A New Method for Learning Decision Trees from Rules

Amany Abdelhalim, Issa Traore
Department of Electrical and Computer Engineering
University of Victoria, P.O.Box 3055 STN CSC,
Victoria, B.C., V8W 3P6, Canada,
Phone: (250) 721-6036, Fax: (250) 721-6052
E-mail: {amany, itraore}@ece.uvic.ca

***Abstract*—Most of the methods that generate decision trees use examples of data instances in the decision tree generation process. This paper proposes a method called "RBDT-1"- rule based decision tree -for learning a decision tree from a set of decision rules that cover the data instances rather than from the data instances themselves. RBDT-1 method uses a set of declarative rules as an input for generating a decision tree. The method's goal is to create on-demand a short and accurate decision tree from a stable or dynamically changing set of rules. We conduct a comparative study of RBDT-1 with three existing decision tree methods based on different problems. The outcome of the study shows that RBDT-1 performs better than AQDT-1 and AQDT-2 which are methods that create decision trees from rules and than ID3 which generates decision trees from data examples, in terms of tree complexity (number of nodes and leaves in the decision tree.**

Key Words— attribute selection criteria, decision rules, data-based decision tree, rule-based decision tree, tree complexity.

I. INTRODUCTION

Decision Trees are one of the most popular classification algorithms used in data mining and machine learning to create knowledge structures that guide the decision making process. The creation of a good knowledge structure is the main step in the development of a decision making system.

The most common methods for creating decision trees are those that create decision trees from a set of examples (data records). We refer to these methods as data-based decision tree methods.

On the other hand, to our knowledge there are only few approaches that create decision trees from rules proposed in the literature which we refer to as rule-based decision tree methods.

There is a major difference between building a decision tree from examples and building it from rules. When building a decision tree from rules the method assigns attributes to the nodes using criteria based on the properties of the attributes in the decision rules, rather than statistics regarding their coverage of the data examples [1].

A decision tree can be an effective tool for guiding a decision process as long as no changes occur in the dataset used to create the decision tree. Thus, for the data-based decision tree methods once there is a significant change in the data, restructuring the decision tree becomes a desirable task. However, it is difficult to manipulate or restructure decision trees. This is because a decision tree is a procedural knowledge representation, which imposes an evaluation order on the attributes. In contrast, rule-based decision tree methods handle manipulations in the data through the rules induced from the data not the decision tree itself. A declarative representation, such as a set of decision rules is much easier to modify and adapt to different situations than a procedural one. This easiness is due to the absence of constraints on the order of evaluating the rules [2].

On the other hand, in order to be able to make a decision for some situation we need to decide the order in which tests should be evaluated in those rules. In that case a decision structure (e.g. decision tree) will be created from the rules.

So, rule-based decision tree methods combine the best of both worlds. On one hand they easily allow changes to the data (when needed) by modifying the rules rather than the decision tree itself. On the other hand they take advantage of the structure of the decision tree to organize the rules in a concise and efficient way required to take the best decision. So knowledge can be stored in a declarative rule form and then be transformed (on the fly) into a decision tree only when needed for a decision making situation [2].

This paper presents a new rule-based decision tree method called *RBDT-1*. To generate a decision tree, the *RBDT-1* method uses in sequence three different criteria to determine the fit (best) attribute for each node of the tree, referred to as *the attribute effectiveness (AE)*, *the attribute autonomy (AA)*, and *the minimum value distribution (MVD)*. In this paper, the *RBDT-1* method is compared to the *AQDT-1* and *AQDT-2* methods which are rule-based decision tree methods, along with the *ID3* which is one of the most famous data-based decision tree methods. The attribute selection criteria in *RBDT-1* method give better results than the other methods' criteria in terms of tree

complexity as we show empirically using several publicly available datasets.

The rest of the paper is structured as follows. Section 2 summarizes the related work. Section 3 presents the rule notation and the rule generation method used. Section 4 presents the *RBDT-1* method. Section 5 presents the decision tree building process. Section 6 presents the results of an experiment in which, based on public datasets, the proposed method is compared to two existing methods for creating decision trees from declarative rules, namely the *AQDT-1* and *AQDT-2* methods, and to the *ID3* algorithm that creates decision trees from data examples. In Section 7, we make some concluding remarks and outline our future work.

