RBDT-1: a New Rule-based Decision Tree Generation Technique

Amany Abdelhalim, Issa Traore, Bassam Sayed

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
{amany, itraore, bassam}@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. 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 rule-based decision tree methods in terms of tree complexity (number of nodes and leaves in the decision tree). It is also shown that *RBDT-1* performs equally well in terms of tree complexity compared with *C4.5*, which generates a decision tree from data examples.

Keywords: attribute selection criteria, decision rules, data-based decision tree, rule-based decision tree, tree complexity.

1 Introduction

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. The *attribute selection criteria* are the essential characteristics in all those methods [1]. These criteria are used to choose the best attributes to be assigned to the nodes of the decision tree. Examples of such criteria include the entropy reduction [2], the gini index of diversity [3], and others [4], [5].

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

Decision trees 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 once constructed. This is because a decision tree is a procedural knowledge representation, which imposes an evaluation order on the attributes in the tree. 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 [6].

On the other hand, in order to be able to make a decision for some situation using the set of rules we need to decide the order in which tests should be evaluated. In that case a decision structure (e.g. decision tree) will be created from the rules. So the methods that create decision trees from rules 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 [6].

In addition to that, generating a decision structure from decision rules can potentially be performed faster than generating it from training examples because the number of decision rules per decision class is usually much smaller than the number of training examples per class. Thus, this process could be done on demand without any noticeable delay [2], [7]. Methods that create decision trees from examples or data require examining the complete tree to extract information about any single classification. Even converting the tree into a set of individual rules could also result in a large amount of rules if the tree is large, which could be the case when the tree is based on a large dataset. Otherwise, with methods that create decision trees from rules, extracting information about any single classification can be done directly from the declarative rules [8].

Although rule-based decision tree methods create decision trees from rules, they could be used also to create decision trees from examples by considering each example as a rule. Data-based decision tree methods create decision trees from data only. Thus, when generating a decision tree for problems were rules are provided e.g. by an expert and no data is available, rule-based decision tree methods are the only applicable solution.

This paper presents a new rule-based decision tree method called *RBDT-1*. To derive the tree, the *RBDT-1* method uses in sequence three different criteria to determine the fit (best) attribute for each node of the tree, which are referred to as the attribute effectiveness (AE), the attribute autonomy (AA), and the minimum value distribution (MVD).

The rest of the paper is structured as follows. Section 2 summarizes the related work. Section 3 discusses the rule generation approach and notations used in this work. Section 4 describes the *RBDT-1* method by illustrating, in particular, the preparation of the rules into a format that will be used by the method, the different criteria used in the attribute selection process, the pruning technique adopted by the method and also by providing an illustration of the method using a small dataset.

Section 5 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* [6] and *AQDT-2* [8] methods, and to the *C4.5* algorithm [9] that creates decision trees from data examples. In Section 6, we make some concluding remarks and outline our future work.

2 Related Work

There are few published works on creating decision structures from declarative rules. The *AQDT-1* method introduced in [6] 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 settings.

The *AQDT-2* method introduced in [8] 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*, *disjointness*, *information importance*, *value distribution*, and *dominance*, which are also applied in the same specified sequence in the method's default settings. In both the *AQDT-1* & *2* methods, the order of each criterion expresses its level of importance in deciding the attribute that 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, which contradicts the method's fundamental idea of being a rule-based decision tree method rather than a data-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. *AQDT-2* being both dependent on the examples as well as the rules results in an increase of the running time of the algorithm remarkably in large datasets especially those with large number of attributes.

In contrast the multi criteria calculations of the *RBDT-1* method, proposed in this work, for generating a decision tree require only the set of rules given to the method as an input, and does not require the presence of the examples used to induce those rules. The calculations of all the method's criteria are based on certain characteristics of the attributes intrinsic to the rules only.

Akiba et al. [10] 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 extracted from each classifier which is a decision tree generated using the data-based decision tree method *C4.5*. They use the extracted rules used to learn a single decision tree. The goal of the method is to provide an approximation of the majority voting classifier rather than an exact matching behavior. The method that they propose depends both on the real examples used to create the classifiers (decision trees) and on a set of training examples that

they create using the rules extracted from the classifiers. The procedure that they follow in selecting the best attribute at each node of the tree is based on the *C4.5* as well. The size of a decision tree learned by Akiba et al. method while using rules extracted from multiple classifiers built by *C4.5* is about 1.2 to 4.2 times the size of a decision tree learned by *C4.5* from the data [10]. As will be shown in the experiments, when using rules extracted from a *C4.5* decision tree, the *RBT-1* method generates a tree that is the same size as the decision tree learned by *C4.5* from data, even smaller in some of the cases with an equal accuracy.

