SHOULD DECISION TREES BE LEARNED FROM EXAMPLES OR FROM DECISION RULES?

by

I. F. Imam
R. S. Michalski

In Reports of Machine Learning and Inference Laboratory, Center for Artificial Intelligence, George Mason University, May, 1993. and Lecture Notes in Artificial Intelligence, Springer Verlag, Proceedings of the First International Workshop on Knowledge Discovery in Databases, Washington, D.C., July 11-12, 1993.

Should Decision Trees be Learned from Examples or from Decision Rules?

Ibrahim F. Imam and Ryszard S. Michalski

Center for Artificial Intelligence
George Mason University
Fairfax, VA. 22030
iimam@aic.gmu.edu & michalski@aic.gmu.edu

Published in
Lecture Notes in Artificial Intelligence, Spring Verlag, the Proceeding of the 7th International Symposium on Methodologies for Intelligent Systems, ISMIS, Trondheim, Norway, June 15-18, 1993

10 pages, almost 5x6 inches. six figures.
This paper invited for submission to the journal of Intelligent Information Systems (JIIS).

Should Decision Trees be Learned from Examples or from Decision Rules?

Ibrahim F. Imam and Ryszard S. Michalski

Center for Artificial Intelligence
George Mason University
iimam@aic.gmu.edu & michalski@aic.gmu.edu

ABSTRACT

A standard method for determining decision trees is to learn them from examples. A disadvantage of this approach is that once a decision tree is learned, it is difficult to modify it to suit different decision making situations. An attractive approach that avoids this problem is to learn and store knowledge in a declarative form, e.g., as decision rules, and then, whenever needed, generate from it a decision tree that is most suitable in any given situation. This paper describes an efficient method for this purpose, called AQDT-1, which takes decision rules generated by the learning system AQ15 and builds from them a decision tree optimized according to a given quality criterion. The method is able to build conventional decision trees, as well as the so-called "skip noder" trees, in which measuring attributes assigned to some nodes may be avoided. It is shown that "skip-node" trees can be significantly simpler than conventional ones. In the experiments comparing AQDT-1 with C4.5, the former outperformed the latter both in terms of the predictive accuracy as well as the simplicity of the generated decision trees.

Key words: machine learning, inductive learning, decision trees, decision rules.

1. Introduction

Methods for learning decision trees from examples have been quite popular in machine learning due to their simplicity. Decision trees built this way can be quite efficient, as long as the decision making situations for which they are optimized remain relatively stable. Problems arise when these situations change, or the assumptions under which the tree was built do not hold. For example, in some situations it may be very difficult to determine the value of a certain attribute on the path from the root. One would like to avoid measuring this attribute, and still be able to classify the example. If the cost of measuring of various attributes changes, it is desirable to restructure the tree so that the "inexpensive" attributes are evaluated first. A restructuring is also desirable if there is a significant variation in the frequency of occurrence of examples from different classes. Restructuring a decision tree is, however, difficult because the tree represents a form of procedural knowledge.

An attractive alternative that avoids the above problems is to learn and store knowledge in the form of decision rules, and to generate from them an appropriate decision tree "dynamically," as needed. Decision rules represent knowledge declaratively, and thus do not impose any order on the evaluation of the attributes. Since the number of rules is typically much smaller than the number of examples, generating decision trees from

rules can potentially be very fast. This way, one can always generate a tree that is tailored to the specific decision situation. For example, one may be able to generate a decision tree that avoids evaluating an attribute that is difficult or impossible to measure, or a decision tree that fits well some particular distribution of the decision classes. In some situations, it may be unnecessary to generate a complete decision tree, but instead only the part with the leaves associated with the decision classes of interest. The proposed approach would generate only the desirable part. This could be interpreted as a generation of "virtual" decision trees.

A disadvantage of this approach is that it requires determining decision rules first. There are, however, very efficient methods for generating decision rules. Also, the rules need to be generated only once, and then can be used many times for generating any type of decision tree according to various decision making situation.

This paper presents a simple and efficient method for generating decision trees from decision rules. The method employs the AQ algorithm for generating rules. It also reports results from experiments comparing it with a well-known method for learning decision trees from examples, implemented in the C4.5 program.

