To the special memory of Cecylia Rauszer—
an outstanding scientist, a magnanimous human being, and a dear friend.

On Learning Decision Structures

Ryszard S. Michalski* and Ibrahim F. Imam

George Mason University
Fairfax, VA. 22030
{michalski, iimam}@aic.gmu.edu

*Also with the Institute of Computer Science,
Polish Academy of Sciences, Warsaw, Poland

Abstract

A decision structure is a simple and powerful tool for organizing decision processes. It differs from a conventional decision tree in that its nodes are assigned tests that can be functions of the attributes, rather than single attributes; the branches stemming from a node can be assigned a subset of attribute values rather than a single attribute value (test outcome); and the leaves can be assigned one or more alternative decisions. This paper describes a methodology for learning decision structures from declarative knowledge expressed in the form of decision rules. The decision rules are generated by an expert, or by an AQ-type inductive learning program (with or without constructive induction). From a given set of rules, one can generate many different decision structures. The proposed methodology generates the one that is most suitable for the given decision-making situation, according to a multicriterion cost function. Experiments with a program implementing the methodology have indicated many advantages of the proposed methodology.

Key words: machine learning, inductive learning, decision structures, decision trees, decision rules, attribute selection, knowledge acquisition, data mining, knowledge discovery.

1 Introduction

To make a correct decision, a decision maker (a human or an intelligent system) needs to know how the choice of decision depends on the characteristics of the decision-making situation. These characteristics are determined by available observations, test results, or other information. A simple structure for representing a decision process, which has been widely used in machine learning and related areas, is a decision tree. In many real-life problems, however, a decision tree that captures fully the decision process would be unnecessarily complex, because of the limited representational power of decision trees. In addition, current machine learning methods for determining a decision tree are inflexible, in the sense that they specify a decision process once and forever, irrespective of the possible changes in the decision-making situation. For example, if a test assigned to some node of the tree is not available in a given situation, the decision process cannot proceed (unless some statistical technique is applied). This stands in contrast with human decision-making, in which a decision maker would seek a substitute test.

The methodology presented in this paper helps to solve both of the above problems. It uses a more powerful *decision structure* rather a decision tree for representing decision processes. To solve the second problem, it creates a decision structure from declarative knowledge, expressed in the form of decision rules, rather than from examples of decisions, as in methods for learning of decision trees (e.g., Quinlan, 1993). The decision rules are learned by an inductive learning program from examples, or supplied by an expert. The main advantage of this approach is that from a single set of decision rules one can potentially generate a large number of alternative decision structures, so the system is free to chose the structure that is most appropriate for a given decision making situation.

2. Problem Definition

A *decision structure* is an acyclic graph that specifies an order in which tests are to be applied in the given decision-making situation to arrive at a decision. The nodes of a decision structure are assigned tests, which may involve determining a value of a single attribute, a function of attributes, or a relation. The branches stemming from a node are assigned disjoint subsets of the test's possible outcomes, and the leaves are assigned a specific decision, or a set of candidate decisions (in case of having insufficient information to drive a decision). A single decision structure can represent in a simple and understandable way a very complex relationship between the outcomes of the tests and the assigned decisions.

A decision structure reduces to a familiar decision tree when each node is assigned a single attribute, the branches from each node are assigned single values of that attribute, and leaves are assigned single. Thus, the problem of generating a decision structure is a generalization of the problem of generating a decision tree. Decision trees are typically learned from a set of examples of decisions. The essential characteristic methods for learning decision trees is an *attribute selection criterion* employed for choosing attributes to be assigned to the nodes of the decision tree being built. Such criteria include entropy reduction (e.g., Quinlan 1979 and 1983), the Gini index of diversity (Breiman et al, 1984), and others (e.g., Mingers, 1989; Chestnik and Bratko, 1991; Chestnik and Karalic, 1991).

A decision tree/structure be an effective tool for describing a decision process if the required tests can be measured, and the class of decision-making situations it was designed for remains constant. A problem will arise when these assumptions do not hold; for example, in some situations, measuring some attributes may be too difficult, too costly, or just impossible. In such situations it is desirable to reformulate the decision structure so that the available and/or “inexpensive” attributes are evaluated first (that is, are assigned to the nodes close to the root), and the "expensive" attributes are evaluated only if necessary (that is, are assigned to the nodes as far away from the root as possible). If a certain attribute cannot be measured, it is useful to either modify the structure so that it does not contain that attribute, or—if this is impossible—to indicate alternative candidate decisions with their probabilities. A restructuring may also be desirable, if there is a significant change in the frequency of occurrence of different decisions.

Restructuring a decision structure (or a tree) in order to suit new requirements is usually quite difficult. This is because a decision structure is a form of procedural knowledge representation, which imposes an evaluation order of tests. In contrast, no evaluation order is imposed by a declarative representation, such as a set of decision rules. Tests (conditions) of rules can be evaluated in any order. Thus, for a given set of rules, one can usually build a large number of logically equivalent decision structures (trees), which differ in the test ordering. Due to the lack of order constraints, a declarative representation (e.g., a ruleset) is much easier to adapt to different situations than a procedural one (e.g., a decision structure or a decision tree). On the other hand, to apply decision rules to make a decision, one needs to decide the order in which tests are evaluated, and thus, one needs a decision structure.

An attractive resolution of these opposite requirements is to acquire and store knowledge in a declarative form, and transform it to a decision structure when it is needed for making a decision. This method allows one to create a decision structure that is most appropriate for a given decision-making situation. Because the number of decision rules per decision class is usually small (much smaller than the number of training examples per class), the process of generating a decision structure from decision rules can be potentially done much faster than generating it from training examples (e.g., Quinlan, 1979; Breiman et al, 1984). This feature has been confirmed in our experiments (see Section 3). Because the process of generating a decision structure from decision rules can be done very quickly, it can be done at the time when a decision structure is needed "on line," without any delay noticeable to the user. In this way, a decision structure can be tailored to any specific decision-making situation.

This approach allows one to generate a decision structure that avoids evaluating an attribute that is difficult or costly to measure. Initial ideas on this approach, and the first system implementing it, AQDT-2, have been described in [Imam and Michalski, 1993]. The original idea for this methodology stems from the research presented in [Michalski, 1978].