In [17], the authors proposed a method called Associative Classification Tree (ACT) for building a decision tree from association rules rather than from data. They proposed two splitting algorithms for choosing attributes in the *ACT*. The first algorithm is based on the confidence gain criterion and the second is based on the entropy gain criterion. In both splitting algorithms the attribute selection process at each node relies on both the existence of rules and the data itself as well. Unlike our proposed method *RBDT-1, ACT* is not capable of building a decision tree from the rules in the absence of data, or from data (considering them as rules) in the absence of rules.

3 Rule Generation and Notations

In order for our proposed method to be capable of generating a decision tree for a certain 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.

The datasets that we use 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 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. In such a case, each possible example in the dataset will belong to one of the predefined classes. So the decision classes induce a partition over the complete set of rules. Let P denote the complete set of rules and R_i denote the set of rules associated with decision class c_i . Hence, we have the following: $i \neq j \Rightarrow R_i \cap R_j = \varnothing$, where $1 \leq i, j \leq m$; $\mathrm{P} = \bigcup_{1 \leq i \leq m} R_i$

In our main experiment, we are comparing the decision tree generated by our proposed method to the *AQDT-1, AQDT-2,* and the *C4.5* methods. Since all the methods under comparison except the *C4.5* method are rule-based decision tree methods, one of the best ways to perform a fair comparison is to use the same rules

extracted from *C4.5* decision tree itself as input to the other three rule-based methods. The method used to extract rules from the *C4.5* decision tree consists of converting each branch – from the root to a leaf – of the decision tree to an if-then rule whose condition part is a pure conjunction. This approach ensures that we will have a collection of disjoint rules.

We used *AQ19* [11] to generate the rule set used to illustrate *RBDT-1* method. *AQ19* is a rule induction program that belongs to the famous *AQ-type* family for machine learning and pattern discovery techniques, which are capable of creating logically disjoint rules.

4 RBDT-1 Method

In this section, we outline the format of the rules required for *RBDT-1* method, the attribute selection criteria of the method, and then summarize the main steps of the underlying decision tree building process. We also present the pruning technique adopted by the method. Finally, we illustrate the steps for generating a decision tree by the *RBDT-1* method using a small rule set.

4.1 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 the same domain containing v_1, v_2 and v_3 as possible values.

Let us assume that the following rules correspond to class *c1*:

r1: c1 ← $a_1=v_1$ & $a_2=v_2$, *r2: c1* ← $a_1=v_3$

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

r1: c1 ← $a_1=v_1$ & $a_2=v_2$ & a_3="don't care", *r2: c1* ← $a_1=v_3$ & 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 representing the last element of the vector. Thus, accordingly the previous two rules will be presented as follows:

r1: (v_1, v_2, *don't care, c1), r2: (*v_3, *don't care, don't care, c1)*

4.2 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):

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

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

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

(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. If more than one attribute achieve the highest *AE* score we will use the next criterion in our method, which is the *Attribute Autonomy* to determine the best attribute among them.

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. If more than one attribute achieve the highest *AA* score, we will use the next criterion in our method which is the *Minimum Value Distribution* to determine the best attribute among them.

For the sake of simplicity, let us assume that the set of attributes that achieved the highest *AE* score are $a_1,...,a_s,\ 2 \le s \le \mathrm{n}$. 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 [8]. 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 [8], 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):

$$ADS(a_j, c_i, c_k) = \begin{cases} 0 & if\ V_{ij} \subseteq V_{kj} \\ 1 & if\ V_{ij} \supseteq V_{kj} \\ 2 & if\ V_{ij} \cap V_{kj} \ne (\varnothing\ or\ V_{ij}\ or\ V_{kj}) \\ 3 & if\ V_{ij} \cap V_{kj} = \varnothing \end{cases} \tag{3}$$

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):

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

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):

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

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

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

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

$$MVD(a_j) = \left| \bigcup_{1 \le i \le m} V_{ij} \right| . \quad (7)$$

(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.