2. Generating Decision Trees from Decision Rules: AQDT-1

Decision trees are normally generated from examples of decisions. The essential aspect of any method for this purpose is the *attribute selection criterion* that is used for choosing attributes to be assigned to the nodes of the tree. Among well-known criteria are the entropy reduction measure (e.g., Quinlan, 1979), the gini index of diversity (Breiman, et al., 1984), and others (Cestnik & Karalic, 1991; Mingers, 1989).

The first algorithm for generating decision trees from examples was proposed by Hunt, Marin and Stone (1966). This algorithm was subsequently modified by Quinlan (1979, 1983), and then improved and/or applied by many authors to a variety of learning problems (e.g., Quinlan, 1983; Breiman, et al., 1984). Later, Quinlan (1986), and Bratko and Kononenko (1987) added "tree pruning" procedures to handle data with noise.

In contrast to the above, the proposed method generates decision trees from decision rules. The method, called AQDT-1 (**AQ**-based **D**ecision **T**rees), assumes that decision rules are generated from examples by the inductive learning system AQ15 (Michalski et al., 1986). AQDT-1 uses an attribute selection criterion that is based on the properties of *the rules*, rather than the properties of the *training examples*. One rule may describe a large number of examples. Decision rules are more powerful knowledge representation than decision trees, because they can directly represent any description in disjunctive normal form, while the latter can represent directly only a disjunctive normal form in which all conjunctions are mutually disjoint. Therefore, when transforming a set of arbitrary decision rules into a decision tree, one faces an additional problem of handling logically intersecting rules (conjunctions).

The proposed method for solving the first problem, i.e., choosing attributes on the basis of the properties of rules, employs a measure of "utility" of an attribute for reducing a given set of rules and some other criteria. The second problem (handling non disjoint rules) can be resolved by introducing additional nodes in the decision tree, or by assigning a "Don't care" value to some branches. This value serves as a connection edge between subtrees corresponding to non-disjoint rules. The branch with such a "value" is traversed when one does not know, or wants to ignore the value of the attribute assigned to the node from which it stems. Decision trees that have nodes with "don't care" values are called, for short, "skip-node" trees.

The input to the program AQDT-1 consists of rules generated by AQ15. Each such rule represents a conjunction of conditions. AQDT-1 creates a data structure for each concept description (a set of rules). This structure has fields such as the number of rules, the number of conditions in each rule, and the number of attributes in the rules. The system also creates an array of attribute descriptions. Each attribute description contains the attribute's name, domain, type, the number of legal values, a list of the values, the number of rules that contain that attribute, and values of that attribute for each rule. The attributes are arranged in the array in the a lexicographic order, first in the descending order of the number of rules that contain that attribute, and second, in the ascending order of the number of the attribute's legal values. AQDT-1 constructs a decision tree from decision rules by recursively selecting the "best" attribute at a given step, and assigning it to the new node. The difference between building a decision tree from rules and building it from examples is that in the former, attributes are selected on the basis of the role the attributes play in the rules, rather than on the basis of the coverage of examples, as in learning from examples.

The criterion for attribute selection should reflect the decision making situation. For example, if measuring different attributes involves different costs, then these costs should be included in the criterion. If some decision classes occur much more frequently than others, the criterion should favor measuring the attributes that occur in the rules for these classes. To be able to compare the AQDT-1 method with the C4.5 program, we assume here a selection criterion that is oriented toward producing trees with the minimum number of nodes. This criterion is composed of three elementary criteria: 1) the total *attribute utility*; 2) the number of different attribute values in the rules; and 3) the number of rules that contain the given attribute. The total attribute utility is the sum of the *class utilities*—the utilities of the attribute for each decision class. Suppose that decision classes are C1, C2,..., Cm. Suppose further that V1, V2,....., Vn are sets of values of some attribute A that occurr in rules for classes C1, C2, ...Cm, respectively. The utility of A for Class Ci, U(A, Ci), is the number of sets Vj, (j=1,...,m, and $j \neq i$) that are disjoint with Vi. plus 1. The total utility, U(A), of the attribute A is:

$$U(A) = \sum_{i=1}^{m} U(A, Ci) \qquad (1)$$

The total utility of an attribute is the highest (m^2) when the attribute occurs in every class description, and has a different value in each of them.

The second criterion prefers attributes that have fewer values in the rules, because nodes that are assigned such attributes will have a smaller fan out (in the case of continuous attributes, they are quantized, and their new "values" are ranges of original values). The third criterion prefers the attribute that occurs in a larger number of rules, because this can help to evaluate all the rules faster.

