J. Opl Res. Soc. Vol. 37, No. 11, pp. 1031–1037, 1986

0160-5682/86 $3.00 + 0.00

Expert Systems—Experiments with Rule Induction

JOHN MINGERS
Polytechnic of the Southbank

There is currently much interest in the development of expert systems. Knowledge acquisition—developing the knowledge base from an expert—is one of the most time-consuming aspects of the process. Work is under way on methods of automating this procedure, one of which is rule induction from a set of examples. This paper introduces rule induction and evaluates some of the practical difficulties.

Key words: expert systems, knowledge acquisition, rule induction

INTRODUCTION

Recently, there has been much interest in and development of expert systems both outside and inside O.R. From a specialized and esoteric area of artificial intelligence in the 70s, the field has blossomed to the extent that there are now many generalized expert systems or 'shells' available, even for microcomputers. This paper reports on some work done in a particular area—rule induction—both to demonstrate its potential and to illustrate some of the difficulties and limitations of one particular approach to rule induction.

After introducing expert systems and the idea of rule induction, the paper will explain the workings of one particular method—the ID3 algorithm by Quinlan[1]—and outline the major problems with it. Some of these will then be taken up in detail. The domain of statistical significance tests will be used as an example.

EXPERT SYSTEMS AND RULE INDUCTION

General introductions to expert systems are by now fairly common.[2–5] In brief, an expert system is a computer program in which the knowledge of a human expert, much of which may be tacit and practical rather than explicit and theoretical, has been systematically reconstructed. The expert system can then be consulted by a non-expert user, and should be able to offer advice based on the knowledge in the system and further information obtained from the user. The system should be able to explain *why* it took certain actions or decisions, in a way that the user can understand. Clearly, the decisions made by the expert system should be similar to those the expert would have made. There are various ways that the knowledge can be represented in the program, the most common being in the form of a system of production rules.

One of the most difficult and time-consuming stages in the development of an expert system is the elucidation, from the expert, of a comprehensive and consistent set of rules—knowledge acquisition. Coding these rules in an expert system shell is relatively straightforward. Work has been done on developing automatic methods for doing this,[6] although it is fair to say that they are only beginning to deal with an extremely difficult problem. This paper is concerned with one of these methods—that of inducing rules from a set of examples or cases.

THE ID3 ALGORITHM

Explanation of ID3

Each item in the set of examples can be classified as a member of a particular category. The problem is to establish a set of rules which will correctly classify the items, and others like them, in an efficient manner. To do this, the expert decides on relevant attributes for the items, and the rules relate the values of these attributes to the categories.

TABLE 1. *Data for significance tests*

Attributes							Category
No. of samples	Scale	Sample size	Normal distr.	Known SD	Paired samples	Equal SD	Test
1	int	large	—	—	—	—	Z-test
1	int	small	Y	Y	—	—	Z-test
1	int	small	Y	N	—	—	t-test
1	int	small	N	—	—	—	sign-test
2	int	large	—	—	—	—	Z-test
2	int	small	Y	Y	—	—	Z-test
2	int	small	Y	N	Y	—	paired-t
2	int	small	Y	N	N	Y	t-test
2	int	small	Y	N	N	N	U-test
2	int	small	N	—	N	—	U-test
2	int	small	N	—	Y	—	Wilcoxon
3	ord	—	—	—	—	—	Kruskal
2	ord	—	—	—	N	—	U-test
2	ord	—	—	—	Y	—	sign-test

—Null value.

Notes:

(1) The set of examples has been carefully selected to cover the different possibilities, rather than being selected at random.
(2) The attributes are a mixture of numerical, logical and nominal.
(3) The null values occur either because the attribute is meaningless in that particular example (e.g. are the samples paired when there is only one sample?), or because the attribute has meaning but the value is irrelevant (e.g. with a large sample, it does not matter if the SD is known or not).

An example should make this clear. In statistics there are various different significance tests for a sample mean, depending on the characteristics of the sample(s) of data—i.e. the number of samples, whether the data is ordinal or interval, whether the sample is large ($\geqslant 30$) or small, whether it is normally distributed, whether the standard deviation is known, and, if there are two samples, whether they are paired and whether the standard deviations are equal. Here, these characteristics are the attributes and the appropriate test is the category. The induced set of rules should enable the correct test for a particular sample of data to be determined. The data for this example is shown in Table 1, although not all the possible types of significance test are covered. Two sets of trees of rules, generated from this data, are shown in Figure 1(a) and (b).

