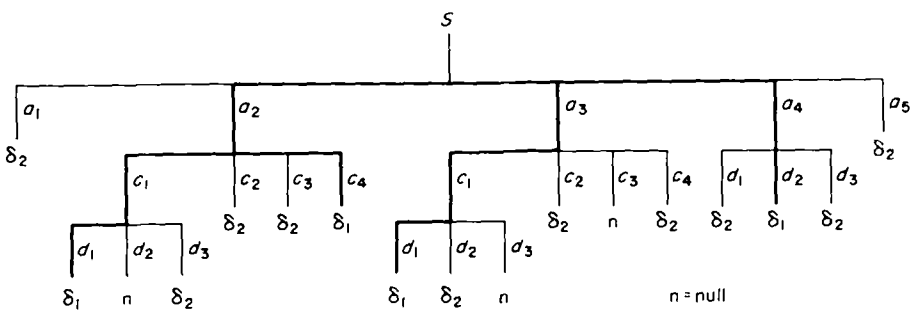


FIG. 5. Decision tree produced from the training set in Table 3.

TABLE 4
*Relative frequency f* vs. *probability p for a small training set*

| $\alpha_x$ | $f(\delta_1 \mid \alpha_x)$ | $p(\delta_1 \mid \alpha_x)$ | $\alpha_x$ | $f(\delta_1 \mid \alpha_x)$ | $p(\delta_1 \mid \alpha_x)$ |
|---|---|---|---|---|---|
| $a_1$ | 0 | 0·083 | $b_4$ | 0 | 0·107 |
| $a_2$ | 0·182 | 0·167 | $c_1$ | 0·267 | 0·357 |
| $a_3$ | 0·333 | 0·083 | $c_2$ | 0 | 0 |
| $a_4$ | 0 | 0·125 | $c_3$ | 0 | 0 |
| $a_5$ | 0 | 0·083 | $c_4$ | 0·167 | 0·071 |
| $b_1$ | 0·222 | 0·107 | $d_1$ | 0·286 | 0·25 |
| $b_2$ | 0·222 | 0·107 | $d_2$ | 0·091 | 0·063 |
| $b_3$ | 0·1 | 0·107 | $d_3$ | 0 | 0 |

8.3. OVER-SPECIALIZATION

Theoretically, the induction algorithm is based on finding the $\alpha_x$ for which $p(\delta_1 \mid \alpha_x)$ is a maximum. In practice, for an incomplete training set, the true probability of occurrence $p$ is unknown, and is approximated by the relative frequency, $f(\delta_1 \mid \alpha_x)$. This approximation of $p$ introduces errors in the estimation of information gain of each $\alpha_x$, which become significant for small training sets, resulting in the selection of an irrelevant attribute–value pair as the best representative of $\delta_1$. Rule B above $(a_3 \wedge c_1 \wedge d_1 \rightarrow \delta_1)$ is an example of this type of error, in which $a_3$ is the unwanted term. The reason for the selection of $a_3$ becomes obvious when the values of $p$ and $f$ for each $\alpha_x$ are compared (see Table 4). It can be seen that $p(\delta_1 \mid a_3)$, is relatively small compared with $p(\delta_1 \mid c_1)$, but as the distribution of $a_3$ is inaccurately represented in the training set, $f(\delta_1 \mid a_3)$ is artifically high, thus leading to the selection of $a_3$ as 'best' attribute–value pair. This in turn leads to the induction of the second too-specific rule, Rule D.

However, this situation can frequently correct itself. Rule B is a specialization of Rule 2, induced incorrectly because of the inaccurate representation of $a_3$ in the training set. Once Rule B has been induced, the instances covered by it are removed from the training set, thus removing the offending bias towards $a_3$. At this stage it is possible that the training set still contains enough instances which are examples of the correct rule, Rule 2, so that Rule 2 can subsequently be correctly induced. As all instances covered by Rule B are also covered by Rule 2, Rule B becomes redundant and can be discarded in the manner described in Section 7.2.2.