These three criteria are combined into one attribute ranking measure using the "lexicographical evaluation functional with tolerances" (LEF; Michalski, 1973). First, the method evaluates attributes on the basis of their utility. If two or more attributes share the same top score or their scores are within the assumed *tolerance range*, the method evaluates these attributes using the second criterion (other attributes are ignored). If again two or more attributes share the same top score, or their scores are within the tolerance range, then the third criterion is used. If there is still a tie, the method selects the "best" attribute randomly.

The second problem of forming decision trees from decision rules is how to start from a single root and represent all the rules in a decision tree, even in cases when some rules do not contain common attributes. One way of overcoming with this problem is to create additional nodes corresponding to "missing" attributes. This method can significantly increase the size of the tree. Another approach, adopted in AQDT-1, is to create a "Don't care" value for attributes that may not be necessary to evaluate.

The following simple example illustrates the AQDT-1 method. Suppose there are three decision classes, C1,C2 & C3, described by the following AQ15-derived DNF expressions:

C1	<=	[x1=3]&[x2=2]	v	[x1=3] &[x3=1 v 3] &[x4=1]
C2	<=	[x1=1 v 2]&[x2=3 v 4]	v	[x1=2] &[x3=1 v 2] &[x4=2]
C3	<=	[x1=1]&[x2=1]	v	[x1=4] &[x3=2 v 3] &[x4=3]

The method turns such descriptions into elementary rules, which have only one attribute value in each condition (no internal disjunction). This is done by "multiplying out" each conjunction in the DNF expressions.

Figure 1 illustrates the process of selecting an attribute for a node in the tree based on the set of elementary rules. The rows "Values in Ci" list values of the attribute in the rules of the decision class Ci. "Value sharing" indicates whether the value of that attribute in the rule of a given class is or is not present in rules of other classes. Attributes x2 and x4 both have total utilities of 9. Therefore, they are evaluated on the second criterion (the number of attribute values in rules). Attribute x4 has fewer values (3) than x2 (4), therefore it is assigned to the root of the tree. In the next step, AQDT-1 modifies its data structure to eliminate the rules containing x4 from all the concept descriptions, and marks x4 as an "in_tree" attribute. This process is repeated until all

rules are eliminated. In this example, there will be two iterations. The second attribute to be chosen is x2 and a don't care value will be added to x4 to merge the tree.

Concept (# of rules)	Attributes				
		x1	x2	x3	x4
C1 (3 rules)	Values in C1	3	2	1, 3	1
	Val. sharing	No	No	Yes	No
	Class utility	3	3	1	3
C2 (6 rules)	Values in C2	1, 2	3 v 4	1 v 2	2
	Val. sharing	Yes	No	Yes	No
	Class utility	2	3	1	3
C3 (3 rules)	Values in C3	1,4	1	2 v 3	3
	Val. sharing	Yes	No	Yes	No
	Class utility	2	3	1	3
Total Attribute Utility		7	9	3	9

Figure 1: Determining the total autility of the attributes.

3. Analyzing Decision Trees Obtained by AQDT-1

The performance of the AQDT-1 method was evaluated by applying it to several learning problems. The most complex problem was to learn a decision rules for characterizing the voting records in the US Congress. Each voting record was described in terms of 19 multivalued attributes. AQDT-1, when run in the conventional mode mode (no "skip-nodes") and with the attribute selection criterion minimizing the number of nodes, produced a decision tree with 20 nodes, and 91.8% predictive accuracy on the testing examples. For comparison, the well-known decision tree learning program, C4.5, was also run on the exactly the same data. C4.5 produced a decision tree with 23 nodes, and 85.7% predictive accuracy. (Both programs were assumed to produce a complete and consistent decision tree with regard to the training examples, i.e., a decision tree that gives 100% correct recognition of the training examples). AQDT-1 was also run in the "skip-node" mode. As shown in Figure 2, the obtained "skip-node" decision tree has only 13 nodes. If the value of "Food_stamp_cap" is 0 or 1 then one can make the decision "Democrat" or "Republican", respectively. The branch "Don't care" stemming from "Food_stamp_cap" allows one to proceed to the next node (i.e., evaluate the next attribute "Occupation") without knowing the value of "Food_stamp_cap".