II. Related work

There are few published works on creating decision structures from declarative rules.

The *AQDT-1* method introduced in [2] is the first approach proposed in the literature to create a decision tree from decision rules. The *AQDT-1* method uses four criteria for selecting the fit attribute that will be placed at each node of the tree. Those criteria are the *cost*[1], the *disjointness*, the *dominance*, and the *extent*, which are applied in the same specified order in the method's default setting.

The *AQDT-2* method introduced in [1] is a variant of *AQDT-1*. *AQDT-2* uses five criteria in selecting the fit attribute for each node of the tree. Those criteria are the *cost*[1], *disjointness, information importance, value distribution,* and *dominance*, which are applied in the same specified order in the method's default setting. In both the *AQDT-1 & 2* methods, the order of each criterion expresses its level of importance in deciding which attribute will be selected for a node in the decision tree. Although both *AQDT-1 & 2* are capable of generating a decision tree from a set of rules, experiments presented in this paper show that our proposed method *RBDT-1* produces a less complex tree in most of the cases. Another point is that the calculation of the second criterion - the *information importance* - in *AQDT-2* method depends on the training examples as well as the rules, which contradicts the method's fundamental idea of being a rule-based decision tree method. *AQDT-2* requires both the examples and the rules to calculate the *information importance* at certain nodes where the first criterion- *Disjointness* - is not enough in choosing the fit attribute. Thus, without the examples, *AQDT-2* might not be able to create the decision tree. *AQDT-2* being both dependent on the examples as well as the rules increases the running time of the algorithm remarkably in large datasets especially those with large number of attributes.

In contrast the *RBDT-1* method, proposed in this work, depends only on the rules induced from the examples in the calculations of the method's criteria.

Akiba et al. [5] proposed a rule-based decision tree method for learning a single decision tree that approximates the classification decision of a majority voting classifier. Their method was proposed as a possible solution to solve the issues of intelligibility, classification speed, and required space in majority voting classifiers. In their proposed method, if-then rules are generated from each classifier (a *C4.5* based decision tree) and then a single decision tree is learned from these rules. Since the final learning result is represented as a single decision tree, problems of intelligibility and classification speed and storage consumption are improved. The procedure that they follow in selecting the best attribute at each node of the tree is based on the *C4.5* method which is a data-based decision tree method. The input to their method requires both the real examples used to create the classifiers (decision trees) and the rules extracted from the classifiers in which they use to create a set of training examples that they use in their method.

III. Rule generation and notations

In this section, we present the notations used to describe the rules used by the method. We also present the methods used to generate the rules that will serve as input to the rule-based decision tree methods in our experiments.

A. Notations

In order for our proposed method to be capable of generating a decision tree for a given dataset, it has to be presented with a set of rules that cover the dataset. The rules will be used as input to *RBDT-1* which will produce a decision tree as an output. The rules can either be provided up front, for instance, by an expert or can be generated algorithmically.

Let $A_1,...,A_n$ denote the attributes characterizing the data under consideration, and let $D_1,...,D_n$ denote the corresponding domains, respectively (i.e. D_i represents the set of values for attribute A_i).

Let $C_1,...,C_m$ represent the decision classes associated with the dataset, and R_i denote the set of rules associated with decision class C_i.

The datasets used later in our experiments are based on classification problems where each example in the dataset belongs to only one class. Thus, a desirable form of a rule-set would be a logically disjoint and complete family of rule-sets. Thus, given a collection of rule-sets, one for each

[1] In the default setting, the *cost* equals 1 for all the attributes. Thus, the *disjointness* criterion is treated as the first criterion of the *AQDT-1* and *AQDT-2* methods in the decision tree building experiments throughout this paper.

class decision, no two rule-sets for two different classes shall logically intersect and the union of all the rule-sets shall cover the whole dataset.

B. Rule generation method

In our experiments we are comparing the decision tree generated by our proposed method to the *AQDT-1, AQDT-2, and* the *ID3* methods. Since all the methods under comparison except the *ID3* method are rule-based decision tree methods, one of the best ways to perform a fair comparison is to use the same rules produced by the *ID3* itself. In addition, using *ID3-based* rules will give us an opportunity to illustrate our method's capability of producing a smaller tree while using the same rules extracted from a decision tree generated by a data-based decision tree method without reducing the tree's accuracy.