4.3 Building the Decision Tree

We describe, in this section, the *RBDT-1* approach for building a decision structure from a set of decision rules. In our case the decision structure is a decision tree which is a single-parent decision structure. 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^1 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. 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.

4.4 Pruning Decision Rules

RBDT-1 is capable of handling the problem of generating a decision tree from noisy training data. In *RBDT-1*, we handle noisy data by removing rules that cover only a small portion of the data that could be considered noise [12]. The examples that were covered by the truncated rules can often be covered by applying an analogical matching procedure. The analogical matching procedure determines the degree of similarity between the examples to be classified and the rules of a given decision class, and selects the best matching decision class [13]. In [14] experiments show that such a rule truncation method not only simplifies decision rules which could lead to a simpler decision tree, but could also improve their prediction accuracy in some cases.

In *RBDT-1*, rules are pruned if their support level is less than or equal to a predefined threshold. The support level of a rule is the percentage of the total number of examples covered by the rule (called *the t-weight*) to the total number of examples in the given decision class.

[1] *CR* will correspond to the whole set of rules at the root node.

4.5 The Weekend Problem

In this section, the steps for generating a decision tree by the *RBDT-1* method will be explained in detail for a small dataset called the Weekend problem. The Weekend problem is a dataset that consists of 10 data records obtained from [15]. We used the *AQ19* rule induction program to induce the rule set shown in Table 1 which will serve as the input to our proposed method *RBDT-1* in this example. *AQ19* was used with the mode of generating disjoint rules and with producing a complete set of rules without truncating any of the rules.

Table 1. The weekend rule set induced by AQ19.

Rule	Description
r1	*Cinema ← Parents-visiting="yes" & weather ="don't care" & Money ="rich"*
r2	*Tennis ← Parents-visiting="no" & weather ="sunny" & Money ="don't care"*
r3	*Shopping ← Parents-visiting="no" & weather ="windy" & Money ="rich"*
r4	*Cinema ← Parents-visiting="no" & weather ="windy" & Money ="poor"*
r5	*Stay-in ← Parents-visiting="no" & weather ="rainy" & Money ="poor"*

In order to choose the fit attribute for the root node of the tree, we first apply the three criteria of the *RBDT-1* method on the attributes in the rules presented in Table 1. The criteria are applied in the same order explained in the previous section. The *AE* calculations for Parents-Visiting, Money and Weather attributes are {1, 0.75, 0.75} respectively. Thus, the candidate attribute with the highest *AE* is the *parents-visiting* attribute. Two branches will be pulled out from the *parents-visiting* attribute corresponding to its two values in the current rules, namely *yes and no*.

A subset of the weekend rule set will be assigned to the branch where *parents-visiting="yes"*. This subset of rules will consist of all the rules with *parents-visiting="yes"* or *"don't care"*. The corresponding subset contains only one rule corresponding to rule {r1} with "*cinema*" as a decision. Thus, a leaf node "*cinema*" will be created and assigned to the branch where *parents-visiting="yes"*. Another subset of the weekend rule set will be assigned to the branch where *parents-visiting="no"* corresponding to rules {r2, r3, r4, r5}. This subset of rules will consist of all the rules with *parents-visiting="no"* or *"don't care"*.

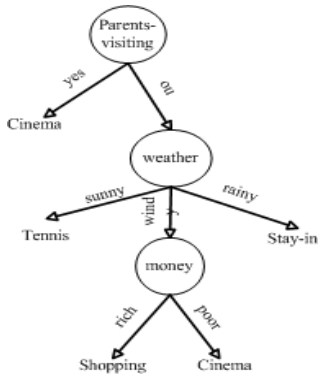

Fig 1. The decision tree generated by RBDT-1 for the weekend problem before pruning.

The *AE* calculations for Weather and Money attributes are {1, 0.75} respectively. Thus, the attribute with the highest *AE* ;the *weather* attribute will be selected as the fit attribute for the branch *parents-visiting="no".* Three branches will be pulled out from the *weather* attribute corresponding to its three values; *sunny, windy, and rainy* found in the *CR.* For the branch where *parents-visiting="no" and weather="sunny"*, there is only one rule that corresponds to that branch with a class decision *tennis* which is rule {r2}. So a leaf node with *tennis* as a decision will be created and assigned to that branch.