This paper presents an extension of the earlier work and a new system, AQDT-2. The new system generates a goal-oriented decision structure from decision rules learned by either the rule learning system AQ15c [Michalski et. al, 1986; Wnek at al., 1995] or AQ17-DCI (Bloedorn et al, 1993; Bloedorn and Michalski, 1997]. By using AQ17-DCI, which has extensive constructive induction capabilities, "oblique" decision structures can be generated (in which nodes corresponding to tests are functions of the original attributes).

Other novel features of the system include a method for controlling the degree of generalization needed during the development of the decision structure, new attribute selection criteria based on decision rules, a new method for combining attribute selection criteria, the ability to generate "unknown" nodes in situations in which there is insufficient information for generating a complete decision structure, the ability to learn decision structures from discriminant decision rules as well as characteristic rules (Michalski, 1983), and finally, the ability to provide the most probably correct decision when the decision process stops due to the inability to measure an attribute associated with some node. The following section describes details of this methodology.

3 Methodology

The methodology assumes that input to the process is in the form of decision rules in VL1 (variable-valued logic system one), which are generated by an AQ-type learning program. For the sake of generality, the methodology can also accept as input a set of examples in the form of sequences of attribute-value pairs. The decision rules may include original attributes, or *derived attributes*, which are functions of the original ones. Derived attributes are generated by the constructive induction program AQ17-DCI (Bloedorn and Michalski, 1997; Bloedorn et al., 1993).

The top-level algorithm proceeds in a way similar to standard methods of building a decision tree from examples. The major difference is that it assigns tests to the nodes using criteria based on the properties of the decision rules, rather than criteria based on the coverage of individual training examples. Other differences are that branches of the structure may be assigned not a single test outcome, but a subset of possible outcomes (corresponding to the internal disjunction of values in a condition of a rule). Individual tests are either single original attributes, or names standing for logical or mathematical expressions involving these attributes. These names represent derived attributes generated by the program for constructive induction AQ17-DCI. From now on, we use terms a "test" and an "attribute" interchangeably.

At each step, the algorithm searches through a set of tests appearing in the working set of decision rules, to select a test with the highest ranking according to a *multicriterion* test selection procedure. The selected test is assigned to the node currently under consideration, initially, the root. The node is expanded by generating branches from it, and assigning to them those values (or groups of values) of the selected test that occur in the working ruleset. The working ruleset is reduced by removing from the rules conditions that are satisfied by the test values on the path from the root to the current node. If the reduced ruleset implies one specific decision, or no more tests can be measured, then the endpoint becomes a leaf, and is assigned that specific decision, or an undetermined decision "?", respectively. Otherwise, the endpoint becomes a node to be expanded in the nest step.

The test selection procedure is based on a combination of elementary criteria, each evaluating one aspect of a test (the way the criteria are combined is explained below). These elementary criteria measure the following aspects:

1) *Cost,* which reflects the difficulty of measuring a test in a given decision-making situation.
2) *Disjointness*, which captures the effectiveness of the test in discriminating among decision rules for different decision classes.
3) *Importance,* which determines a measure of importance of the test in the working set of decision rules.
4) *Value distribution*, which characterizes the distribution of the test importance over all of its of values.
5) *Dominance*, which measures the frequency of the test occurrence in the rules.

These criteria are explained in more detail below.

Cost: The cost of a test is defined by the user. It expresses the total effort (including the measurement cost) needed to measure and apply the test in a given decision-making situation; if the test cannot be measured in a given situation, the cost is infinite.

Disjointness: This criterion measures the degree to which values of a test are different in the rulesets of different classes. Let us suppose that decision classes are $C_1, C_2, \ldots, C_m$, and decision rulesets for these classes are given. For a given test A, let $V_1, V_2, \ldots, V_m$, denote sets of the values (outcomes) of A that are present in the conditions of the rulesets for classes $C_1, C_2, \ldots, C_m$, respectively. If a ruleset for some class, say, C_i contains a rule that does not involve test A, then V_i is assumed to be the domain of A (the set of all possible values of A).

Definition 1. *Class disjointness*, D(A, C_i) *of test A for the ruleset of class* C_i, is the sum of the degrees of disjointness, D(A, C_i, C_j), between the ruleset for C_i and rulesets for C_j, j=1, 2,...m, j ≠ i. The degree of disjointness between a ruleset for C_i and the ruleset for C_j is defined by:

$$D(A, C_i, C_j) = \begin{cases} 0, & \text{if } V_i \subseteq V_j \\ 1, & \text{if } V_i \supset V_j \\ 2, & \text{if } V_i \cap V_j \neq \phi \text{ or } V_i \text{ or } V_j \\ 3, & \text{if } V_i \cap V_j = \phi \end{cases} \qquad (1)$$

where ϕ denotes the empty set.

Definition 2. The *disjointness* of the test A for evaluating a given set of decision rules is the sum of the degrees of class disjointness of each decision class:

$$\text{Disjointness(A)} = \sum_{i=1}^{m} D(A, C_i), \qquad \text{where} \qquad D(A, C_i) = \sum_{i=1,\, i \neq j}^{m} D(A, C_i, C_j) \qquad (2)$$

The disjointness of a test ranges from 0, when the same test values are in the rulesets of all classes, to 3*m*(m-1), when every ruleset of a given class contains a different set of the test values. Selecting a test with the maximum possible disjointness produces a node in the decision structure whose children can be immediately assigned decision classes.

<u>Importance:</u> This criterion is based on the *importance score* (IS), introduced in [Imam et al., 1993]. In the obtained rules, each test is assigned a “score” that represents the total number of training examples that are covered by the rules involving this test. Decision rules learned by an AQ-type learning program are accompanied with information on their strength.

The rule strength is characterized by its *t-weight* and *u-weight.* The *t*-weight (*total-weight*) of a rule for some class is the number of examples of that class covered by the rule. The importance score of a test is the sum of the t-weights of all rules that contain that test in their condition part (u-weights are used for another purpose—see Sec. 3). Suppose that given is a set of decision rules for decision classes $C_1,..,C_m$, and tests $A_1,..,A_n$ involved in these rules. The number of rules associated with class C_i is denoted by "r_i".

