Generating Rules From Examples

Bijan Arbab IBM Los Angeles Scientific Center, USA

Donald Michie Turing Institute, Glasgow, UK

ABSTRACT

Automatic decision-tree generators have been used in production of expert systems. A number of experiments indicate the need for more linear (and hence more understandable) yet efficient decision trees. An algorithm is implemented for constructing decision trees optimized with respect to linearity It also improves on a previous linearizing algorithm (AOCDL) with respect to execution efficiency.

INTRODUCTION

Thii work describes tools for generating decision trees that are optimized with respect to linearity and are more efficient than those generated by Bratko's AOCDL [3] The rule generator is specialized to as to obey stated constraints corresponding to the above two properties. Rule induction takes advantage of one of the expert's most reliable and highly developed skills [9], teaching by example, and avoids the need to resort to dialogue-acquisition of rules, traditionally recognized as the bottle-neck problem of knowledge engineering. However, decision trees derived from situation-action pairs are inherently less descriptive for expressing concepts than first-order or multivalued logic used in other projects [16] [8] [5], The lack of descriptive power is primarily associated with the absence of quantified variables.

R. Quinlan [12] and A Shapiro [13] have demonstrated that generation of decision trees from a set of examples provided by a domain expert is a practical method for knowledge acquisition (see also A Martelli and U. Montanari [6] for generation of optimal trees). Quintan's ID3 uses an information theoretic approach to control a "best-first no backtrack" search, producing decision trees of high, but not optimal, execution-efficiency. Bratko's AOCDL uses backtrack heuristic search. ID3 ignores the human understandability criterion for induced rules while AOCDL ignores the efficiency criterion. ID3's attribute selection criterion, based on entropy, promotet efficient decision tree execution on the machine. However, the decision trees are not easily understood by humans. AOCDL is beuristically guided by a non-linearity (branching) measure Arbitrarily branching structures are hard for a human to keep mental track of. So one idea is to only allow for linear or almost linear decision trees [10]. A decision tree is said to be linear if every node has at most one non-terminal son. Note that even trees with high branching ratios (multiple-value-attributes) and multiple decision classes can be linear. The relation between linear trees and understandability has been experimentally investigated by Shapiro and Niblett in [14] [15]. In two separate classification tasks in chess end-games, structured representations with tree-linearity constraint were uniformly understandable, whereas representations in the form of arbitraily branching decision trees were uniformly opaque. Our Rule Generator (RG) produces decision trees that are linear where such trees exist. In cases where such a tree does not exist, the most linear tree is constructed. The derived trees are efficient at execution time. These two requirements, linearity and efficiency, are inversely related. A balanced tree is shallower and more efficient for machine execution than a linear tree. In synthesizing decision trees, however, we always trade efficiency for linearity, in much the same way that structured programming trades efficiency for program clarity and readability.

The presence of a domain expert makes "structured induction" possible, which breaks the problem into subproblems A detailed description of structured induction is given by Shapiro and Niblett [13] [14]. With structured induction the size of the example sets is never large, e.g., at most in the order of tens It has been found by Quinlan [12] that small example sets are sufficient to generate rules capable of classifying even large domains with high reliability.

We have developed RG under the assumptions that:

1 Structured induction is feasible

2 Linearity of decision trees is to be optimized even at the expense of efficiency.

3 Efficiency of decision trees is to be increased only subject to the constraint that linearity is not affected

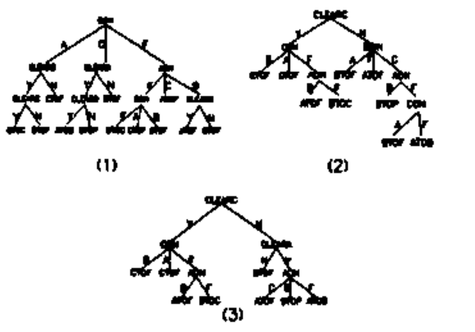


Figure 1: Trees with different linearity and efficiency

Decision trees in Figure 1 correspond to an example set taken from planning domain for building an arch, see [4]. Each node of the tree corresponds to an attribute, leaf nodes represent decision classes, and the labels on the arcs are the attribute's values. These trees were induced by (1) Expert-Ease [7], a commercial version of the ID3-derived ACLS algorithm which produces efficient but non-linear trees; (2) AOCDL, which maximizes linearity but not efficiency; and (3) RG, which maximizes linearity and promotes efficiency. The exact non-linearity and efficiency (execution cost) measures of these trees can be seen for comparison in Figure 7 under EX4.

