Arbogodaï, a New Approach for Decision Trees

Djamel A. Zighed[1], Gilbert Ritschard[2], Walid Erray[1], and Vasile-Marian Scuturici[1]

[1] ERIC Laboratory, University of Lyon 2, C.P.11 F-69676 Bron Cedex, France
`zighed@univ-lyon2.fr`
[2] Dept of Econometrics, University of Geneva, CH-1211 Geneva 4, Switzerland
`ritschard@themes.unige.ch`

Abstract. Decision tree methods generally suppose that the number of categories of the attribute to be predicted is fixed. Breiman et al., with their Twoing criterion in CART, considered gathering the categories of the predicted attribute into two superclasses. In this paper, we propose an extension of this method. We try to merge the categories in an optimal unspecified number of superclasses. Our method, called *Arbogodaï*, allows during tree growing to group categories of the target variable as well as categories of the predictive attributes. At the end, the user can chose to generate either a set of single rules or or a set of multi-conclusion rules that provide interval like predictions.

1 Introduction

Induction trees are among the most popular supervised methods proposed in the literature. They are appreciated for the simplicity and the high efficacy of the algorithms, for their ease of use and for the easily interpretable results they provide. Hastie et al. [3], p. 313, designate them as the learning tool that comes closest to the requirements of an "off-the-shelf" method.

Many induction trees methods have been proposed so far in the literature. Some like ID3 [6], C4.5 [7] and CHAID [4, 5] build n-ary trees, others like CART [2] produce binary trees or, like SIPINA and Branching Programs [11, 12], latticed graphs that generalize trees by allowing the merging of nodes.

All these methods were originally intended for categorical attributes and require therefore that quantitative variables be discretized. This discretization can be done at once before growing the tree. Most of the tree growing methods, nevertheless, handle quantitative variables in an automatic manner by dynamically choosing the optimal discretization thresholds at each node. Some methods also attempt to reduce the number of categories of nominal attributes by partitioning them into a smaller number of classes. CART, for example, merges the categories into two new superclasses at each new split. This has the advantage of avoiding to uselessly increase the number of nodes. Indeed, the higher the number of nodes, the greater are the chances that some of them will have too few cases to get reliable estimates of the response classes probabilities.

There are two main ways for partitioning the values of the predictive attributes. The first is for instance a characteristic feature of CHAID [4]. At each node, the local discriminating power of each categorical attribute is tested using all possible partitions of its values. Partitions in two or more groups are explored. Thus, for each split, a predictor is selected simultaneously with its locally best partition.

The second strategy is used for instance by Breiman et al. [2] in their CART method. At each node, CART looks only for the best bi-partition of each predictor. It generates thus only binary trees. With their Twoing criterion, the authors of CART propose however also a strategy that extends their principle to the response variable. When the response is multi-valued, using Twoing is equivalent to seek, for every predictor, simultaneously the best bi-partition of its values and the best bi-partition of the response values. The Twoing is the value of the Gini impurity for the best couple of bi-partitions and is used for selecting the split variable at each node.

In this paper, we extend the principle of a simultaneous search of a double bi-partition. We combine the CHAID and CART approaches. Like CHAID we look at each step for the best not necessarily binary partition of the attributes. Like CART with Twoing we explore also the partitioning of the values of the target variable. Unlike CART, we do not, however, restrict ourself to bi-partitions. At each step we look for the simultaneous grouping of the predictor values and of the target variable values that optimizes the chosen criterion. We make use here of results given in [9, 10]. This gives rise to a new induction tree method that we call *Arbogodaï*. This kind of tree is characterized by a number of value classes of the target variable that varies from one node to the other. It is dynamically determined at each new split. When the majority class in a leaf contains several response values, the corresponding prediction rule becomes a multiple conclusion rule. For instance, we can get a rule like "a female customer aged between 30 and 40 with a monthly income ranging form 4000 to 5000 euros will chose a red or blue car". Indeed, we can easily compute which of the two colors is more frequent in the leaf. Hence, we can also derive classical simple rules. With *Arbogodaï*, the user has the possibility to chose the kind of rule that best suits her needs.

The paper is organized as follows. Section 2 introduces the notations and recalls the basic induction tree concepts. Section 3 discusses the optimal reduction of the crosstable that crosses at each node the target variable with the predictive attribute. Then Section 4 introduces the *Arbogodaï* algorithm that looks for such an optimal reduction when testing the attributes at each new node. In section 5, we specify the mutiple conclusion nature of the induced rules and propose adapted error rates for *Arbogodaï* trees. We report also experimentations that attempt to compare the generalization performances of *Arbogodaï* with other tree algorithms. Finally, in Section 6 we make some concluding remarks.

2 Principle of Induction Trees and Notations

Let Ω be the population concerned by the learning problem. The profile of any member ω of Ω is described by p variables, $X_1, \ldots, X_p$, called either exogenous variables, predictive attributes or predictors. These variables can be qualitative or quantitative. The set of values taken by X_j is denoted by $\mathcal{X}_j$. We consider also a target attribute C, sometimes called response, endogenous or dependent variable, and designate by $\mathcal{C}$ the set of response values. Like the X_j's, C can be qualitative or quantitative. Since the attributes X_j and the target variable C take only a finite number of different values in a given dataset, the sets $\mathcal{X}_j$ and $\mathcal{C}$ are finite. We denote by m_j the number of different values taken by the attribute X_j and by ℓ the number of different response values c_i. Thus, $\mathcal{C} = \{c_1, \ldots, c_\ell\}$. The goal of induction trees is then to generate a model $\phi(X_1, \ldots, X_p)$ in the form of a decision tree for predicting the value of C from the knowledge of the values taken by the predictive attributes. The tree ϕ is induced from a training sample $\Omega_L \subset \Omega$.