There are two rules corresponding to the branch where *parents-visiting="no" and weather="windy"* which are rules {r3, r4}. One of the rules corresponds to the decision class *shopping* and the other rule corresponds to the decision class *cinema*. Both rules depend on the value of the *money* attribute for producing the decision. The *money* attribute will be chosen as the fit attribute since it is the only candidate attribute left. A node will be created and assigned the attribute *money*. Two branches will be pulled out from the *money* node. A leaf node will be assigned the class decision *cinema* for the branch where *money*= "*poor"*. Another leaf node will be assigned the class decision *shopping* for the branch where *money*="*rich"*.

For the branch where *Parents-Visiting="no" and Weather="rainy"*, there is only one rule that corresponds to that branch with a class decision *stay-in*, thus it will be assigned to that branch. Overall, the corresponding decision tree created by the proposed *RBDT-1* method for the weekend problem is shown in Fig 1. It consists of 3 nodes and 5 leaves with 100% classification accuracy for the data. The decision tree created by *AQDT-1* & *2* using the same set of rules results in a tree of 5 nodes and 7 leaves - which is a bigger tree than that created by *RBDT-1* - with 100% classification accuracy for the data.

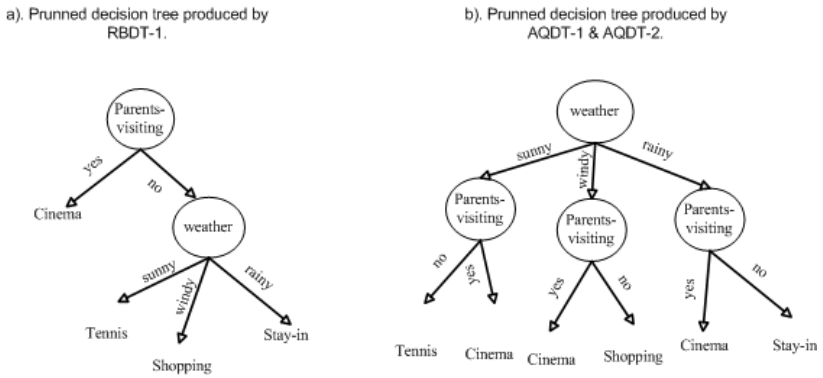

Fig 2. The pruned decision tree generated by RBDT-1, AQDT-1 and AQDT-2 for the weekend problem.

As indicated above, in *RBDT-1*, rules are pruned if their support level is less than or equal to a predefined threshold. Fig 2 shows a decision tree obtained after pruning the

decision rules for the weekend problem in Table 1. In this case, we removed rules with the lowest support level. The produced decision tree by *RBDT-1* misclassified one example out of 10 giving a predictive accuracy of 90% and consists of 2 nodes and 4 leaves. Fig 2 also shows the decision tree produced by *AQDT-1* & *2* using the same pruned rules; although the produced tree has the same accuracy as the tree produced with *RBDT-1*, it is bigger in size since it has 4 nodes and 6 leaves. When applying *C4.5* to the dataset of the weekend problem it resulted in a tree of the same size and accuracy as *RBDT-1* in Fig 2.

5 Experiment and Results

In order to evaluate the *RBDT-1* method, we conducted an experiment using 16 publicly available datasets, including the weekend dataset used in the previous section.

Table 2. Two sets of C4.5 Decision trees with prune option on and off along with the number of extracted rules and accuracy from each.

Dataset	Prune option off			Prune option on		
	Tree Size (# nodes, # leaves)	**# Extracted Rules**	**Acc[2]**	**Tree Size (# nodes, # leaves)**	**# Extracted Rules**	**Acc**
Weekend	*(2,4)*	*4*	*90 %*	*(2,4)*	*4*	*90 %*
Lenses	*(3,4)*	*4*	*91 %*	*(3,4)*	*4*	*91 %*
Chess	*(5, 10)*	*10*	*100 %*	*(5, 10)*	*10*	*100 %*
Car	*(52, 134)*	*134*	*96 %*	*(51,131)*	*131*	*96 %*
Tic-Tac-Toe	*(69, 139)*	*139*	*95 %*	*(47, 95)*	*95*	*93 %*
Connect-4	*(*5317, 10635*)*	10635	*91 %*	*(2142, 4285)*	*4285*	*87 %*
Nursery	*(*264, 680*)*	680	*99 %*	*(152, 359)*	*359*	*98 %*
Balance	*(22, 89)*	*89*	*80 %*	*(8,33)*	*33*	*75 %*
MONK's 1	*(29, 56)*	*56*	*97 %*	*(13,28)*	*28*	*100 %*
MONK's 2	*(88, 166)*	*166*	*85 %*	*(1,1)*	*1*	*67 %*
MONK's 3	*(5, 14)*	*14*	*100 %*	*(5,14)*	*14*	*100 %*
Zoo	*(8, 13)*	*13*	*99 %*	*(8, 13)*	*13*	*99 %*