RG incorporates linearity and efficiency measures within an AO* [11] algorithm as heuristics to guarantee optimal linearity. The efficiency of the decision trees is increased according to each candidate attribute's expected information contribution if appended at the given point in the tree, i.e. attributes with high information content will be placed as high in the decision tree as possible thus increasing probability of classification to occur as early as possible.

DESIGN PRINCIPLES

We mentioned that experts are generally adept at communicating their expertise by means of examples. The examples thus form a language through which knowledge is communicated. There are three parts to this language: attributes, classes, and examples, the latter being defined in terms of attribute values and classes

With respect to a specified example set an attribute has decider status: total, partial, or non-decider An attribute's decider status is defined as follows:

1. Total Decider, if the attribute partitions the example set such that each partition belongs to a single class

2. Partial Decider, if the attribute partitions the example set such that all but one partition belong to a single class.

3. Non-Decider, if neither of the above is true.

Decider status of an attribute plays an important role in our search for linear decision-trees. If an attribute must be selected from a set of total or partial deciders, then the linearity of the final tree is not affected by the choice of a particular attribute, but efficiency can depend on this choice. However, selection of a non-decider attribute can affect efficiency and invariably destroys linearity. Consider the example set in Figure 2. For simplicity we have assumed binary attributes and only two classes.

A1	A2	A3	A4	A5	class
t	f	f	t	t	c1
t	f	t	f	f	c1
t	f	f	t	f	c1
f	t	f	t	t	c2
t	t	f	t	f	c2
t	f	t	t	f	c2
f	f	t	f	t	c1
t	f	f	t	t	c1
t	t	t	t	f	c2
f	f	f	f	f	c1
f	t	f	f	t	c2

Figure 2: An example set

It so happens that in the above example set all candidate attributes for the top of the tree are non-deciders However, different attributes lead to various non-linear trees Using attribute A2 at the top leads to a tree of the form shown on the left side of Figure 3 while using either one of attributes A1, A3, A4 or A5 leads to a tree of the form shown on the right.

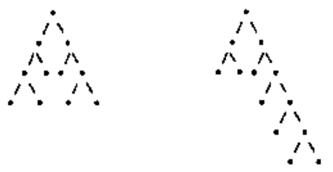


Figure 3: Trees with different linearity measure

Clearly the tree on the right is a more linear tree (we recall that linear decision trees are easier to understand). Therefore, when selecting an attribute one must consider their effect on the overall linearity and efficiency of the decision tree. In general, making the right selection requires a search procedure which is described in later sections.

We have mentioned linear versus non-linear trees and have used degree of linearity as a measure of desirability for trees. This concept mutt be formalised to allow comparison of trees on the basis of their non-linearity. Some desirable characteristics of a function to compute non-linearity of trees are:

1. An intuitive, yet formal, basis.

2. Sensitivity to the size of trees.

3 Sensitivity to location of non-linearity in a tree.

Bratko [3] has proposed such a function The non-linearity measure which he proposes is based on the fact that traversal of a linear tree requires scanning through contiguous memory locations and minimizes jumps across the memory. This is one possible reason why linear decision trees are easier to understand than are non-linear trees.

Let T be a decision tree whose root is A and subtrees are S1, S2, ... Sm, as in Figure 4.

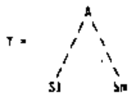


Figure 4: An abstract tree

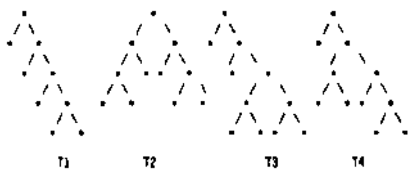
The proposed measure for Non-Linearity is:

$$NL(T) = (1/m) \times \sum_{i=1}^{m} [NL(Si) + (m-i) \times s(Si)]$$

where NL(T) denotes Non-Linearity of T. NL(T)=0 when T is a class value (leaf node) and the number of internal nodes of the tree, s(T), is defined as follows:

$$s(T) = 1 + \sum_{i=1}^{m} s(Si)$$

where s(T)=0 if T is a class value It is assumed that Si are sorted in increasing order of s(Si) Non-linearities of four trees are shown in Figure 5