The second option that we use for generating rules is to use an *AQ-type* rule induction program. *AQ-type* programs such as *AQ19* [6], *AQ21* [7], are a family of programs for machine learning and pattern discovery, which are capable of inducing rules from data. In our upcoming experiments we used the *AQ19* program for creating logically disjoint rules which we refer to as *AQ-based* rules.

IV. RBDT-1 METHOD

In this section, we describe the *RBDT-1* method by first outlining the format of the input rules used by the method and the attribute selection criteria of the method, and then summarizing the main steps of the underlying decision tree building process.

A. Preparing the rules

The decision rules must be prepared into the proper format used by the *RBDT-1* method. This is done by assigning a "don't care" value to all the attributes that were omitted in any of the rules. The "don't care" value is equivalent to listing all the values for that attribute. For example, suppose that we have three attributes A_1, A_2, *and* A_3, with V_1, V_2, and V_3 as possible values.

Let us assume that the following rules correspond to class C_1:

R_{11}: $C_1 \leftarrow A_1=V_1 \,\&\, A_2=V_2$

R_{12}: $C_1 \leftarrow A_1=V_3$

The preparation of these two rules will result in the following formatted rules:

R_{11}: $C_1 \leftarrow A_1=V_1 \,\&\, A_2=V_2 \,\&\, A_3=$"don't care".

R_{12}: $C_1 \leftarrow A_1=V_3$ & $A_2=$"don't care" & $A_3=$"don't care".

Each rule is submitted to *RBDT-1* in the form of an attribute-value vector. This vector will contain first the values of the attributes that appear in the rule followed by the class decision as the last element of the vector. Thus, accordingly the previous two rules will be transformed into the following attribute-value vectors:

R_{11}: (V_1, V_2, don't care, C_1)

R_{12}: (V_3, don't care, don't care, C_1)

B. Attribute selection criteria

The *RBDT-1* method applies three criteria on the attributes to select the fittest attribute that will be assigned to each node of the decision tree. These criteria are *the Attribute Effectiveness*, *the Attribute Autonomy*, *and the Minimum Value Distribution*.

- Attribute effectiveness (AE). *AE* is the first criterion to be examined for the attributes. It prefers an attribute which has the most influence in determining the decision classes. In other words, it prefers the attribute that has the least number of "don't care" values for the class decisions in the rules, as this indicates its high relevance for discriminating among rule sets of given decision classes. On the other hand, an attribute which is omitted from all the rules (i.e. has a "don't care" value) for a certain class decision does not contribute in producing that corresponding decision. So it is considered less important than the other attributes which are mentioned in the rule for producing a decision of that class. Choosing attributes based on this criterion maximizes the chances of reaching leaf nodes faster which on its turn minimizes the branching process and leads to producing a smaller tree.

Using the notation provided above (see section 3), let V_{ij} denote the set of values for attribute a_j involved in the rules in R_i, which denote the set of rules associated with decision class c_i, $1 \le i \le m$. Let *DC* denote the 'don't care' value, we calculate $C_{ij}(DC)$ as shown in (1):

$$C_{ij}(DC) = \begin{cases} 1 & if\ DC \in V_{ij} \\ 0 & otherwise \end{cases} \qquad (1)$$

Given an attribute a_j, where $1 \le j \le n$, the corresponding attribute effectiveness is given in (2).

$$AE(a_j) = \frac{m - \sum_{i=1}^{m} C_{ij}(DC)}{m} \qquad (2)$$

(Where *m* is the total number of different classes in the set of rules).

The attribute with the highest *AE* is selected as the fit attribute.

- Attribute autonomy (AA). *AA* is the second criterion to be examined for the attributes. This criterion is

examined when the highest *AE* score is obtained by more than one attribute. This criterion prefers the attribute that will decrease the number of subsequent nodes required ahead in the branch before reaching a leaf node. Thus, it selects the attribute that is less dependent on the other attributes in deciding on the decision classes. We calculate the attribute autonomy for each attribute and the one with the highest score will be selected as the fit attribute.