(a)

```
samples
  one :ssize
       small :normal
              y  :sd
                known   :Z-TEST
              unknown   :T-TEST
              n  :SIGN-TEST
       large :Z-TEST
  two :scale
       ordinal  :paired
                 y  :SIGN-TEST
                 n  :U-TEST
       interval :ssize
                 small :normal
                        y  :sd
                          known   :Z-TEST
                        unknown   :paired
                                   y  :PAIRED-T
                                   n  :sd's
                                        equal    :T-TEST
                                        unequal  :U-TEST
                          n  :paired
                             y  :WILCOXON
                             n  :U-TEST
                 large :Z-TEST
  more :KRUSKAL-WALLIS
```

(b)

```
scale
  ordinal  :samples
            < 3  :paired
                  y  :SIGN-TEST
                  n  :U-TEST
            >=3  :KRUSKAL-WALLIS
  interval  :samples
            < 2  :ssize
                  small  :normal
                         y  :sd
                           known   :Z-TEST
                         unknown   :T-TEST
                         n  :SIGN-TEST
                  large  :Z-TEST
            >=2  :ssize
                 small  :normal
                        Y  :sd
                          known   :Z-TEST
                        unknown   :paired
                                   y  :PAIRED-T
                                   n  :sd's
                                        equal    :T-TEST
                                        unequal  :U-TEST
                        n  :paired
                           y  :WILCOXON
                           n  :U-TEST
                 large  :Z-TEST
```

FIG. 1. *Rule trees for significance tests.*

The basic rule induction algorithm is explained in Quinlan.[7] It is limited in that it deals with only two classes and will not accept numerical attributes—only nominal or logical ones. It works as follows:

(1) Take each attribute in turn, and calculate a measure of how well the values of the attribute split the data into their classes. The measure used is of particular importance and will be explained below. Ideally, each attribute value would only be associated with one particular class, and the data would thus be classified. At worst, the classes may be equally spread between the attribute values, that attribute therefore providing no help.
(2) Choose the best attribute and partition the data according to the values of the attribute.
(3) For each partitioned set, repeat steps 1 and 2 until all the data is correctly classified or no more attributes are available because they have all been used along a particular branch. In this case, a new attribute would be needed to classify the data correctly.
(4) Quinlan was working with very large sets of data (hundreds of thousands) which could not all be dealt with at once, and so had a further loop, in which rule trees were refined by using different sub-sets of the whole data, but this does not really concern us.

Of prime importance is the measure used to evaluate particular attributes, and this is based on Shannon's classic work in information theory.[8]

Suppose we receive one message from a set of n possible messages. How much information does this contain?

If they are all equally likely, then the greater the number of possibilities, the more information the receipt of a particular one will provide. A convenient measure for this information content is:

$$H = \log(n).$$

The base of the log determines the units of measurement, e.g. $\log_2$ gives bits, $\log_{10}$ gives digits. So one digit out of the 10 possible has an information content of

$$\log_2(10) = 3.32 \text{ bits} \quad \text{or} \quad \log_{10}(10) = 1 \text{ digit}.$$

If the messages are not equally likely, then a more general form is

$$H = -K\sum p_i \log(p_i),$$

where K is a constant to do with the units which can be set to 1, and p_i is the probability of the ith message.

This has a minimum value of zero if there is only one possible message, and a maximum value of $\log(n)$ if all messages are equally likely—the situation of maximum information gain.

The measure is used in ID3 in the following way.

To measure the power of a particular attribute in classifying some set of the data, a contingency table can be formed. The following is an example for the attribute 'sample size' in terms of just two classes.

		Class		
Attribute		Z-test	t-test	
Sample size	Small	2	2	4
	Large	2	0	2
		4	2	6

How well does knowledge about sample size discriminate between a Z-test and a t-test in comparison with the other possible attributes?

Using Quinlan's notation, where M and B represent amounts of information and C represents class, and using the relative frequencies as estimates of the probabilities, we can calculate as follows:

The information needed to classify the data without knowledge of the attribute 'sample size' is

$$M(C) = -4/6\log(4/6) - 2/6\log(2/6)$$

$$M(C) = 0.636514.$$

Suppose that the attribute value was known. If it were 'small', then how much information is necessary?

$$M(C|\text{small}) = -2/4\log(2/4) - 2/4\log(2/4) = 0.693147.$$

Similarly,

$$M(C|\text{large}) = 0 \text{ (since it is then definitely a } Z\text{-test).}$$

Taking a weighted average of these two gives a measure of the information needed given the attribute in general:

$$B(C|\text{sample size}) = 4/6*0.693147 + 2/6*0.0 = 0.462098.$$

Finally, the expected information *gain*, given knowledge of sample size, is

$$M(C) - B(C|\text{sample size}) = 0.636514 - 0.462098 = 0.174416.$$

This is calculated for each of the possible attributes and the greatest one chosen at that particular stage. This particular example has used natural logs, but this has no effect on the ordering of the attributes, and the procedure is easily extended to more than two classes or attribute values.