NL(T1) = 0
NL(T2) = (1 / 2) × [(2 − 1) × s(S1) + (2 − 2) × s(S2) + NL(S2)] = 1
NL(T3) = 1 / 8
NL(T4) = 1 / 4

Figure 5: Non-linearity measure of four trees

TI is absolutely linear thus its non-linearity measure is zero. T2 is veTy close to being a balanced tree, non-linearity one. T3 is preferred to T4, i.e this function is sensitive to the location of non-linearity within a tree (the lower non-linearities occurs in a tree the lower (better) its measure).

Consider the example set of Figure 2. Two equally linear decision trees for classifying this example set are shown in Figure 6

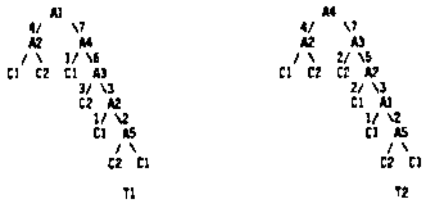


Figure 6: Trees with different execution cost

Labels on the arcs correspond to the number of examples per value of each attribute Let c(a,) represent the execution cost of an attribute. There are 11 examples in the original example set and the execution cost for each tree can be computed on the basis of how early in the decision tree a classification takes place, one way of computing this cost is as follows:

$$\textit{Average cost for T1} =$$

$$\frac{[11 \times c(a_1) + 4 \times c(a_2) + 7 \times c(a_4) + 6 \times c(a_3) + 3 \times c(a_2) + 2 \times c(a_5)]}{11}$$

$$\textit{Average cost for T2} =$$

$$\frac{[11 \times c(a_4) + 4 \times c(a_2) + 7 \times c(a_3) + 5 \times c(a_2) + 3 \times c(a_1) + 2 \times c(a_5)]}{11}$$

Assuming execution cost of each attribute has unit cost, c(a,)=I. the execution cost for trees T1 and T2 are 3 0 and 2 9 respectively, i.e. T2 is about 3% more efficient than Tl Thus, it is desirable for attributes with high information content (entropy) to appear as early as possible in a decision tree This increases the probability of a classification to occur as soon as possible RG employs entropy as the selection criterion for increasing efficiency, as in [12]

We adapted Bratko's measure of non-linearity and used an attribute selection criterion that promotes execution efficiency of the resulting decision tree RG incorporates the notions of linearity and efficiency into an AO* [11] search technique. Details of RG are in [1] [2]

The state space for finding a decision tree is finite and decreasing with the number of variables since the number of attributes and examples are finite An "And/Or" tree is used to represent the state space "OR" nodes correspond to candidate attributes and "And" nodes are subproblems that must be solved The root can be considered as an "And" node Each node may be labeled as solved, closed or open. Solved nodes mean that a solution has been reached from this node, a closed node means that a solution under current consideration incorporates this node internally, a node is open if it is neither closed nor solved.

During the expansion of the search tree, an optimistic estimate for non-linearity is used in conformity with the AO* algorithm for searching "And/Or" graphs This estimate differentiates between total, partial and non-decider attribute Thus, if there are total deciders among the candidate attributes the search tree is expanded using them and the nodes are labeled as solved All partial decider attributes are considered if no total deciders exist and non-decider attributes are considered only if there are no total or partial deciders

The optimal solution path is marked in the search tree according to (1) non-linearity of the partially constructed decision tree; (2) number of expected internal nodes and; (3) the attribute's entropy measure. The entropy measure is used simply as a tie breaker between attributes which produce equally linear decision trees Thus, optimality with respect to linearity is guaranteed while efficiency is only enhanced. When RG terminates the optimal decision tree can be constructed by tracing markers from the root node to the bottom and recording the attributes and their values

RESULTS WITH RG

RG was used to induce rules for some examples selected from the planning domain (construction of an arch and sorting a stack of blocks) and chess-end games (some examples from Shapiro's Ph.D. thesis [13]) in addition to some artificially constructed example sets. For the most part the rules synthesized by RG were more linear than those induced by ID3. The exceptions occurred when ID3 happened to construct a fully linear decision tree. ID3 produces more efficient decision trees than AOCDL or RG. This is to be expected because RG (and AOCDL) emphasizes the linearity criterion before efficiency. However, the decision trees generated by RG were more efficient than those produced by AOCDL since this program, AOCDL, only optimize! the linearity criterion. The decision trees generated by RG are more understandable than those generated by AOCDL, since RG optimizes efficiency without destroying linearity. For an example see Figure 1. The following are five examples that demonstrates the differences between these programs; see Arbab [1] for listings of the examples.