[2] *Acc* refers to the accuracy of the rules extracted and is calculated as the percentage of the number of examples correctly classified by the rules to the total number of examples.

Breast-C	*(27, 113)*	*113*	*85 %*	*(2,4)*	*4*	*74 %*
Lung -C	*(7, 12)*	*12*	*90 %*	*(6, 10)*	*10*	*87 %*
Primary-T	*(56, 67)*	*67*	*57 %*	*(41, 47)*	*47*	*58 %*
Voting	*(18, 19)*	*19*	*92 %*	*(5,6)*	*6*	*93 %*

Other than the weekend dataset, all the datasets appearing in Tables 3 and 4 were obtained from the UCI machine learning repository [16].

The evaluation consisted mainly of comparing the *RBDT-1* method with the *AQDT-1, AQDT-2,* and *C4.5* methods in terms of the complexity and accuracy of the decision trees produced. Since we were comparing our proposed method to *AQDT-1 & 2* methods which are all rule-based decision tree methods, it was a good idea to compare their performance with multiple rule sets.

The rules used for comparing the decision trees of *RBDT-1, AQDT-1, and AQDT-2* in the experiment are *C4.5-based rules.* Since *C4.5* is capable of handling datasets with missing values, we were capable of experimenting with both rules extracted from complete and incomplete datasets.

In our experiment, we used the *C4.5* method under its default settings to generate two different decision trees for each dataset. One tree was generated with the pruning option turned on and the other with the pruning option turned off. Since the comparison was based on 16 datasets, 32 decision trees were generated. From each decision tree we extracted a rule set - as explained in section 3- which served as the input to the other three rule-based methods under comparison. So accordingly 32 rule sets were used in this experiment. The process is summarized in Table 2.

In Table 3, we illustrate the results of the comparison between *RBDT-1, AQDT-1, AQDT-2* and *C4.5* in two experiments, labelled experiment1 and experiment2. In each experiment the *C4.5* decision tree was generated from the whole set of examples of each dataset and the rules extracted from that tree served as input to the other three rule-based decision tree methods. In experiment1, the pruning option was turned off, while in experiment2 the pruning option was turned on. The name(s) appearing under each experiment correspond(s) to the method(s) that generated the smallest tree, while "=" indicates that all four methods produced the same tree.

Table 3. Comparison of the tree complexity of RBDT-1, AQDT-1, AQDT-2 & C4.5

Dataset	Experiment1	Experiment2	Dataset	Experiment1	Experiment2
Weekend	*RBDT-1, C4.5*	*RBDT-1, C4.5*	**MONK's 1**	*RBDT-1*	=
Lenses	*RBDT-1, C4.5*	*RBDT-1, C4.5*	**MONK's 2**	=	=
Chess	=	=	**MONK's 3**	*AQDT-1 & 2*	*AQDT-1 & 2*
Car	*RBDT-1, C4.5*	*RBDT-1, C4.5*	**Zoo**	*RBDT-1, C4.5*	*RBDT-1, C4.5*
Tic-Tac-Toe	=	=	**Breast-C**	=	=

Connect-4	*RBDT-1*	*RBDT-1, C4.5*	**Lung-C**	*RBDT-1, C4.5*	*RBDT-1, C4.5*
Nursery	*RBDT-1, AQDT-1 & 2*	=	**Primary-T**	*RBDT-1, C4.5*	*RBDT-1, C4.5*
Balance	=	=	**Voting**	*AQDT-1 &2*	=

Results in Table 3 show that, in terms of tree size, *RBDT-1* performs better than *AQDT-1 & 2* in most of the rule sets. Based on the results in the two experiments, *AQDT-1 & 2* produce a larger tree by an average of 146.33 nodes with the exception of 3 rule sets where *RBDT-1*'s tree is larger by an average of 3 nodes. In addition, the results illustrate that *RBDT-1* is as effective as *C4.5* except in experiment1, where our method produced a slightly smaller tree for the *connect-4*, *MONK's 1* and *nursery* rule sets. In terms of accuracy, the four methods have equal performance.