*Definition 3***.** The *importance score*, $IS(A_j)$, of the test A_j is determined by:

$$IS(A_j)= \sum_{i=1}^{m} IS(A_j, C_i), \qquad \text{where} \qquad IS(A_j, C_i.) = \sum_{k=1}^{r_i} R_{ik}(A_j) \tag{3}$$

and R_{ik}, the weight of a test A_j in the rule R_k of class C_i is given by:

$$R_{ik}(A_j) = \begin{cases} t-\text{weight} & \text{if } A_j \text{ belongs to rule } R_{ik} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $i=1,..,n$; $i_k=1,..,r_i$; $j=1,..,m$.

<u>Value distribution:</u> This criterion represents the importance score normalized by the size of the test's domain (the number of legal values).

*Definition 4***.** A *value distribution*, $VD(A_j)$ of a test A_i is defined by:

$$VD(A_j) = IS(A_j) / v_j \tag{5}$$

where "v" is the number of legal values of A_j.

Experiments have shown that this criterion is especially useful when *discriminant* decision rules (Michalski, 1983) are used as the source rules.

<u>Dominance</u>: This criterion measures the number of rules involving a test. Because rule conditions may differ in the number of values linked by the internal disjunction to an attribute, to calculate the dominance of a test, the rules are counted as if they were converted (multiplied out) to rules without internal disjunction.

For example, multiplying out the rule [x3=1 v 3] & [x4=1] => C yields two rules [x3=1] & [x4=1] => C and [x3=3] & [x4=1] => C.

A test selection procedure is based on one or more of the above elementary criteria. A user selects elementary criteria that appear to be most relevant to a given application problem, and arranges them into a *lexicographic evaluation functional with tolerances* (LEF) [Michalski, 1973].

LEF consists of a list of <elementary criterion--tolerance> pairs, where tolerance is in %. LEF is used to select the best test in the following way. All available tests are evaluated on the first elementary criterion in the LEF, and those that score within the tolerance range from the best score are selected for an evaluation by the next elementary criterion, etc. This process continues until only one test remains, or the criteria list is exhausted (in which case the test with the highest score on the first criterion is chosen). The default LEF is :

$$<\text{Cost } \tau 1; \text{Disjointness}, \tau 2; \text{Importance}, \tau 3; \text{Value distr.}, \tau 4; \text{Dominance}, \tau 5> \quad (6)$$

where tolerances $\tau 1$, $\tau 2$, $\tau 3$, $\tau 4$ and $\tau 5$ (in percentage) have default value 0, and the default value of the cost of each test is 1. The default tolerance 0% means that only tests receiving the maximum score on a given elementary criterion are passed to the evaluation by the next criterion.

4 Illustrative Applications

This section illustrates the method presented above by applying it to a practical problem. The problem is to determine a decision structure for evaluating the structural quality of tall building designs. The design quality is classified into four classes: *High* (C1), *Medium* (C2), *Low* (C3), and *Infeasible* (C4).

Initial data were in the form of examples of designs classified by an expert into the above classes. Each example was characterized by seven attributes: number of stories (x1), bay length (x2), wind intensity (x3), number of joints (x4), number of bays (x5), number of vertical trusses (x6), and number of horizontal trusses (x7). The data consisted of 335 examples, of which 220 (66%) were randomly selected to serve as training examples, and 115 (34%) were used for testing the obtained decision structures. In the first phase, general decision rules were determined by applying inductive learning program AQ15c (a new version of AQ15 [Michalski, et al. 1986] written in "C" [Wnek et al. 1995]). Figure 1 presents decision rules obtained.

Decision class C1 (High)		
1	[x1=1][x6=1][x4=1v3][x5=1,2][x7=1..3]	(t :18, u :18)
2	[x1=3][x2=1][x3=1][x5=1][x6=1][x4=1v3][x7=1v3v4]	(t :3, u : 3)
3	[x1=5][x2=2][x3=2][x5=2][x4=3][x6=1][x7=2v3]	(t :2, u : 2)
4	[x1=1][x6=1][x2=2][x4=3][x5=1v2][x7=4]	(t :2, u : 2)
5	[x1=3][x2=1][x4=1][x6=1][x7=1][x3=2][x5=1v2]	(t :2, u : 2)
6	[x1=1][x3=1][x6=1][x2=2][x4=1v3][x7=1v3][x5=3]	(t :2, u : 2)
7	[x1=2][x5=2][x2=1][x6=1][x4=3][x7=4]	(t :2, u : 2)
Decision class C2 (Medium)		
1	[x1=2..4][x4=3][x5=2,3][x6=1][x7=2v3]	(t :28, u :19)
2	[x1=2..4][x2=2][x4=3][x5=1,2][x6=1][x7=3,4]	(t :17, u : 6)
3	[x1=2v4][x4=3][x5=1][x6=1][x7=3v4]	(t :10, u : 4)
4	[x1=1v3v5][x4=3][x5=3][x6=1][x7=2v4]	(t :10, u : 2)
5	[x1=3v5][x4=3][x5=2v3][x6=1][x7=1v4]	(t : 9, u : 4)
6	[x1=2][x4=1][x6=1][x7=1]	(t : 7, u : 6)
7	[x1=3v4][x2=2][x3=2][x4=1v3][x5=1v3][x6=1][x7=1v2]	(t : 6, u : 4)
8	[x1=3v5][x2=2][x3=1][x7=1][x4=1v2][x6=1v3]	(t : 5, u : 5)
9	[x1=1][x2=1][x6=1][x4=3][x5=1,2][x7=4]	(t : 4, u : 4)
10	[x1=1][x5=1][x2=2][x4=2][x6=2][x7=1..3]	(t : 4, u : 4)
11	[x1=1v2][x2=1][x6=1][x4=1v3][x5=3][x7=1v4]	(t : 4, u : 2)
Decision class C3 (Low)		
1	[x1=2..5][x4=1v2][x5=1v3][x6=2v4]	(t :41, u :32)
2	[x1=1..4][x4=2][x5=2][x6=2v3][x7=2..4]	(t :27, u :20)
3	[x1=1v3][x2=1][x4=2][x5=1v2][x6=2,3]	(t :19, u : 6)
4	[x1=1v2v4][x4=2][x5=2v3][x6=3v4][x7=1]	(t :13, u : 8)
5	[x1=5][x2=2][x4=2][x5=2][x6=3][x7=2..4]	(t : 5, u : 5)
Decision class C4 (Infeasible)		
1	[x1=5][x2=2][x3=2][x4=1,3][x5=1][x6=1]	(t : 4, u : 4)
2	[x1=5][x2=2][x3=1][x5=1][x6=1][x4=3][x7=3]	(t : 1, u : 1)

Concatenation of conditions means conjunction, t denotes the total weight of a rule, and u denotes the unique weight.