	RG		AOCDL		ID3	
	NL	Cost	NL	Cost	NL	Cost
EX1	0	2.7	0	2.7	0	2 7
EX2	0.5	2 91	0.5	3 0	1.0	2 54
EX3	0	2.25	0	2.25	0 5	2.0
EX4	1 0	2 76	1 0	3.5	2 11	1.12
EX5	0	3 0	0	3.42	0	3.0

Figure 7: Performance analysis of programs

The above table indicates that RG produces decision trees that are as linear as those produced by AOCDL but more efficient, also, the produced decision trees are more linear than those produced by ID3 Therefore, RG has been successful in its goal i e producing the most linear decision tree while enhancing execution efficiency

ACKNOWLEDGMENTS

We would like to thank Stephen Crocker, Michel Melkanoff and Stott Parker for their continued advice, and Farhad Arbab for his comments on this paper, and Gary Silverman, Jim Moore and Rina Dechter for providing many helpful suggestions throughout the project Many thanks arc also due to the IBM Los Angeles Scientific Center for providing resources and support

BIBLIOGRAPHY

[1] Arbab, B.. Building Expert Systems by Generating Rules from Examples, IBM Los Angeles Scientific Center report (LA. CA. 1984)

[2] Arbab, B and Michie, D.. Generating Expert Rules from Examples in Prolog, Machine Intelligence 11 (Eds Hayes J. E, Michie. D and Richards. J) (1985)

13] Bratko, I, Generating Human-Understandable Decision Rules. E Kardelj University Ljubljana (Working paper) (Yugoslavia, 1983).

[4] Dechter, R and Michie, D, Structured Induction of Plans and Programs, IBM Los Angeles Scientific Center report (LA, CA, 1984)

[51 Hayes-Roth, F and McDermott, J., Knowledge Acquisition from Structural Description. Proceedings of the Fifth IJCAI (Cambridge, Mass., 1977) p 356 to 362

[6] Martelli, A. and Montanari, U.. Optimizing Decision Trees through Heunstically Guided Search, Communications of the ACM 21 (1973) p 1025 to 103

[7] McLaren, R., *Expert-East User Manual,* Glasgow Intelligent Terminals Ltd (1983)

[8] Michalski, R. S, Pattern Recognition as Rule-Guided Inductive Inference, IEEE Transactions on Pattern Analysis and Machine Intelligence Vol. PAM1-2, No. 4 (1980) p 349 to 361.

[9] Michie. D., The State of The Art in Machine Learning. *Introductory Readings in Expert Systems (Ed. D. Michie)* (1982) p 208 to 228.

[10] Michie, D., 'Mind-like' Capabilities in Computers, a note on computer induction, Cognition 12 (1983) p. 97 to 108.

[11] Nilsson, N *J., Principles of Artificial Intelligence.* Tioga Publishing Co. (Palo Alto, CA, 1980).

[12] Quinlan, J. R., Learning Efficient Classification Procedures and their Applications to Chess end-games, *Machine Learning; An Artificial Intelligence Approach (Eds. Michalski. R. S.. Carbonel, G. and Muchell. T.)* (Palo Alto, CA. 1982) p. 391 to 411.

[13] Shapiro, A., *Ph.D. thesis. The Role of Structured Induction in Expert Systems.* University of Edinburgh: Machine Intelligence Research Unit (1983).

[14] Shapiro, A. and Nibleit, T., Automatic Induction of Classification Rules for a Chess Endgame, *Advances in Computer Chess 3 (Ed. Clarke. M. R. B.)* (1982) p. 73 to 92.

[15] Shapiro. E. Y., *Inductive Inference of Theories From Facts.* Yale University: Department of Computer Science (1981).

[16] Vere, S. A., Inductive Learning of Relational Productions, *Pattern-Directed Inference Systems (Eds. Waterman. D. A. and Hayes-Roth. F)* (New York, 1978).