These problems are inherent in many induction algorithms and successful solutions to them will be extremely difficult to find.

## 9. Comparison of ID3 and PRISM

This final section demonstrates the performance of PRISM on a training set containing a large number of examples. The training set is provided by the King–Knight–King–Rook chess end-game on which Quinlan performed his original experiments (Quinlan, 1979*a*). The problem is to find a rule set which will determine for each configuration of the four pieces, whether knight's side is lost two-ply in a black-to-move situation. Quinlan tackled the problem in stages, by first

placing severe constraints on the number of allowable configurations of the pieces, and then gradually relaxing these constraints until he could apply his algorithm successfully to the original unrestricted problem. He identified a total of seven problems of increasing complexity. The training set described below is provided by the third of these problems.

There are seven attributes:

*a*: distance from black king to knight, values 1, 2 or 3,
*b*: distance from black king to rook, values 1, 2 or 3,
*c*: distance from white king to knight, values 1, 2 or 3,
*d*: distance from white king to rook, values, 1, 2 or 3,
*e*: black king, knight, rook in line, values t or f,
*f*: rook bears on black king, values t or f,
*g*: rook bears on knight, values t or f.

There are two possible classifications—lost and safe, and the training set consists of 647 instances†. The decision tree produced by ID3 is shown in Fig. 6. It has 52 branches, and if these are trivially converted into separate rules, there are a total of 337 terms. In contrast, the rule set produced by PRISM has 15 rules and 48 terms:

1. $e_f \rightarrow$ safe,
2. $f_f \rightarrow$ safe,
3. $g_f \rightarrow$ safe,
4. $b_1 \wedge d_2 \rightarrow$ safe,
5. $b_1 \wedge d_3 \rightarrow$ safe,
6. $a_1 \wedge c_2 \rightarrow$ safe,
7. $a_2 \wedge c_2 \rightarrow$ safe,
8. $a_1 \wedge c_3 \rightarrow$ safe,
9. $a_2 \wedge c_3 \rightarrow$ safe,
10. $a_3 \wedge b_2 \wedge e_t \wedge f_t \wedge g_t \rightarrow$ lost,
11. $b_3 \wedge c_1 \wedge e_t \wedge f_t \wedge g_t \rightarrow$ lost,
12. $a_3 \wedge b_3 \wedge e_t \wedge f_t \wedge g_t \rightarrow$ lost,
13. $b_2 \wedge c_1 \wedge e_t \wedge f_t \wedge g_t \rightarrow$ lost,
14. $a_3 \wedge b_1 \wedge d_1 \wedge e_t \wedge f_t \wedge g_t \rightarrow$ lost,
15. $a_2 \wedge b_1 \wedge c_1 \wedge d_1 \wedge e_t \wedge f_t \wedge g_t \rightarrow$ lost.

Both the decision tree and the above rule set classify all 647 instances correctly, but an expert system using the decision tree as its knowledge base would require significantly more tests to be performed.

There is also one less obvious difference between the outputs, which is that the decision tree would classify the illegal instance $(a_1 \,\&\, b_1 \,\&\, c_1 \,\&\, d_1 \,\&\, e_t \,\&\, f_t \,\&\, g_t)$ as safe, whereas the rule set produced by PRISM is unable to classify it.

† There is one combination of the seven attributes $(a_1 \,\&\, b_1 \,\&\, c_1 \,\&\, d_1 \,\&\, e_t \,\&\, f_t \,\&\, g_t)$ which is illegal and therefore not included in the training set.