Figure 1: Decision rules learned by AQ15c from the wind bracing data .

As one can see in Figure 1, the first rule in each ruleset (set of rules for one class) is by far the strongest rule for that class. Rules with a very small u-weight (the number of examples covered by a given rule and no previous rule in the set) point out to rare, special cases, or errors in the data. The study presented in (Bergadano et al., 1992) demonstrates that one can obtain a very high predictive accuracy by selecting only the strongest rule from each class (and ignoring the remaining rules), and then using it with a *flexible matching* procedure when determining the class membership of a new example. In this study, to illustrate the methodology, all rules were preserved.

Table 1 presents results (scores) from applying elementary criteria to each attribute (test) in the rules from Figure 1. These scores were used to select the best test for the root of the decision structure to be created from the rules. For each class, the row marked "Values" lists values occurring in the ruleset for this class.

For evaluating the disjointness of an attribute, say **A**, each rule in the ruleset that does not contain **A** is assumed to contain an additional empty condition [**A**= a v b ...k], where a, b, ...k are all the legal values of **A.**

Class		**Attributes**						
		x1	**x2**	**x3**	**x4**	**x5**	**x6**	**x7**
C1	Values	1,2,3,5	1,2	1,2	1,3	1,2,3	1	1..4
	Class disjointness	1	1	0	2	1	3	0
C2	Values	1..5	1,2	1,2	1,2,3	1,2,3	1,2,3	1..4
	Class disjointness	2	1	0	3	1	4	0
C3	Values	1..5	1,2	1,2	1,2	1,2,3	2,3,4	1..4
	Class disjointness	2	1	0	4	1	8	0
C4	Values	5	1	1,2	1,3	1	1	1..4
	Class disjointness	0	0	0	2	0	3	0
Attribute Disjointness		5	3	0	11	3	18	3
Attribute Importance		245	82	25	245	233	245	181
Attribute value distribution		49	41	13	82	78	61	45
Attribute Dominance		45	34	42	33	40	30	54

Table 1: Values of selection criteria for each attribute for the wind bracing problem.

Assuming the default LEF, that is, <Cost 0%; Disjointness, 0%; Importance, 0%; Value distr., 0%; Dominance, 0%>, and the Cost equal 1 for all attributes, attribute x6 is chosen for the root (as it has the highest disjointness). Branches stemming from the root are marked by single values, or groups of values of variable x6, according to the way in which they occur in the decision rules (groups of values subsumed by other groups are removed; Imam and Michalski, 1993). The nodes attached to the branches are assigned rules that satisfy conditions specified by the assigned attribute values. If rules assigned to some node are all of the same class, then this node becomes a leaf, and assigned the name of that class. This process is repeated for each node until all nodes become leaves, or there is no more tests available. Figure 2 presents a decision structure generated by AQDT-2 from decision rules shown in Figure 1 (using the default LEF).

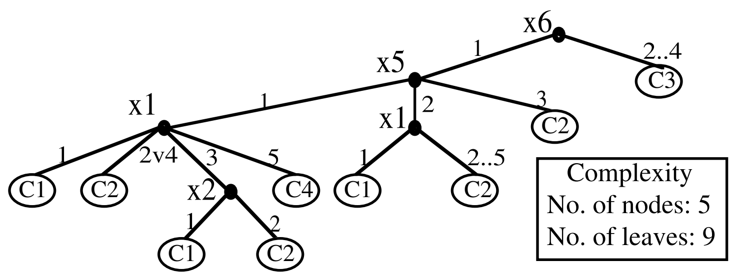


Figure 2: A decision structure determined by AQDT-2 for the wind bracing problem.

The decision structure was tested on testing examples, and the prediction accuracy was *88.7%* (102 testing examples were classified correctly and 13 were misclassified). For comparison, the well known C4.5 program for decision trees was also applied to the same problem (Quinlan, 1990). C4.5 was used with the default window setting (maximum of 20% the number of examples and twice the square root the number of examples), and the number of trials was set to one. The decision tree produced by C4.5 was considerably more complex (it had 17 nodes and 43 leaves, as compared to 5 nodes and 9 leaves in the AQDT decision structure), and its prediction accuracy was lower (84.3%, as compared to 88.7%; 97 testing examples were classified correctly and 18 were misclassified). Thus, AQDT-2 generated decision structures were much simpler than decision trees obtained from C4.5, and their predictive performance was higher.

5 Special Issues

5.1 Handling the Attribute Cost

As described in Section 2, the LEF criterion can take into consideration the cost of measuring attributes (tests). If some attribute has high cost, or is impossible to measure (infinite cost), as indicated in LEF, at each step the system tries to select a "cheaper" attribute among available alternatives.

Figure 3 shows a decision structure obtained from the rules in Figure 1 under the condition that x1 is unavailable (infinite cost). As one can see, although x1 is unavailable, a definite decision still can be assigned in many cases. In some cases, however, a specific decision cannot be reached. Such leaves, called *indecision nodes* , are marked by ?. In Figure 3, indecision nodes are those at which a definite decision cannot be reached without knowing x1.

The decision tree was tested on 115 new examples, of which 71 were classified correctly, 14 incorrectly, and 30 were assigned the "?" decision.