Evaluation of ID3

The original motivation for Quinlan's work was to find a small set of rules which could classify as many as possible of a very large set of known examples. In this, ID3 was very successful. Looking at a particular chess ending (rook vs knight) in which positions can be classified by an expert as lost or not within, say, three moves, there is a total of 1.8 million possible positions, of which 474,000 are lost. Few general rules were known to describe all these particular examples, but ID3 found, in a matter of a few seconds, a rule tree of just 88 leaves which could account for every instance. Determining the 49 attributes to use, however, took Quinlan 2 months!

Within O.R., the possible applications of this method would seem to have a rather different emphasis from that of efficiently classifying extremely large data sets. There are two main areas: first, knowledge which is fairly determinate but is not available in a form appropriate to incorporate in an expert system, e.g. practical knowledge in someone's head or complex procedures, rules or laws; secondly, probabilistic information where data could be collected and the rules extracted which best explain this data in a way analogous to the use of multiple regression.

The major problems

Taking the basic ID3 algorithm as a starting point, there are a number of difficulties as far as its practical use is concerned:

(i) The algorithm cannot deal with numerical attributes, only with logical or nominal ones.
(ii) It is very difficult to decide whether the set of examples and resulting tree are both *correct* and *adequate*, particularly since different trees can easily be generated from the same data.
(iii) The algorithm assumes that the data is completely deterministic and has no contradictions in it. Faced with stochastic data, it produces enormously bushy trees in an attempt to classify every single item. Can it be successfully extended to deal with stochastic data, and what is the validity of the results?
(iv) The rule trees are difficult for humans to interpret and understand and therefore check. Can they be post-processed in some way to overcome this?
(v) The algorithm does not guarantee to find the best rule; and indeed, what might 'best' mean in this context—the most efficient, the smallest, the most easily understandable?

The rest of this article will examine the first two of these problems. Current work on the others will be mentioned under "Further Research".

EXPLORING THE PROBLEMS OF ID3

The following work is based on programs written in Pascal and developed on both a DEC 10 and a BBC micro.

Using numeric attributes

As it stands, the algorithm will only allow logical or nominal attributes, so that each one has only a limited range of known values. Obviously, in O.R. we need to deal with numerical variables (i.e. attributes), either integer or real. In fact, since real values can be made integer to the desired number of significant digits with appropriate scaling, it is only necessary to deal with integer attributes.

This still creates a significant combinatorial problem. An integer attribute may have a large range of possible values, and it is necessary to determine the best set of one or more break-points within this range to split it up into sub-ranges. One way round the problem entirely is to get the expert to specify the appropriate ranges and thereby effectively make the attribute a nominal one. However, it would be more useful, particularly in uncertain situations, for the system to work out the best possibilities itself from the actual data. In this case, it must be done not just once but every time the attribute is evaluated within the tree, as the actual data available at each branch will differ.

To give an idea of the computational scale of this problem, consider an attribute with values in the range 100–200. It may well be that not all of these possible values occur—say there are 30 actual possibilities, including the end-points. To split this into just two ranges, 28 possibilities (i.e. 30 less the end-points) need be considered. To split it into three ranges, two break-points are needed. There are ${}^{28}C_2$ (378) possibilities, and to split it into four ranges there are ${}^{28}C_3$ (3276) possibilities. Allowing for five ranges gives 20,475—a total of 24,157 in all. Each of these possibilities has to be evaluated in terms of the information measure, as if it were an attribute in its own right, at each point in the tree. This would increase the run-time by the order of thousands.

Some attempts were made to pre-analyse the data in some way to pick out the most promising ranges of values, but examples could always be constructed where they missed out the best possibilities.