For the sake of simplicity, let us assume that the set of attributes that achieved the highest *AE* score are $a_1,...,a_s,\ 2 \le s \le n$. Let $v_{j1},...,v_{jp_j}$ denote the set of possible values for attribute a_j including the "don't care", and R_{ji} denote the rule subset consisting of the rules that have a_j appearing with the value v_{ji}, where $1 \le j \le s$ and $1 \le i \le p_j$. Note that R_{ji} will include the rules that have don't care values for a_j as well.

The *AA* criterion is computed in terms of the Attribute Disjointness Score (ADS), which was introduced by [1]. For each rule subset R_{ji}, let $MaxADS_{ji}$ denote the maximum ADS value and let ADS_List_{ji} denote a list that contains the *ADS* score for each attribute a_k, where $1 \le k \le s, k \ne j$.

According to [1], given an attribute a_j and two decision classes c_i and c_k (where $1 \le i,k \le m; 1 \le j \le s$), the degree of disjointness between the rule set for c_i and the rule set for c_j with respect to attribute a_j is defined as shown in (3):

$$ADS(A_j, C_i, C_k) = \begin{cases} 0 & if\ V_{ij} \subseteq V_{kj} \\ 1 & if\ V_{ij} \supseteq V_{kj} \\ 2 & if\ V_{ij} \cap V_{kj} \ne (\varnothing\ or\ V_{ij}\ or\ V_{kj}) \\ 3 & if\ V_{ij} \cap V_{kj} = \varnothing \end{cases} \quad (3)$$

The *Attribute Disjointness* of the attribute a_j; $ADS(a_j)$ score is the summation of the degrees of class disjointness $ADS(a_j, c_i, c_k)$ given in (4):

$$ADS(a_j) = \sum_{i=1}^{m} \sum_{\substack{1 \le k \le s \\ i \ne k}} ADS(a_j, c_i, c_k) \quad (4)$$

Thus, the number of *ADS_List* that will be created for each attribute a_j as well as the number of *MaxADS* values that are calculated will be equal to p_j. The $MaxADS_{ji}$ value as defined by [8] is $3 \times m \times (m-1)$ where *m* is the total number of classes in R_{ji}. We introduce the *AA* as a new criterion for attribute a_j as given in (5):

$$AA(a_j) = \frac{1}{\sum_{i=1}^{p_j} AA(a_j, i)} \quad (5)$$

Where $AA(a_j, i)$ is defined as shown in (6):

$$AA(a_j, i) = \begin{cases} 0 & if\ \ MaxADS_{ji} = 0 \\ 1 & if\ \left(\left(MaxADS_{ji} \ne 0 \right) \wedge \left(\left(s = 2 \right) \vee \left(\left(\exists l : MaxADS_{ji} = ADS\_List_{ji}[l] \right) \right) \right) \right) \\ 1 + \left[(s-1) \times MaxADS_{ji} - \sum_{l=1, l \ne j}^{s} ADS\_List_{ji}[l] \right] & otherwise \end{cases} \quad (6)$$

The *AA* for each of the attributes is calculated using the above formula and the attribute with the highest *AA* score is selected as the fit attribute. According to the above formula, $AA(a_j, i)$ equals zero when the class decisions for the rule subset examined corresponds to one class, in that case *MaxADS=0*, which indicates that a leaf node is reached (best case for a branch). $AA(a_j, i)$ equals 1 when *s* equals 2 or when one of the attributes in the *ADS_list* has an *ADS* score equal to *MaxADS* value (second best case). The second best case indicates that only one extra node will be required to reach a leaf node. Otherwise $AA(a_j, i)$ will be equal to 1 + (the difference between the *ADS* scores of the attributes in the *ADS_list* and the *MaxADS* value) which indicates that more than one node will be required until reaching a leaf node.

- Minimum value distribution (MVD). The *MVD* criterion is concerned with the number of values that an attribute has in the current rules. When the highest *AA* score is obtained by more than one attribute, this criterion selects the attribute with the minimum number of values in the current rules. *MVD* criterion minimizes the size of the tree because the fewer the number of values of the attributes the fewer the number of branches involved and consequently the

smaller the tree will become [1]. For the sake of simplicity, let us assume that the set of attributes that achieved the highest *AA* score are $a_1,...,a_q$, $2 \le q \le s$. Given an attribute a_j (where $1 \le j \le q$), we compute corresponding *MVD* value as shown in (7).

$$MVD(A_j) = \left| \bigcup_{1 \le i \le m} V_{ij} \right| \qquad (7)$$

(Where |X| denote the cardinality of set *X*).