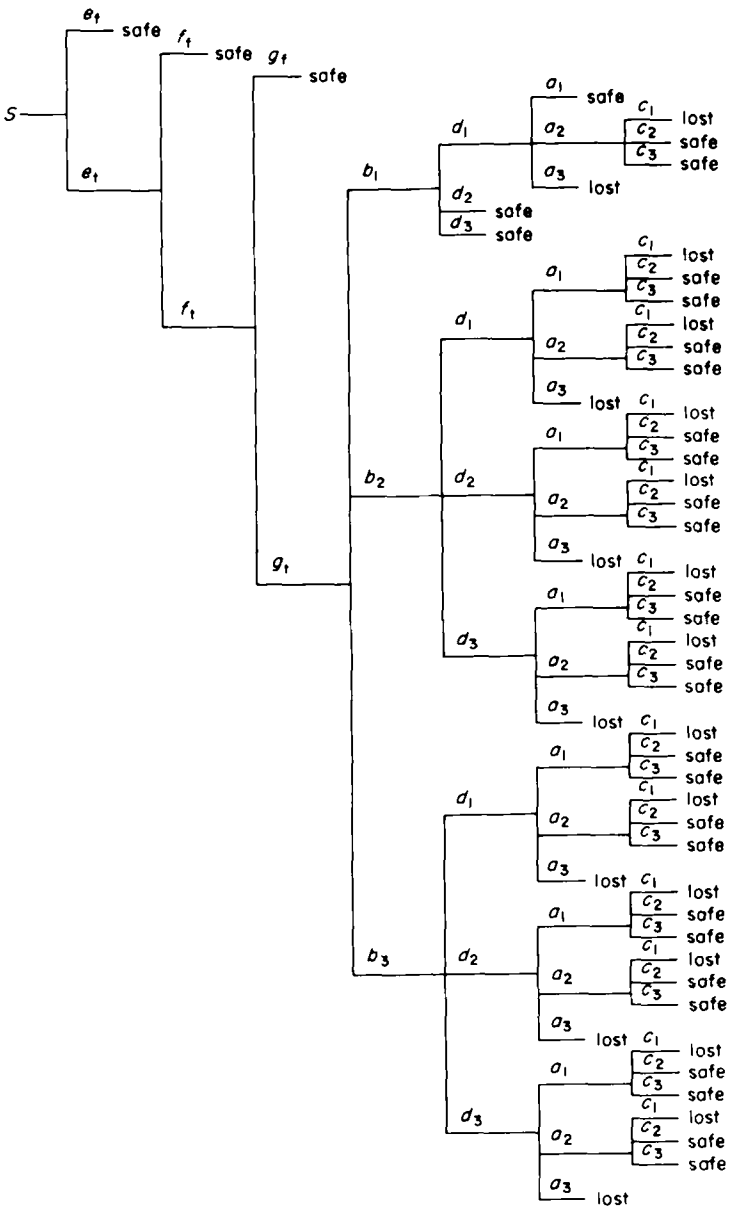


FIG. 6. Decision tree for Quinlan's third problem.

## 10. Summary and conclusions

One of the major criticisms of the ID3 algorithm is that its decision tree output is not suitable for use in expert systems whose control structure is based on the forward or backward chaining of modular rules, particularly if these rules are also used for explanation purposes. Attempts at converting decision trees into modular rules have had limited success because large and complex trees often contain a lot of redundancy, and simplification of these trees requires generalization techniques similar to those used in rule induction. It has been easier to implement expert systems whose control structure is designed to operate on decision trees.

However, the use of unmodified decision trees can have serious consequences in some domains, because the inherent redundancy requires that the results of irrelevant tests be known before a decision can be made. In medicine, these tests may require surgery, or alternatively may take up valuable time; in other domains, they may be extremely costly to perform. An expert system which uses such a decision tree must know the result of a requested test before it can decide on the next test to perform.