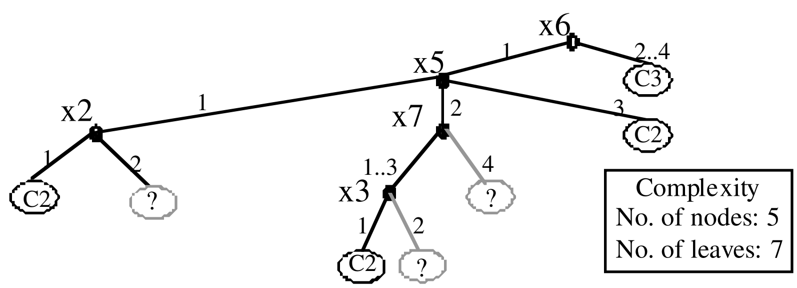


Figure 3: A decision structure learned without x1.

5.2 Assigning Probability Estimates to Alternative Decisions

As illustrated above, when some attribute(s) cannot be measured, the decision structure may not be able to assign a definite decision in some cases. In such cases, if no more information can be obtained, but a decision must be made at an indecision node, it would be useful to know an estimate of the probability distribution of different candidate decisions (Smyth, Goodman & Higgins, 1990). We will present here a method for computing a *data-based* estimate of such probabilities.

This estimate is based on the assumption that the probabilities of alternative decision classes at an indecision node can be estimated by the frequency of training examples of these classes at that node. Let us suppose that we have a complete decision structure in which leaves correspond to specific decisions, but the structure was constructed in such a way that all the attributes from the root to some node, IND, can be measured, and beyond that node cannot be measured. Thus, the node IND is an indecision node. IND can be associated with a set of decision rules of different classes that are satisfied by the values of attributes on the path from the root to that node. These rules can be divided to two parts:

—the *context part*, which is a conjunction of conditions satisfied by the values of attributes along the path from the root to IND (exclusively), and

—the *free part*, which contains one or more rules for each candidate class; each such rule contains attributes assigned to a path from IND (inclusively) to one of the leaf nodes.

Suppose, without loss of generality, that IND can be assigned only class C1 or C2. Let N1 denote the number of known examples of C1 that are covered by the free part of the rules of class C1, and N2 denote the number of known examples of C2 that covered by the free part of the rules of class C2. If the free parts of rules of some class, say Ci, are disjoint, then Ni is simply the sum of t-weights of these rules, otherwise, the sum must be reduced by the number of examples in the intersection. A computationally simpler process is to compute Ni as the sum of *u-weights* (unique-weights) of the rules. The data-based probability estimates p'(C1) and p'(C2) that an event satisfying the context part of N is of class C1 or C2, respectively, is computed as:

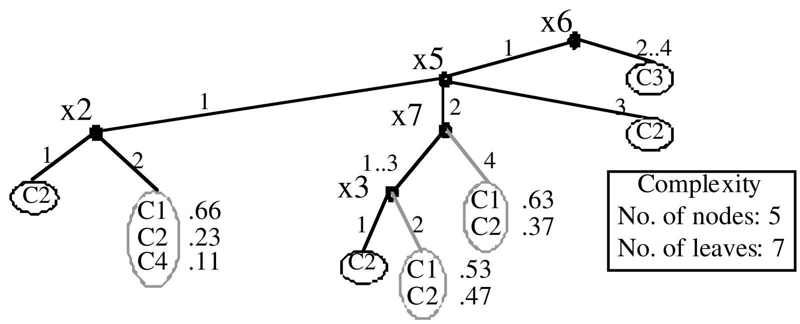
$$p'(C1) = N1/(N1+ N2) \qquad \text{and } p'(C2) = N2/(N1+ N2) \tag{7}$$

If node N is associated with a subset of possible classes, Ci, $i \in I$, then we have:

$$p'(Ci) = Ni/ S, i \in I \tag{8}$$

where S is the total number of (seen) examples that satisfy the context part at IND.

Figure 4 presents a decision structure from Figure 3 in which indecision nodes (leaves marked by ? in Fig. 3) were assigned candidate decisions with data-based probability estimates computed according to (8). Let us consider node x2. The number of examples of decision classes C1, C3, and C4 that satisfy the context part of rules in Fig. 1, were N1=31, N2=11, and N4=5. Thus, the data-based probability estimates according to (8), are p'(C1)= . 66, p'(C2) = .23, and p'(C4) =.11.



Predictive accuracy : 88.7 %.

Figure 4: A decision structure from figure 3 with data-based estimates of class probabilities at indecision nodes.

The data-based estimates p'(Ci), as defined in (8), are computationally simple, but cannot be made if the decision rules have been obtained from an expert, so that one cannot determine examples that satisfy the rules.

Another problem with these estimates is that they do not take into consideration the fact that decision rules associated with IND are typically generalizations of training examples; thus they cover unseen examples. The predictive accuracy of these generalizations, which can be estimated experimentally, influences the value of the probabilities of candidate classes at indecision nodes.

This consideration calls for introducing another measure, p''(Ci), in which the probability of an arbitrary example belonging to class Ci (when it satisfies the context part of node IND), depends on the degree of generalization represented by the rules and their predictive accuracy. If the predictive accuracy of the rules associated with IND is guaranteed to be 100%, such a *generalization-based* probability estimate could be computed as:

$$p''(Ci) = Pi / P, \; i \in I \tag{9}$$

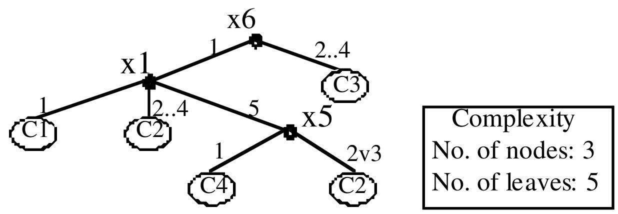
where Pi is the number of all *possible* examples that satisfy the free part of the rules for class Ci, and P is the number of all possible examples that satisfy any of the rules in the free part. The predictive accuracy of decision rules is, however, normally less than 100%; therefore, the formula

(9) is only a rough approximation. A better approximation would be to take into consideration the estimate of predictive accuracy for the portion of the rules that covers the unseen examples of each class. There is also an issue of how to handle examples that are not covered by any rules. We leave the problem of deriving the most adequate generalization-based class probability estimates for future research.