In the end, the only way forward was to restrict consideration to the case of only two ranges of values—i.e. only one break-point—thus giving 28 possibilities to examine in the above example. Although this may seem a major restriction, in practice it is not, because the effect of a number of ranges can be maintained by allowing the same integer attribute to be used repeatedly down a branch of the tree. For example, suppose an attribute is first split into the ranges $100 < 135$ and $135 \leqslant 200$. Following the first branch, the system may subsequently choose that attribute again and split it into $100 < 115$ and $115 \leqslant 135$, so that within the tree as a whole, the attribute has actually been split into three ranges. This is not to say, of course, that the same three ranges would have been chosen if it were allowed to consider them all at once. Examples of this will be discussed below.

Assessing the correctness and adequacy of rule-trees

After producing a rule-tree from a set of examples, the problem is to assess the validity of the tree. There are two different dimensions involved here—first, its correctness, and secondly, its adequacy.

A rule-tree is correct if the rules in it are judged by the expert not to be in error. This is mainly a matter of developing an appropriate set of attributes, ensuring that there are no mistakes in the data and resolving any contradictory examples. The main problem is the difficulty of actually comprehending and fully grasping all the rules embodied in the tree, because this is not a very natural way for humans to think. People would generally tend to group together rules giving similar outcomes; for example:

Do a Z-test IF (interval data AND large sample)
OR (interval data AND small sample AND standard deviation known)

Here, the ANDs are going along a branch, while the ORs are combining across branches.

However, having a tree which appears to be correct does not necessarily mean that the tree will actually be adequate for the real problem domain, nor even that it will be the best or only tree

to fit that particular set of data. A rule-tree will be adequate only if it can deal correctly with *all* the possibile situations it might have to face. Yet this can never be guaranteed, since the data on which it is based will inevitably be only a sample of all the possibilities. There will almost certainly be particular combinations of the various attributes and their values which are not thought of and made explicit. It is not easy to spot these combinations by examining a single tree, as they often involve fairly subtle distinctions. One helpful approach is to compare two or more *different* trees generated from the *same* data. It is quite easy to generate different trees in a number of ways. In the algorithm, the attributes to branch on are chosen by the information measure but could be chosen by any method—even randomly. The resulting trees would be inefficient but would still correctly classify the data.

The sorts of possibilities which can easily be missed will be demonstrated with rule trees generated from the significance test data. The rule trees are shown in Figure 1(a) and (b). The reason for the difference between these two is that in Figure 1(a) the attribute 'number of samples' was treated as nominal with the values one, two or more, whereas in Figure 1(b) it was treated as an integer attribute with a range of values 1–3. These rules have been induced from the data like a decision tree. To decide which significance test to use with a particular sample of data, the rules [in Figure 1(a)] suggest first checking how many samples—one, two or more. If there is one sample, check the sample size—small or large. If the sample size is large, then a Z-test is recommended; otherwise check if the distribution is normal and so on.

The difference in treatment of the 'number of samples' attribute has led to quite different trees. With size treated as nominal, it has three values and is chosen as the first test. When treated as integer, however, it has only two possible values ($<$ or $\geqslant$) and is no longer chosen first. This is because the information measure tends to be larger the more rows or columns there are in the contingency table (like the chi-square test to which it is related). This leads it to be biased in favour of attributes with a greater number of possible values. Notice that different sub-ranges have been chosen in the 'ordinal' and 'interval' branches of the tree.

At first sight, either of these trees seen by itself appears correct, but in fact both have limitations, as shown below. The difficulty is in fully comprehending the implications of the tree.

(i) In (a), the Kruskal–Wallis test is selected if there are more than two samples, without checking that the data is ordinal, whereas in (b) it can only be chosen if the data is ordinal. This is because there is only one example of more than two samples, and so it can be classified without reference to any other attributes.

(ii) In (a), where there is only one sample, the Z-test and t-test could be selected without checking that it is interval data. This is because all the one-sample examples are interval data, so the scale attribute does not distinguish between them. In (b), where scale is the first attribute, this problem does not arise.

(iii) In (b), if the scale is ordinal and there are less than three samples (i.e. one or two), it checks whether the samples are paired. This clearly makes no sense if there is in fact only one sample. Again, this is because there are no examples of one-sample ordinal data.

All these demonstrate the same points. A tree by itself might well look correct at first sight, but on closer inspection may not cope with all situations in the right way.

