RBDT-1C mathematical supporting document

The RBDT-1C algorithm is a rule-based method for optimisation of a decision tree structure which does not require any training data for optimisation. The RBDT-1C algorithm is used to determine the ordering of the attributes at the nodes of the decision tree based on four selection criteria: attribute effectiveness (AE), attribute autonomy (AA), the minimum value distribution (MVD) and attribute hierarchy (AH). The RBDT-1C is introduced as a modification to the existing RBDT-1 algorithm [1], allowing additional case specific constraints. The AH criterion is designed to incorporate any additional constraints arising from the legislation, which is calculated for all of the attributes to further reduce the complexity of the decision tree by assessing whether subsequent attributes can be excluded owing to the attribute taking a specific value. The attributes $A_1, \dots, A_n$ correspond to the questions resulting from the rules in a certain set of regulations. These are used to categorise an item of interest at each node of the decision tree. The item of interest represents the aspect(s) upon which the regulations might act. The decision classes $C_1, \dots, C_m$ represent the potential classifications. The complete set of rules for the classification problem is given the notation R, where R_i describes the set of rules associated with decision class C_i. The set of values V_{ij} for each attribute is the set of potential answers to each question. For a rule within R_i, corresponding to decision class C_i, some attributes have to take a specific value in order to satisfy the classification C_i, whereas other attributes can take any value as it will not affect the outcome of the device classification. An attribute which does not affect the classification outcome within a rule can be given any value, which is represented by the following notation: '$*$'.

Attribute effectiveness

The AE score quantifies the importance of an attribute when the determining the classification of an item. An attribute is favoured when the outcome of the classification is dependent on the value it takes. Using notation previously introduced: for a decision class C_i (where $1 \leq i \leq m$) with a corresponding set of rules R_i, an attribute A_j can take one of a set of values V_{ij}. The influence of each of the attributes described above can be quantified by $C_{ij}(*)$ outlined in equation (1).

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

The AE score of A_j can be calculated using equation (2), where m is the number of decision classes.

$$Attribute\ Effectiveness\ (A_j) = \frac{m - \sum_{i=1}^{m} C_{ij}(*)}{m} \quad (2)$$

The attribute with the highest AE score is selected as the root node of the decision tree. For attributes with the same AE score, the AA is then considered.

Attribute Autonomy

The AA criterion aims to reduce the number of subsequent nodes after the root node before a leaf node is reached. This is achieved by prioritising attributes that are less dependent on other attributes to determine the final decision class. The AA is calculated for all attributes, $A_1, \dots, A_S$, $2 \leq s \leq n$, with the same AE score. In order to calculate the AA, the attribute disjointness score introduced by Michalski et al. [2], is first considered. The attribute disjointness score (ADS) is the sum of the class disjointness. The degree of disjointness between two decision classes C_i and C_k can be calculated, for a given attribute A_j, based on the set of values that the attribute A_j takes for class C_i versus class C_k, using equation (3).

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

Once the ADS has been calculated for each of the possible combinations of decision classes, the disjointness of the attribute is found by summing the degrees of the class disjointness using equations (4) and (5).

$$ADS(A_j, C_i) = \sum_{\substack{1 \leq k \leq m \\ i \neq k}} ADS(A_j, C_i, C_k) \quad (4)$$

$$ADS(A_j) = \sum_{i=1}^{m} ADS(A_j, C_i) \quad (5)$$

Two more criteria are introduced in order to calculate the AA: ADS_{list} and $MaxADS$. ADS_{list} is calculated for each of the attributes A_l, where $1 \leq l \leq s$, and s is the number of attributes with the same AE score. The ADS_{list} criterion is calculated multiple times for each attribute, once for each of the values the attribute can take from V_{ij}. The number of times the ADS_{list} criterion is calculated for each attribute is d_j, where d_j is the cardinality of the set of values V_{ij} for the attribute A_l. In order to calculate ADS_{list}, a value (v_i) must first be assigned to the attribute A_l. The ADS_{list} is calculated from a new subset of rules R_{il}. Any rule in which attribute A_l takes the assigned value v_i is included in the subset of rules R_{il} from which the ADS_{list} value is calculated from. ADS_{list} is a list of the ADS values calculated from R_{il} for all the attributes A_s, where $s \neq l$, with the same AE score as A_l – as shown in equation (6).

$$ADS_{list}(A_l, v_i) = [ADS(A_1), \dots, ADS(A_s)]\ where\ 1 \leq l \leq s, s \neq l, using\ R_{il} \quad (6)$$

$MaxADS$ is the maximum value of the attribute disjointness score for the attribute A_l where n is the total number of classes in R_{il} – as shown in equation (7).

$$MaxADS(A_l, v_i) = 3 \times n \times (n-1) \tag{7}$$

The AA for attribute A_l is a function of the AA score for the attribute at each of its possible values v_i, shown by equation (8), where the AA score, $AA(A_l, v_i)$, is defined in equation (9).

$$AA(A_l) = \frac{1}{\sum_{i=1}^{d_j} AA(A_l, v_i)} \tag{8}$$

$$AA(A_l, v_i) = \begin{cases} 0 & if\ MaxADS(A_l, v_i) = 0 \\ 1 & if\ ((MaxADS(A_l, v_i) \neq 0)((s = 2)(\ MaxADS(A_l, v_i) = ADS_{list}(A_l, v_i)))) \\ 1 + [\ (s-1)(MaxADS(A_l, v_i) - \sum_{\substack{l=1 \\ l \neq j}}^{s} ADS_{list}(A_l, v_i))\ ] & otherwise \end{cases} \tag{9}$$

The $MaxADS(A_l, v_i)$ is zero when n is one, as a leaf node has been reached if there is only one possible classification. When there are only two attributes with the same AE score ($s = 2$), the ADS_{list} will only have one value. If this value in ADS_{list} is equal to the $MaxADS(A_l, v_i)$, and $MaxADS(A_l, v_i)$ does not equal zero, then only one extra node is required to reach a leaf node. In this situation $AA(A_l, v_i)$ is given a maximum value of one. If neither of these conditions apply then the $AA(A_l, v_i)$ score is based on the difference between the sum of the ADS scores in $ADS_{list}(A_l, v_i)$ and the $MaxADS(A_l, v_i)$ value.