The idea of "Don't care" branches (or "skip nodes") allows one to build a decision tree from rules that do not intersect logically. Introducing such "Don't care" branches not only makes the resulting "skip-node" decision tree simpler, but also may allow one to reach a decision when values of some attributes are unknown. This way, a "skip-node"

decision tree makes possible in some situations to avoid measuring an attribute when it is not logically necessary, which is a problem with conventional decision trees (Michalski, 1990).

Let us define the *accuracy* of a decision tree against a set of examples as the percentage of examples that are correctly classified by the tree out of the total number of examples. We will distinguish between two types of accuracy: the *level* accuracy and the *accumulative* accuracy. The level accuracy of a leaf node is the percentage of the correctly classified examples at this node. The level accuracy at a non-leaf node is the percentage of the number of examples that are not classified to any class at that node. The accumulative accuracy of a leaf is the *total* percentage of the correctly classified examples at this point, including those classified correctly at higher nodes.

In Figure 2, L denotes the level accuracy, and A denotes the accumulative accuracy at a given node. For example, the leaf node "Republican" at the first level of the tree has level accuracy of 13/20 (65%). The node "Republican" at subsequent levels has the level accuracy 15/20 (75%), 19/20 (95%), and 11/20 (55%), respectively. In contrast, the accumulative accuracy of the "Republican" nodes at different levels is 13/20 (65%), 18/20 (90%), 20/20 (100%), and 20/20 (100%). The accumulative accuracy at the second level was computed by adding the 13 examples classified correctly at level one, and 5 examples classified correctly, out of the 13 unclassified examples at the first level.

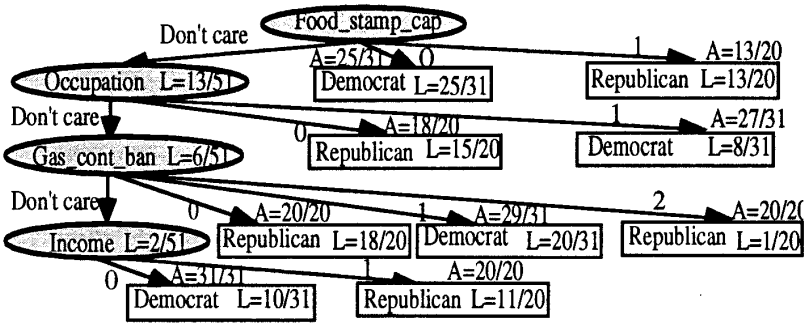

Figure 2: AQDT-1 derived "skip-node" decision tree for the US Congress Voting data.

Let us explain how the AQDT-1-generated tree classifies testing examples. First, it creates an initial hypothetical conclusion by matching the first subtree, which consists of the root and its leaves (in fig. 2), then, it confirms or contradicts this hypothesis through the other subtrees (through Don't care nodes). AQDT-1 uses the level accuracy to choose a conclusion in cases where there is a contradiction with the initial hypothesis. The testing example (Food_stamp_cap=0, Occupation=0, Gas_cont_ban=0, income=2) or (0,0,0,2) will assign an initial conclusion "Democrat" with accuracy 80.6%. During the confirmation process, we notice a contradiction from the second and third subtrees.

The percentages for these subtrees are 75*13/51=19.1% and 95*6/51= 11.1%, respectively. In the last path, there is no value 2, we consider such case against the initial conclusion with percentage (55*2/51=2.2%). If the subtraction (80.6-32.4=48.2%) is less than 50% (in case of two classes), the final conclusion is the contradictory one. Other wise, and also in cases where there are some subtrees that support the initial conclusion, the following method is used. Assume the example, (0,0,1,1) where the third subtree only supports the initial conclusion. Based on the information in Figure 2, we assume that the level accuracy of each contradictory path is 100%, and we consider the contradictory conclusion as initial one, then compare the contradictory percentage for each conclusion. In our example, consider the level accuracy of "Occupation" as 100%. At that level, there are 15 Republican and 8 Democrat examples are classified correctly. There are 28 (28*100/51=54.9%) unclassified examples. A 54.9% will be the level accuracy of "Food_stamp_cap", and (54.9*25/31=44.7%) is the contradictory percentage to the class Republican. That means if the correct concept is Republican, the contradictory percentage is 44.7% from the first subtree, but if it is Democrat, the contradictory percentage is 19.1% from the second subtree. In this example, the contradictory percentage of the "Income" subtree is also less than 44.7%.