CONCLUSIONS

This paper has introduced the idea of inducing rules from sets of examples in order to speed up the development of knowledge bases in expert systems. Based on Quinlan's ID3 algorithm, this approach does efficiently produce logically structured sets of rules. However, there are significant problems with the approach as it stands:

(i) It is difficult to ensure that a set of rules induced from a sample of particular instances is adequate for dealing correctly with its entire domain of application, even in well defined domains where there is no noise in the data. First, a rule tree can only take account of the actual examples it is given, and it can be very difficult to detect any shortcomings in the data set from a single tree, not least because it does look so logical. Secondly, even with

a comprehensive set of examples, different trees can be generated correctly from the *same* data, and these will classify certain instances not included in the actual examples differently. Having more than one tree helps in the process of spotting holes in the data but by no means guarantees the results.

(ii) ID3 will only work satisfactorily with deterministic data, and yet this is relatively rare within the practical world of O.R. Work has been done on this area, both by Quinlan[9] and Hart,[10] in trying to reduce the very large, bushy trees which are produced. Further results in extending ID3 to deal with stochastic data will be reported in a future paper.

(iii) Rule trees, while appropriate for use in computer systems, are not easily understood by humans—certainly not in terms of their detailed implications. This contributes to the problem of assessing their validity, and limits their use outside computer programs—for example in manuals or for stimulating discussion or ideas.

Nevertheless, this is an interesting and potentially useful area worthy of future research.

AREAS FOR FURTHER RESEARCH

(i) Rule induction, in general terms, is a method for discovering underlying relations in sets of examples. As such, it should be comparable to well known statistical methods such as multiple regression and, more particularly, discriminant analysis. It would therefore be very useful if studies could be undertaken comparing induction with these methods to discover if it is comparable or even better, and if so, under what conditions.

(ii) To help with the problems of rule validity, methods for assessing the reliability of a rule tree should be explored—for example, special test data such as actual examples from the situation, randomly generated examples or even an attempt at systematically testing all the possible combinations of attribute values.

(iii) The fact that different trees can correctly represent the same data raises the question of whether one is better than another. Is it a matter of size or efficiency—the fewer branches and leaves the better? Or is it that the tree should match the expert's way of thinking, or that it should be the most easily understood, or that there should be some quantitative measure?

(iv) The information measure used by Quinlan is only an heuristic—other simpler ones are used by Hunt *et al.*[11] in their original work, and there are also statistical measures such as chi-square that perform the same function. How do these all compare with each other? Also, are there are more efficient ways of choosing the ranges for numerical attributes?

(v) Finally, methods of making the rules more transparent are needed. This could be either by post-processing the rule tree and re-arranging the rules so that they are more easily understood, or by structuring the way they are generated in the first place. Some work has been done in this latter area by Michie and Shapiro which they call structured induction.[12,13]

REFERENCES

1. J. R. Quinlan (1979) Discovering rules from large collections of examples: a case study. In *Expert Systems in the Micro Electronic Age* (D. Michie, Ed.). Edinburgh University Press.
2. R. Forsyth (Ed.). (1984) *Expert Systems: Principles and Case Studies*. Chapman & Hall, London.
3. D. Michie (Ed.) (1982) *Introductory Readings in Expert Systems*. Gordon & Breach, New York.
4. D. Michie and R. Johnston (1984) *The Creative Computer: Machine Intelligence and Human Knowledge*. Viking, London.
5. D. Lenat (1984) Computer software for intelligent systems. *Scient. Am.* 152–160.
6. R. Michalski, J. Carbonell and T. Mitchell (Eds) (1984) *Machine Learning: An Artificial Intelligence Approach*. Springer, New York.
7. J. R. Quinlan (1984) Learning efficient classification procedures and their application to chess end games. In *Machine Learning: An Artificial Intelligence Approach* (R. Michalski *et al.*, Eds). Springer, New York.
8. C. Shannon and W. Weaver (1949) *The Mathematical Theory of Communication*. University of Illinois.
9. R. Quinlan (1983) Learning from noisy data. *Proceedings of the International Workshop on Machine Learning*, Department of Computer Science, Illinois University.
10. A. Hart (1984) Experience in the use of an inductive system in knowledge engineering. In *Research and Developments in Expert Systems* (M Bramer, Ed.). Cambridge University Press.
11. E. Hunt, J. Marin and P. Stone (1965) *Experiments in Induction*. Academic Press, New York.
12. D. Michie (1984) Automating the synthesis of expert knowledge. *ASLIB Proc.* **36**, No. 9, 337–343.
13. A. Shapiro and T. Niblett (1982) Automatic induction of classification rules for a chess end game. In *Advances in Computer Chess* (M. Clarke, Ed.), Pergamon Press, Oxford.