When the lowest *MVD* score is obtained by more than one attribute, any of these attributes can be selected randomly as the fit attribute. In our experiments in case where more than two attributes have the lowest *MVD* score we take the first attribute.

V. BUILDING THE DECISION TREE

In the decision tree building process, we select the fit attribute that will be assigned to each node from the current set of rules *CR* based on the attribute selection criteria outlined in the previous section. *CR* is a subset of the decision rules that satisfy the combination of attribute values assigned to the path from the root to the current node. *CR* will correspond to the whole set of rules at the root node.

From each node a number of branches are pulled out according to the total number of values available for the corresponding attribute in *CR*.

Each branch is associated with a reduced set of rules *RR* which is a subset of *CR* that satisfies the value of the corresponding attribute. If *RR* is empty, then a single node will be returned with the value of the most frequent class found in the whole set of rules. Otherwise, if all the rules in *RR* assigned to the branch belong to the same decision class, a leaf node will be created and assigned a value of that decision class. The process continues until each branch from the root node is terminated with a leaf node and no more further branching is required.

VI. EXPERIMENTS

In this section, we present the evaluation of our proposed method by comparing it with the *AQDT-1 &2*, and the *ID3* based on the 12 public datasets listed in Table 1. Other than the weekend dataset, all the datasets were obtained from the UCI machine learning repository [9].

Two experiments were conducted, one comparing the methods using *ID3-based* rules as input for the rule-based methods and another experiment using rule-sets generated by an *AQ-type* induction program; *AQ19*.

Our evaluation consisted of comparing the decision trees produced by the *RBDT-1*, *AQDT-1*, *AQDT-2*, and *ID3* methods for each dataset in terms of tree complexity (number of nodes and leaves) and accuracy. No tree pruning is applied by any method in this comparison. The results are summarized in Tables 1 and 2. In both tables, the name of the method that produced a less complex tree appears in the table under the *method* column, on the other side *"="* indicates that the same tree was obtained by all methods under comparison. All four methods run under the assumption that they will produce a complete and consistent decision tree yielding 100% correct recognition on the training examples.

TABLE 1. Summary of the 12 datasets used in our experiments.

Dataset Name	# Records	Dataset Name	# Records
Weekend	*10*	**Connect-4**	*67557*
Lenses	*24*	**Nursery**	*12960*
King-Rook vs. King-Knight chess	*2021*	**Balance Scale**	*625*
Car evaluation	*1728*	**MONK's 1**	*432*
Zoo	*101*	**MONK's 2**	*432*
Shuttle Landing Control	*253*	**MONK's 3**	*432*

In the first experiment the rules used as input to *RBDT-1*, *AQDT-1, and AQDT-2* were extracted from the *ID3* decision tree built from the whole set of examples of the datasets in Table 1. Thus, the *ID3-based* rules used in this experiment cover the whole set of examples (100% coverage). The size of the extracted *ID3-based* rules is equal to the number of leaves in the *ID3* tree.

Table 2 illustrates that, for 7 of the datasets, the *RBDT-1* produced a smaller tree than that produced by *AQDT-1 & 2* with an average of 274 nodes less while producing a same tree for the rest of the datasets. On the other hand, *RBDT-1* produced a smaller tree in 5 datasets compared to *ID3* with an average of 142.6 nodes less while producing a same tree for the rest of the datasets. The classification accuracies for all four methods were equal.

TABLE 2. Comparison of tree complexities of the methods using ID3-based rules.