4. Decision Tree Pruning

When input data may have errors (noise), it is often useful to prune the obtained decision tree. Such pruning protects the tree from overfilling. Various approaches have been described in (Mingers, 1989; Breiman, et al, 1984; Quinlan, 1986, 1987; Niblett & Bratko, 1986; Cestnik et al, 1987; Clark & Niblett, 1987; Smyth et al, 1990; Cestnik & Bratko, 1991). These pruning approaches differ in their use of various criteria to decide whether or not to prune at a certain stage.

In this paper, we will study the meaning and effect of pruning on a decision tree learned from rules. Pruning the decision tree will be simpler in terms of where to prune because of the structure of the learned decision tree. The meaning of removing one node of the decision tree is equivalent to removing all the alternative conjunctions (conjunctions which contain the attribute and its value) from the rules. As can be seen, the meaning of pruning here is very crucial because of our assumption that the rule-base is complete (AQ15 guarantees 100% match with the training examples). Our pruning approach takes into consideration that the decision tree is learned from rules, not from examples. The pruning strategy is based on pruning whole nodes at the lowest level with their Don't care node. Figure 3 shows the learned decision tree from the Congressional Voting rules. The pruning takes place at the dotted lines, and the numbers on the left represent the accuracy after exchanging the Don't care node with the associated decision.

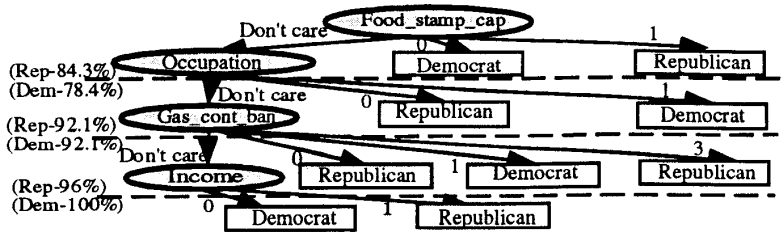

Figure 3: A "skip-node" decision tree and various pruning points at the dotted lines.

5. Comparing Decision Trees from AQDT-1 and C4.5

This section presents an experiment comparing pruned decision trees learned from rules by AQDT-1 with those learned from examples by C4.5. Both programs were applied to the same data on US. congressional voting, described above. The results presented here (for all experiments) are the best results from running C4.5, with both default windows (10 trials; the max. of 20% and twice the square root of the number of examples) and 100% windows (10 trials). C4.5 created a tree with 23 nodes before pruning. After pruning, the tree had 7 nodes with error rate 11.8%.

Figure 4 shows a comparison between the accuracy of the C4.5 and AQDT-1 decision trees with different degrees of pruning. The comparison is done on the training examples (fig. 4a) and testing examples (fig. 4b) examples. From Figure 4, we can see that trees that were learned from rules match the examples better than those trees that were learned from examples with different degrees of pruning.

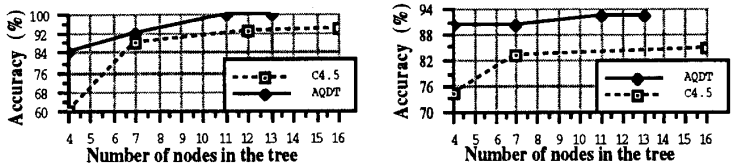

Figure 4: The increase of the accuracy of AQDT-1 and C4.5 generated trees with the number of nodes.

The rest of this section describes two experiments comparing AQDT-1 with C4.5. The experiments were done to show the effect of the size of source data on AQDT-1 and C4.5 in terms of number of nodes and accuracy. In this experiment, the accuracy is calculated against the testing examples. The first experiment involved the second congressional voting data set, which consisted of 112 examples, two concepts to be learned, and 102 testing examples. Figure 5 shows the dependency of accuracy and number of nodes on a set of different relative size of the training examples. The second experiments used the so-called MONK1 data, consists of 124 training examples, two

concepts, and 432 testing examples. Figure 6 shows the dependence of the accuracy and the number of nodes on a set of different relative size of the training examples.

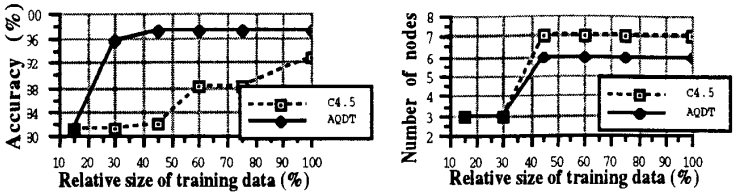

Figure 5 : Comparing decision trees for the US Congressional Voting problem.