5.3 Simplifying Decision Structures by Rule Truncation

As mentioned earlier, decision rules with a small weight (total and/or unique) may be indicative of errors in the data. Studies presented in (Michalski et al., 1986; Bergadano et al., 1992; Imam and Michalski, 1993) show that such rules can be truncated, and that such a truncation can be useful even when "light" rules are not necessarily covering only noisy examples. Examples that are left uncovered by truncating such rules can be often covered by applying a flexible matching procedure to the remaining decision rules. Flexible matching is done by computing *a degree of match* between an example and rules of candidate classes, and selecting decision corresponding tot he best match (Michalski et al., 1986).

The default setting of AQDT-2 requires pruning decision rules with a support level of 3% or less. The support level is the percentage of the total number of examples covered by a rule (called *the t-weight*) to the total number of examples in the given decision class). Figure 5 shows a decision structure obtained after pruning decision rules (that were used for determining decision structure in Figure 2). In this case, rules with 10% or lower relative t-weight were removed. The produced decision structure has a slightly lower predictive accuracy in comparison with the decision structure in Figure 2 (88% vs. 88.7%); but it has only 3 nodes and 5 leaves vs. 5 nodes and 9 leaves that are in the structure in Figure 2.



Predictive accuracy: 88%.

Figure 5: A simplified decision structure due to the rule truncation for the same problem as decision structure in Figure 2.

This example shows that the rule truncation may significantly simplify a decision structure, without significantly affecting the prediction accuracy. In fact, in some experiments, such a rule truncation not only simplifies decision rules (and thus leads a simpler decision structure), but also *improves* their prediction accuracy (Bergadano et al., 1992).

6. Applying AQDT-2 to Congressional Voting 81 Problem

To test AQDT-2 on a real-world problem, it was applied to learning patterns in the U.S. congressional voting (1981 Congress Voting Record). There are two decision classes: “Democratic Voting Pattern” and “Republican Voting Pattern.” Each voting record of a democrat or a republican in the U.S. congress was described in terms of 19 multivalued attributes. In the experiment described here, 51 voting records were used (31 Democratic and 20 Republican).

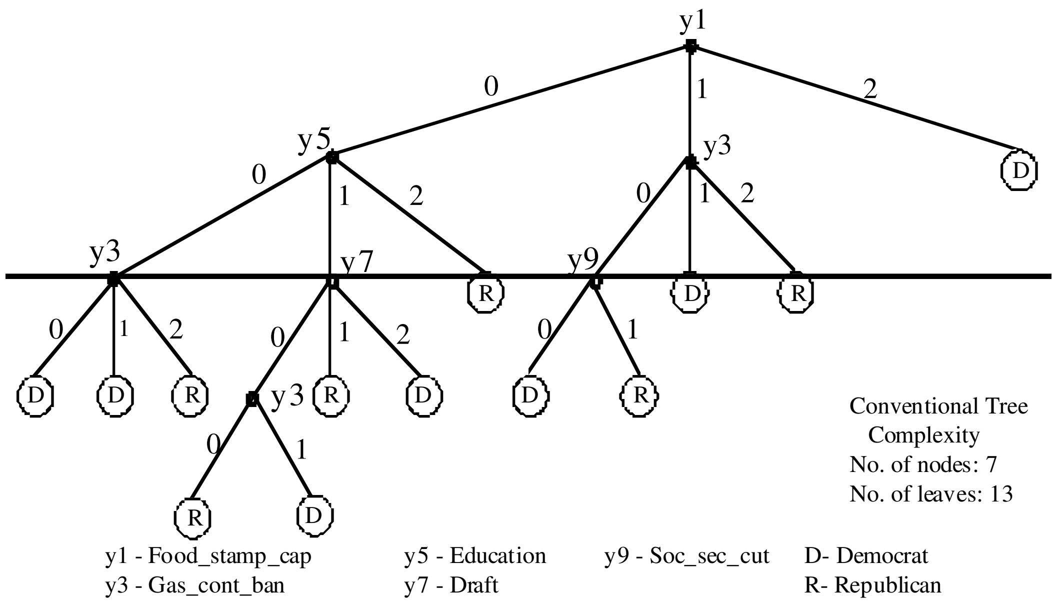
The AQ15 inductive learning program generated four rules for the “Democratic Voting Pattern,” and 7 rules for the “Republican Voting Pattern.” Only 10 of 19 original attributes were used in the obtained rules. Table 2 lists attributes involved in the decision rules, and their legal values (domains). For simplicity, original symbolic values have been mapped into isomorphic numerical values. These numerical values correspond to the symbolic values listed in Table 2. The AQDT-2 program produced a decision tree with 20 nodes. The prediction accuracy of the tree on the testing examples was 91.8%.

For comparison, C4.5, was also run on exactly the same data. It produced a decision tree with 23 nodes, and its prediction accuracy on the same testing examples was 85.7%. Both programs were run under the assumption that they will produce a complete and consistent decision tree with regard to the training examples.

Figures 6 and 7 present decision structures generated by AQDT-2 for the above Congressional Voting-1981 problem. Figure 6 shows a decision structure, generated from AQ15 rules (without constructive induction), and Figure 7 shows a compact decision tree, generated from AQ17-DCI rules (with constructive induction). The compact decision tree contains some nodes that are assigned constructed attributes: y20, y21 and y22. These attributes represent simple mathematical relations on the initial attributes. The attribute y20 is defined by the relation y7+y3=1 v 2 {The attribute takes value T (true) whenever the sum of the numeric values of y7 and y3 equals one or two, and value F (false), otherwise.} Attribute y21 is defined by the relation y12+y9 ≤ 3, and attribute y22 is defined by the relation y12-y4= 0 v 1.