The primary focus of the attribute autonomy criterion is to reduce the depth of the decision tree, meaning it chooses an attribute that minimises the number of subsequent attributes needed to reach a leaf node. This in turn also has the effect of minimising the width of the decision tree. The AA criterion is more effective than simply using the ADS as it not only looks at which attributes are dependent on each other, but can also differentiate between different levels of dependencies, which results in a less complex tree.

Minimum value distribution

The minimum value distribution (MVD) is the third that is used to optimise the decision tree using the RBDT-1 algorithm. It is only considered if there are more than one attribute with both the same AE and AA score. The MVD is dependent solely on the cardinality of the set of values for an attribute V_{ij}. If an attribute has fewer potential values, then this will result in fewer branches and consequently the tree's depth and width will be reduced. For a set of attributes $A_1, \dots, A_q$, where $2 \leq q \leq s$, the MVD value is given by equation (10), where $|X|$ denotes the cardinality of set X. V_{ij} corresponds to the set of values for attribute A_j for each decision class C_i – where m is the maximum number of classes.

$$MVD(A_j) = \left| \bigcup_{1 \le i \le m} V_{ij} \right| \quad (10)$$

Where attributes have the same AE, AA and MVD scores, the order in which they are evaluated in a decision tree can be randomly assigned.

Attribute hierarchy

Once the order of the attributes has been determined by the AE, AA and MVD criteria, any additional optimisations can be assessed using the attribute hierarchy (AH) criterion. This can be defined as the number of subsequent attributes that can be excluded owing to the attribute in question A_j taking a specific value v_i. This is calculated multiple times for each attribute, once for each of the potential values that the attribute can take, not including '$*$'. The exclusions are included in the structure of the decision tree, but are not prioritised over the AE, AA or MVD values when determining the sequencing.

Before calculating the AH score, attribute exclusion notation needs to be introduced. Previously, if an attribute did not affect the classification outcome within a given rule then it was given the notation '*'. For the attribute exclusion score, if any subsequent attribute corresponding to any of the classification rules does not affect the conformity outcome then it will be given the following notation: '**'. To calculate the AH for each of the attributes, the exclusion scores will first need to be calculated for all attributes A_j, where $1 \le j \le n$. When attribute A_j takes a value v_i, V_{ik} is the set of values for attribute A_k, where $1 \le k \le n$ and $k \ne j$ within the decision class C_i, where $1 \le i \le m$. The exclusion score is calculated for each of the potential values of A_j using equation (11).

$$A_{exc}(A_j, v_i) = \begin{cases} 1 & if\ ** \in V_{ik} \\ 0 & otherwise \end{cases} \quad (11)$$

The number of times the A_{exc} score is calculated for each attribute is d_j, where d_j is the cardinality of the set of values V_{ij} for the attribute A_j. The attribute hierarchy is calculated by summing across the A_{exc} score for each attribute, as shown in equation (12).

$$AH(A_j) = \sum_{i=1}^{d_j} A_{exc}(A_j, v_i) \quad (12)$$

For attributes with an AH score greater than zero, subsequent attributes in the decision tree with a value "**" can be excluded in the decision pathway where attribute A_j takes a value v_i.

```
Input: A set of rules
Output: A decision tree
Initialisation: RR ← the whole set of rules (TR); B ← ∅; A ← the whole set of attributes in RR
Begin
do
    if (rules in RR belong to the same decision class) then Create a leaf node with the value of the decision class in
    RR
    elseif (A = ∅) then Create a leaf node with the value of the majority decision class
    else
        Apply AE criterion on the attributes in A
        singleton ← true
        if (the fit attributes produced by AE criterion are more than one) then
            Calculate AA criterion for attributes in A
            if (the fit attributes produced by the AA criterion are more than one) then
                Calculate MVD criterion for the attributes in A
                if (the fit attributes produced by MVD criterion are more than one) then
                    Calculate AH criterion for the attributes in A
                    if (the fit attributes produced by AH criterion are more than one) then
                        Create a node with an attribute a for one of the attributes selected
                        randomly
                        singleton ← false
                    endif
                endif
            endif
        endif
        if (singleton = true) then Create a node with the fit attribute a and update A: A ← A – {a}
        endif
    Create a new branch b for each value of the fit attribute a
    Assign B ← B ∪ B_new where B_new is the set of new branches stemming from the fit attribute
    endif
    if B = ∅ then
        Select a branch b from B and update B: B ← B – {b}
        RR ← set of rules corresponding to b
    endif
while (B ≠ ∅)
Return the tree consisting of the nodes a and branches b
end
```

Figure S1. pseudocode for the decision tree-building process employed by modified RBDT-1C; based on the pseudocode developed by Abdelhalim et al. (2014) [1]

References

[1] A. Abdelhalim, I. Traore, and Y. Nakkabi, "Creating Decision Trees from Rules using RBDT-1," Computational Intelligence, vol. 32, no. 2, pp. 216–239, 2014.

[2] R. S. Michalski and I. F. Imam, "Learning problem-oriented decision structures from decision rules: The AQDT-2 system," Lecture Notes in Computer Science Methodologies for Intelligent Systems, pp. 416–426, 1994.