Dataset	Method	Dataset	Method
Weekend	*RBDT-1*	**MONK's 1**	*RBDT-1*
Lenses	*RBDT-1, ID3*	**MONK's 2**	*RBDT-1, AQDT-1 & 2*
Chess	=	**MONK's 3**	=

Car	*RBDT-1, ID3*	**Zoo**	=
Shuttle-L-C	=	**Nursery**	*RBDT-1*
Connect-4	*RBDT-1*	**Balance**	*RBDT-1, ID3*

While using *ID3-based* rules, *AQDT-1 & 2* methods led to a syntactical instable decision tree in some cases such as for the Monk's 1, balance scale, car, and connect-4 problems. That is, the structure of the final decision tree varied considerably in size each time the process is repeated for the same original training examples. Naturally, this syntactical instability is undesirable, for instance, if the methods are applied in knowledge acquisition.

Table 3 presents the results of another experiment that we conducted for comparing the decision tree complexity and accuracy of the *RBDT-1, AQDT-1 & 2* methods along with the *ID3* method. In this experiment we used rules generated by the *AQ19* rule induction program with 100% correct recognition on the datasets used as input for the rule-based methods under comparison.

TABLE 3. Comparison of tree complexities of the methods using AQ-based rules.

Dataset	Method	Dataset	Method
Weekend	*RBDT-1*	**Monk's2**	*RBDT-1*
lenses	*RBDT-1, ID3*	**Monk's3**	=
Zoo	*RBDT-1*	**Chess**	=
Car	*RBDT-1*	**Balance**	*RBDT-1, ID3*
Monk's1	*RBDT-1*	**Shuttle-L-C**	*RBDT-1, AQDT-1 & 2*

Based on the results of the comparison in Table 4, the *RBDT-1* method performed better than the *AQDT-1 & 2* and the *ID3* methods in most cases in terms of tree complexity by an average of 33.1 nodes less than *AQDT-1 & 2* and by an average of 88.5 nodes less than the *ID3* method. The accuracy classification of all four methods was equal.

VII. CONCLUSIONS AND FUTURE WORK

The *RBDT-1* method proposed in this work allows generating a decision tree from a set of rules rather than from the whole set of examples. Following this methodology, knowledge can be stored in a declarative rule form and transformed into a decision structure when it is needed for decision making. Generating a decision structure from decision rules can potentially be performed much faster than by generating it from training examples.

In our experiments, our proposed method was compared to two other rule-based decision tree methods; *AQDT-1* and *AQDT-2* using *ID3-based* rules and *AQ-based* rules. *RBDT-1* was also compared to the *ID3* method which is a data-based decision tree method. Based on the results of the comparison the *RBDT-1* method performs better than the other three methods under comparison in most cases in terms of tree complexity and achieves at least the same level of accuracy.

REFERENCES

[1] R. S. Michalski and I. F. Imam, "Learning problem-oriented decision structures from decision rules: the AQDT-2 system", In Proceedings of 8th International Symposium Methodologies for Intelligent Systems. Lecture Notes in Artificial Intelligence, 869, Springer Verlag, Heidelberg, 1994, pp. 416-426.

[2] I. F. Imam, and R. S. Michalski "Should decision trees be learned from examples of from decision rules?", Source Lecture Notes in Computer Science. In Proceedings of the 7th International Symposium on Methodologies, 689, 1993, pp. 395–404.

[3] J. R. Quinlan "Discovering rules by induction from large collections of examples", In D. Michie (Edr), Expert Systems in the Microelectronic Age, Edinburgh University Press, 1979, pp. 168-201.

[4] I. H. Witten, and B. A. MacDonald, "Using concept learning for knowledge acquisition", International Journal of Man-Machine Studies, 1988, pp. 349-370.

[5] Y. Akiba,, S. Kaneda, and H. Almuallim, "Turning majority voting classifiers into a single decision tree", In Proceedings of the 10th IEEE International Conference on Tools with Artificial Intelligence, 1998, pp. 224-230.

[6] R. S. Michalski, and K. Kaufman, "The AQ19 system for machine learning and pattern discovery: a general description and user's guide", Reports of the Machine Learning and Inference Laboratory, MLI 01-2, George Mason University, Fairfax, VA, 2001.

[7] J. Wojtusiak, "AQ21 User's Guide". Reports of the machine learning and inference laboratory, MLI 04-5, George Mason University, 2004.

[8] Colton, S. Online Document, Available: http://www.doc.ic.ac.uk/~sgc/teaching/v231/lecture11.html. 2004

[9] J. Demsar,, B. Zupan, G. Leban, UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science, 2007.