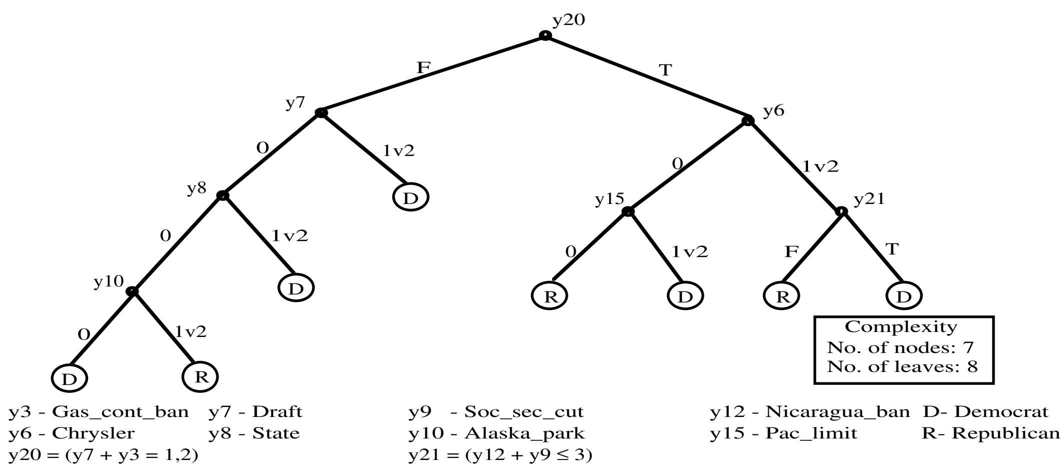
Attribute	Legal Values		
y1- Food_stamp_cap	0 - no	1 - yes	2- not registered
y2- Occupation	0 - known	1 - unknown	
y3- Gas_cont_ban	0 - no	1 - yes	2 - not registered
y4- Income	0 - low	1 - medium	2 - high
y5- Education	0 - no	1 - yes	2 - not registered
y6- Chrysler	0 - no	1 - yes	2 - not registered
y7- Draft	0 - no	1 - yes	2 - not registered
y8- State	0 - northwest	1 - northeast	2 - not registered
y9- Soc_sec_cut	0 - no	1 - yes	2 - not registered
y10- Alaska_parks	0 - no	1 - yes	2 - not registered
y11- Wind_tax_limit	0 - no	1 - yes	2 - not registered
y12- Nicaragua_ban	0 - no	1 - yes	2 - not registered
y13- Fair_housing	0 - no	1 - yes	2 - not registered
y14- Nuc_power	0 - no	1 - yes	2 - not registered
y15- Pac_limit	0 - no	1 - yes	2 - not registered
y16- Mx_cut	0 - no	1 - yes	2 - not registered
y17- Osha_cut	0 - no	1 - yes	2 - not registered
y18- Hosp_cost_cont	0 - no	1 - yes	2 - not registered
y19- Population	0 - small	1 - medium	2 - large

Table 2: The US Congressional voting attributes and their legal value sets.



The predictive accuracy on testing examples: 91.8%.

Figure 6: A decision structure obtained by AQDT-2 for the Congressional Voting-1981 problem without constructive induction.

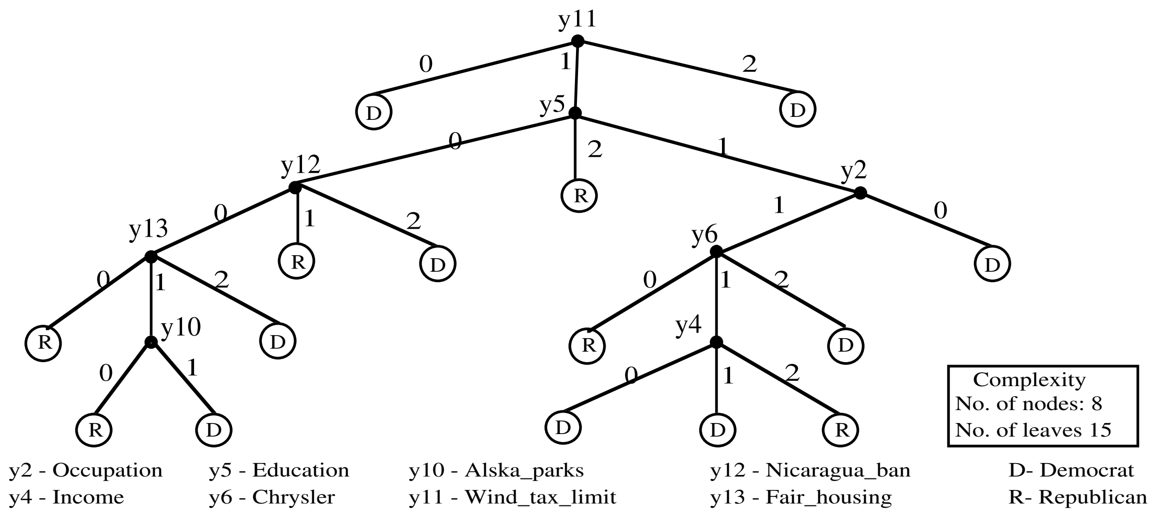


The predictive accuracy on testing examples: 91.8%.

Figure 7: A decision structure obtained by AQDT-2 for the Congressional Voting-1981 problem with constructive induction.

For comparison, Figure 8 presents a decision tree generated by C4.5 for the same problem. Comparing decision trees in Figures 6, 7 and 8, one can notice that decision structures generated from decision rules (conventional and compact) had a considerably higher predictive accuracy

(91.8% vs. 85.7%) on testing examples, as well as were simpler than the tree generated directly from examples (by C4.5).



The predictive accuracy on testing examples: 85.7%.

Figure 8: A decision tree obtained by C4.5 for the Congressional Voting-1981 problem.

7. Conclusion

The presented methodology advocates building decision structures (or decision trees) from decision rules, rather than directly from examples. A justification for such an approach is that decision rules are a form of declarative knowledge representation, and as such permit a greater flexibility of interpreting knowledge they represent. One can also argue that such a process has parallels to human decision making, because people mostly store declarative representations, and from them derive decision procedures they need in any given situation.

From a rule representation, one can potentially derive many equivalent or example-set equivalent decision structures. {*By example-set equivalent* decision structures are meant structures that assign the same decisions to the examples in the training set}. In the presented methodology, the above flexibility is exploited for deriving a decision structure that is most suitable for a given decision-making situation.

The concept of decision structure has been demonstrated to have an advantage over conventional decision trees in terms of the simplicity of representing the same decision function. The methodology has been impelemented in AQDT-2 program. While the methodology is oriented toward creating decision structures from decision rules, it can also be used for creating decision structures from examples (since examples can be viewed as specific rules). In the experiments on applying AQDT-2 to various problems, the obtained decision structures were significantly simpler, sometimes by a wide margin, than decision trees obtained by program C4.5, while their predictive accuracy was comparable or superior. Further research is needed to determine if these findings were only for the problems we worked with, or represent a general pattern. The methodology permits a user to employ several elementary criteria in determining a decision structure that is most suitable for a given decision-making situation. The available elementary criteria include cost, disjointness, importance, value distribution, and dominance.