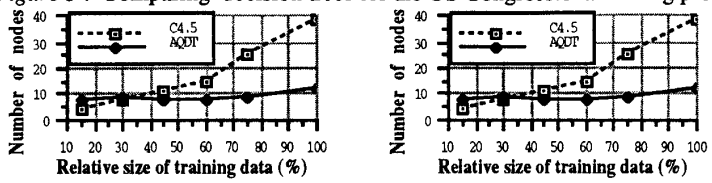

Figure 6: Comparing decision trees for the MONK1 problem.

Figures 5 & 6 show that AQDT-1 produced decision trees with fewer nodes and a higher accuracy than C4.5.

6. Conclusion

The paper presented the AQDT-1 method for efficiently determining decision trees from decision rules. The main difference between determining trees from decision rules and determining them from examples is in the attribute selection function. In the former case, the attribute selection criterion evaluates the role of attributes in the rules, while in the latter it evaluates the coverage of traninig examples. The primary property used in the proposed method is the "total utility" of an attribute for reducing the decision rules.

The method employs the AQ15 inductive learning program for learning decision rules from examples. A major advantage of the proposed approach is that it assists in determining decision trees suitable for different decision making situations, for example, when one wants to avoid measuring some "expensive" attribute. Another advantage is that the method can build decision trees with "Don't care values" on some branches. It has been demonstrated that decision trees with such "Don't care values" can be significantly simpler than conventional decision trees. In the experiments, the AQDT-1 method outperformed the C4.5 method for learning decision trees from examples, both in terms of the predictive accuracy and the simplicity of the generated decision trees.

ACKNOWLEDGMENTS

The authors thank M. Hieb, K. Kaufman, J. Wnek, M. Maloof, H. Vafaie and E. Bloedorn for valuable comments and suggestions.

This research was supported in part by the National Science Foundation under grant No. IRI-9020266, in part by the Defense Advanced Research Projects Agency under the grant No. N00014-91-J-1854, administered by the Office of Naval Research, and the grant No. F49620-92-J-0549, administered by the Air Force Office of Scientific Research, and in part by the Office of Naval Research grant No. N00014-91-J-1351.

REFERENCES

Bratko, I. & Lavrac, N. (Eds.), *Progress in Machine Learning,* Sigma Wilmslow, England, Press, 1987.

Bratko, I. & Kononenko, L. "Learning Diagnostic Rules from Incomplete and Noisy Data," *Interactions in AI and statistics,* B. Phelps, (edt.), Gower Technical Press, 1987

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

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

Clark, P. & Niblett, T. "Induction in Noisy Domains," *Progress in Machine Learning,* I. Bratko and N. Lavrac, (Eds.), Sigma Press, Wilmslow, 1987.

Hunt, E., Marin, J. & Stone, P., *Experiments in induction,* NY: Academic Press, 1966.

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, 1973.

Michalski, R.S., Mozetic, I., Hong, J. & Lavrac, N., The Multi-Purpose Incremental Learning System AQ15 and Its Testing Application to Three Medical Domains,"*Proceedings of AAAI-86,* Philadelphia, PA, 1986.

Michalski, R.S., "Learning Flexible Concepts: Fundamental Ideas and a Method Based on Two-tiered Representation,"*Machine Learning: An Artificial Intelligence Approach, Vol. III,* Y.Kodratoff & R.S.Michalski (Eds.), Morgan Kaufmann, pp. 63-111, 1990.

Mingers, J., "An Empirical Comparison of selection Measures for Decision-Tree Induction," *Machine Learning,* pp.319-342, Vol. 3, No.4, Kluwer Academic Pub., 1989.

Niblett, T. & Bratko, I., "Learning Decision Rules in Noisy Domains," *Proceeding Expert Systems 86,* Brighton, Cambridge: Cambridge University Press, 1986.

Quinlan, J.R., "Discovering Rules By Induction from Large Collections of Examples," *Expert Systems in the Microelectronic Age,* Ed. D. Michie, Edinburgh Unv. Press, 1979.

Quinlan, J.R., "Learning Efficient Classification Procedures and Their Application to Chess End Games," 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., "Induction of Decision Trees,"*Machine Learning* Vol. 1, No. 1, Kluwer Academic Publishers, 1986.

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