Redundancy is clearly an undesirable feature of a decision tree, but as this report points out, it is an inherent weakness in the strategy employed for induction, and can only be remedied by radically altering this strategy. By minimizing the average entropy of a set of instances, ID3 does not pay any attention to the fact that some attributes or attribute values may be irrelevant to a particular classification. This report suggests that a better strategy would be to maximize the information contributed by an attribute–value pair to knowing a particular classification. The report outlines a new induction algorithm, PRISM, which is based on this strategy, and describes some of the results obtained by applying it to different training sets.

PRISM produces its results as a set of modular rules which are maximally general when the training set is a complete one. The accuracy of rules induced from an incomplete training set depends on the size of that training set (as with all induction algorithms) but is comparable to the accuracy of a decision tree induced by ID3 from the same training set, despite the gross reduction in number and length of the rules.

## References

A-Razzak, M., Hassan, T. & Pettipher, R. (1985). EX-TRAN7 (expert translator); a FORTRAN-based software package for building expert systems. In Bramer, R. A. Ed., *Research and Development in Expert Systems: Proceedings of the Fourth Technical Conference of the British Computer Society Specialist Group on Expert Systems.* Cambridge: Cambridge University Press. pp. 23–30

Bundy, A., Silver, B. & Plummer, D. (1984). An analytical comparison of some rule learning programs. *Technical Report 215,* Department of Artificial Intelligence, University of Edinburgh.

Carbonell, J. G., Michalski, R. S. & Mitchell, T. M. (1983). An overview of machine learning. In Michalski, R. S., Carbonell, J. G. & Mitchell, T. M. Eds, *Machine Learning: An Artificial Intelligence Approach.* Palo Alto: Tioga. pp. 3–23

Cendrowska, J. (1984). Practical requirements for rule induction: a critical analysis of current methodologies. *Technical Report,* The Faculty of Mathematics, The Open University, Milton Keynes.

EDWARDS, E. (1964). *Information Transmission.* London: Chapman and Hall.

GOLDMAN, S. (1968). *Information Theory.* New York: Dover Publications.

HART, A. E. (1985). Experience in the use of an inductive system in knowledge engineering. In BRAMER, M. A., Ed., *Research and Development in Expert Systems: Proceedings of the Fourth Technical Conference of the British Computer Society Specialist Group on Expert Systems.* Cambridge: Cambridge University Press. pp. 117–126

LAVRAC, N., VARSEK, A., GAMS, M., KONONENKO, I. & BRATKO, I. (1986). Automatic construction of the knowledge base for a steel classification expert system. In *Proceedings of the Sixth International Workshop on Expert Systems and their Applications.* Avignon, France, pp. 727–740.

MICHIE, D. (1983). Inductive rule generation in the context of the fifth generation. In MICHALSKI, R. S. Ed., *Proceedings of the International Machine Learning Workshop.* University of Illinois. pp. 65–70

O'RORKE, P. (1982). A comparative study of inductive learning systems AQ11P and ID-3 using a chess endgame problem. *Technical Report UIUCDCS-F-82-899,* Department of Computer Science, University of Illinois.

QUINLAN, J. R. (1979*a*). Discovering rules from large collections of examples: a case study. In MICHIE, D., Ed., *Expert Systems in the Micro-Electronic Age.* Edinburgh: Edinburgh University Press. pp. 168–201

QUINLAN, J. R. (1979*b*). Induction over large databases. *Technical Report HPP-79-14,* Heuristic Programming Project, Stanford University.

QUINLAN, J. R. (1983*a*). Learning efficient classification procedures and their application to chess endgames. In MICHALSKI, R. S., CARBONELL, J. G. & MITCHELL, T. M., Eds, *Machine Learning: An Artificial Intelligence Approach.* Palo Alto: Tioga. pp. 463–482

QUINLAN, J. R. (1983*b*). Learning from noisy data. In MICHALSKI, R. S., Ed. *Proceedings of the International Machine Learning Workshop.* University of Illinois.

SHANNON, C. E. & WEAVER, W. (1949). *The Mathematical Theory of Communication.* Urbana: University of Illinois Press. (Published in 1964).