A major advantage of the methodology is that it allows one to very efficiently create a decision structure from decision rules that is optimized for any given decision-making situation. The time of generating a decision structure from decision rules in the cases we investigated was negligible. Therefore, it is easy to experiment with many different criteria for decision structure generation in order to obtain the most desirable one for a given problem.

ACKNOWLEDGMENTS

The first author had the opportunity of presenting basic ideas contained in this paper at the *7th International Symposium on Methodologies for Intelligent Systems* in Trondheim, Norway in June 1993. Dr. Cecylia Rauszer was in the audience, and after the talk made complimentary comments about the presentation. The memory of these comments will always stay in the heart of the speaker. This was the last time he had the privilege to talk with Inka (as Dr. Rauszer was called by her friends), with whom he has had previously many research discussions, and who was his personal friend.

The authors thank Ken Kaufman and James Mitchell for his comments and criticism of earlier versions of this paper. The material presented here is a substantial improvement and an adaptation of the material presented by the authors at the 8th International Symposium on Methodologies for Intelligent Systems in Charlotte, October 16-19, 1994.

This research was conducted in the Machine Learning and Inference Laboratory at George Mason University. The Laboratory's research activities are supported in part by the National Science Foundation under grants IRI-9510644 and DMI-9496192, in part by the Office of Naval Research under grant N00014-91-J-1351, in part by the Defence Advanced Research Projects Agency under grant No. N00014-91-J-1854 administered by the Office of Naval Research, and in part by the Advanced Research Projects Agency under grants F49620-92-J-0549 and F49620-95-1-0462 administered by the Air Force Office of Scientific Research.

REFERENCES

Bergadano, F., Matwin, S., Michalski R. S. and Zhang, J., "Learning Two-tiered Descriptions of Flexible Concepts: The POSEIDON System," *Machine Learning*, Vol. 8, No. 1, pp. 5-43, January 1992.

Bloedorn, E. and R. S. Michalski, "DATA-DRIVEN CONSTRUCTIVE INDUCTION: A Methodology and Experiments," *Reports of Machine Learning and Inference Laboratory*, George Mason University, 1997 (to appear).

Bloedorn, E., Wnek, J., Michalski, R.S., and Kaufman, K., "AQ17: A Multistrategy Learning System: The Method and User's Guide", Report of Machine Learning and Inference Laboratory, MIL-93-12, George Mason University, 1993.

Bohanec, M. and Bratko, I., "Trading Accuracy for Simplicity in Decision Trees", Machine Learning Journal, Vol. 15, No. 3, Kluwer Academic Publishers, 1994.

Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J., "Classification and Regression Structures", Belmont, California: Wadsworth Int. Group, 1984.

Cestnik, B. & Bratko, I., "On Estimating Probabilities in Structure Pruning", *Proceeding of EWSL 91*, (pp. 138-150) Porto, Portugal, March 6-8, 1991.

Cestnik, B. & Karalic, A., "The Estimation of Probabilities in Attribute Selection Measures for Decision Structure Induction" in *Proceeding of the European Summer School on Machine Learning*, July 22-31, Priory Corsendonk, Belgium, 1991.

Imam, I.F. and Michalski, R.S., "Learning Decision Structures from Decision Rules: A method and initial results from a comparative study", in *Journal of Intelligent Information Systems JIIS,* Vol. 2, No. 3, pp. 279-304, Kerschberg, L., Ras, Z., & Zemankova, M. (Eds.), Kluwer Academic Pub., MA, 1993.

Imam, I.F., Michalski, R.S. and Kerschberg, L., "Discovering Attribute Dependence in Databases by Integrating Symbolic Learning and Statistical Analysis Techniques", *Proceeding of the First International Workshop on Knowledge Discovery in Database*, Washington, D.C., July, 11-12, 1993.

Michalski, R.S., "AQVAL/1-Computer Implementation of a Variable-Valued Logic System VL1 and Examples of Its Application to Pattern Recognition," *Proceeding of the First International Joint Conference on Pattern Recognition*, (pp. 3-17), Washington, DC, October 30- November 1, 1973.

Michalski, R.S., "A Theory and Methodology of Inductive Learning," Chapter in *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. Carbonell and T. Mitchell (Eds.), TIOGA Publishing Co., Palo Alto, pp. 83-134, 1983.

Michalski, R.S, "Designing Extended Entry Decision Tables and Optimal Decision Trees Using Decision Diagrams", *Technical Report No. 898*, Urbana: University of Illinois, March 1978.

Michalski, R.S., Mozetic, I., Hong, J. & Lavrac, N., "The Multi-Purpose Incremental Learning System AQ15 and its Testing Application to Three Dedical Domains", *Proceedings of AAAI-86*, (pp. 1041-1045), Philadelphia, PA, 1986.

Mingers, J., "An Empirical Comparison of Selection Measures for Decision-Structure Induction", *Machine Learning*, Vol. 3, No. 3, (pp. 319-342), Kluwer Academic Publishers, 1989a.

Quinlan, J.R., "Discovering Rules by Induction from Large Collections of Examples", in D. Michie (Edr), *Expert Systems in the Microelectronic Age*, Edinburgh University Press, 1979.

Quinlan, J.R., "Learning Efficient Classification Procedures and Their Application To Chess End Games," in R.S. Michalski, J.G. Carbonell and T.M. Mitchell, (Eds.), *Machine Learning: An Artificial Intelligence Approach*. Los Altos: Morgan Kaufmann, 1983.

Quinlan, J. R. "Probabilistic Decision Structures," in Y. Kodratoff and R.S. Michalski (Eds.), *Machine Learning: An Artificial Intelligence Approach*, *Vol. III*, San Mateo, CA, Morgan Kaufmann Publishers, (pp. 63-111), June, 1990.

Smyth, P., Goodman, R.M., and Higgins, C., "A Hybrid Rule-based/Bayesian Classifier", Proceedings of ECAI-90, Stockholm, 1990.