6 Conclusion 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.

Modifications to the data are handled easier in rule-based methods than in data-based methods. This is because the modifications are applied to the rules rather than the decision tree itself. At the same time rule-based methods could transform the rules to a decision tree once we need to decide the order in which tests should be evaluated in those rules. Rule-based decision tree methods although designed to create decision trees from rules, could also generate decision trees from data examples. Rule-based decision tree methods are the only solution for generating a decision tree for applications where no data is available and only rules exist. The price of the *RBDT-1* advantages is the need to generate rules first before being capable of generating the tree. However, there are efficient rule learning systems available.

Experiments conducted in this work illustrates that in terms of tree complexity our proposed method *RBDT-1* performs better than *AQDT-1 & 2* in most of the rule sets, with an equal accuracy classification, while it is as effective as *C4.5* in terms of tree complexity and accuracy.

In our future work we will conduct more experiments using rule sets produced by different rule generation methods. We will also extend our method to address the problem of learning from rules that do not logically intersect. We intend to apply our method in application fraud detection where fraud data is not easily available and instead a rule-base could be created based on heuristics and expert knowledge. *RBDT-1* can summarize the rule-base into a decision tree for more readability and for obtaining the fastest decision.

References

1 Imam, I. F.: An Empirical Comparison between Learning Decision Trees From Examples and From Decision Rules. In: 9th International Symposium on Methodologies for Intelligent Systems, Zakopane (1996)

2 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, pp. 168--201 (1979)

3 Breiman, L., Friedman, J. H., Oishen, R. A., Stone, C. J.: Classification and Regression Structures. Belmont, California: Wadsworth Int. Group (1984)

4 Cestnik, B., Karalie, A.: The Estimation of Probabilities in Attribute Selection Measures for Decision Structure Induction. In: Proceeding of the European Summer School on Machine Learning, , Priory Corsendonk, Belgium, pp.22--31 (1991)

5 Mingers, J.: An Empirical Comparison of Selection Measures for Decision-Structure Induction. Machine Learning, 3(3): pp. 319--342. Kluwer Academic Publishers (1989).

6 Imam, I. F., Michalski, R. S.: Learning Decision Trees from Decision Rules: A Method and Initial Results from a Comparative Study. J. JIIS. 2(3): pp. 279-304 (1993)

7 Witten, I. H., MacDonald, B. A.: Using Concept Learning for Knowledge Acquisition. J. IJMMS. pp. 349--370 (1988)

8 Michalski, R. S., Imam, I. F.: Learning Problem-Oriented Decision Structures From Decision Rules: the AQDT-2 System. In: 8th International Symposium Methodologies for Intelligent Systems. Lecture Notes in Artificial Intelligence 869. pp. 416--426. Springer Verlag, Heidelberg, (1994)

9 Quinlan, J., R. C4.5: Programs for Machine Learning. San Mateo, CA Morgan Kaufmann (1993).

10 Akiba, Y., Kaneda, S., Almuallim, H.: Turning Majority Voting Classifiers Into A Single Decision Tree. In: 10th IEEE International Conference on Tools with Artificial Intelligence, pp. 224--230 (1998)

11 Michalski, R. S., Kaufman, K.: 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).

12 Michalski, R. S., Imam, I. F.: On Learning Decision Structures. Fundamenta Informaticae, 31(1): pp. 49--64 (1997)

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

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

15 Colton, S.: Online Document, http://www.doc.ic.ac.uk/~sgc/teaching/v231/lecture11.html. (2004)

16 Asuncion, A., Newman, D.J.: UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science (2007)

17 Chen, Y., and L. T. Hung,. 2009. Using decision trees to summarize associative classification rules. Expert Syst. Appl. Pergamon Press, Inc. Publisher, 36(2): pp. 2338-2351.