The growing process of the tree is quite simple. As illustrated in Figure 1, the set Ω_L is iteratively split by means of, at each step, one of the predictive attributes $X_1, \ldots, X_p$. The goal is to get distributions among the values of the target variable that are as different as possible. The leaves of the trees obtained at each step t of the growing process define a partition S_t of Ω_L that becomes finer and finer with t. The root of the tree corresponds to the trivial partition $S_0 = \{\Omega_L\}$. The tree given in Figure 1 partitions Ω_L in three subsets corresponding to the nodes s_2, s_3 and s_4. In leaf s_3 for example, we have the set of cases of Ω_L that take values $X_1 =$ male and $X_2 < 5000$. At step t, the partition S_t is derived from the previous one S_{t-1} by seeking the best leaf-attribute couple (s_k, X_j), i.e. that for which the splitting of $s_k \in S_{t-1}$ according to the values of X_j maximizes the gain of information on the target variable between S_{t-1} and S_t. The gain of information is usually measured as the reduction in uncertainty for the target variable or as the increase in the strength of association between the partition and the target variable. The growing process stops when the criterion can no longer be improved or when some stopping criterion is reached.

Let n be the grand total of cases, n_{ik} the number of cases with value c_i for the target variable in the class (leaf) s_k of the partition S, $n_{.k}$ the total number of cases in leaf s_k, $n_{i.}$ the total number of cases with value c_i. The corresponding observed frequencies are denoted respectively by f_{ik}, $f_{.k}$ and $f_{i.}$, and $f_{i|k} = n_{ik}/n_{.k}$ stands for the conditional frequency of value c_i in the leaf s_k.

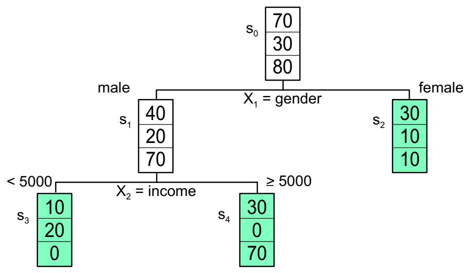

Fig. 1. An induced tree

Table 1. Contingency table defined by X_j at a node s

	x_{j1}	$\ldots$	x_{jk}	$\ldots$	x_{jm_j}	Total
c_1	n_{11}	$\ldots$	n_{1k}	$\ldots$	n_{1m_j}	$n_{1.}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$		$\vdots$	$\vdots$
c_i	n_{i1}	$\ldots$	n_{ik}	$\ldots$	n_{im_j}	$n_{i.}$
$\vdots$	$\vdots$		$\vdots$	$\ddots$	$\vdots$	$\vdots$
c_ℓ	$n_{\ell 1}$	$\ldots$	$n_{\ell k}$	$\ldots$	$n_{\ell m_j}$	$n_{\ell .}$
Total	$n_{.1}$	$\ldots$	$n_{.k}$	$\ldots$	$n_{.m_j}$	n

At any node s of a tree, an attribute X_j defines a partition of the cases in s. This partition is described by the columns of the $\ell \times m_j$ contingency table (Table 1) that crosses the target variable (rows) with X_j (columns).

The criteria used to measure the gain of information brought by a split defined by X_j are computed from this table. For instance, some methods try to maximize the reduction in uncertainty as measured by entropies. In this case, the uncertainty after the split is defined as the weighted mean of the uncertainty of the columns of the contingency Table 1

$$I(S) = \sum_{k=1}^{m_j} \frac{n_{.k}}{n} h(f_{1|k}; \ldots ; f_{i|k}; \ldots ; f_{\ell|k}) \tag{1}$$

where $h()$ is, for example, the Shannon entropy, $-\sum_{i=1}^{\ell} f_{i|k} \log_2 f_{i|k}$, or the quadratic entropy, also known as the Gini diversity index, $\sum_{i=1}^{\ell} f_{i|k}(1 - f_{i|k})$. Alternatively, some methods like CHAID, optimize the strength or the statistical significance of the association between the resulting partition (columns of Table 1) and the target variable (rows of Table 1).

3 Optimal Reduction of a Contingency Table

Let us recall that CHAID tries, at each step, to merge the columns of crosstables like Table 1 to find the best grouping of values for each candidate attribute, i.e. the grouping that optimizes the criterion. CHAID makes no change, however, on the values of the target variable. *Arbogodaï*, like the Twoing approach in CART, considers merging both columns and rows. Unlike the Twoing rule that looks for the best solution among 2×2 tables only, we seek however the best cross-partition without constraints on the number of rows and columns. This section discusses this issue. First, we motivate the approach by showing that the best n-ary split can sometimes be missed by successive binary splits. Then, we examine strategies for determining the best simultaneous partition of the rows and the columns. We show that the search of the optimal solution becomes rapidly untractable when the number of values of the predictive and/or target variable exceeds 5 or 6. This leads us to use a heuristic to get a quasi-optimal solution.

3.1 Motivation of n-ary Partitions

Consider the crosstable of Table 2. The best bi-partition of its columns is $S_{\text{bin}} = \{\{a, b\}, \{d, e\}\}$, whether we use the Gini, the Twoing, the significance of Pearson's Chi-Squares or an association measure like the t of Tschuprow. Now, the best 3 way partition is $S_{\text{3way}} = \{\{a\}, \{b, d\}, \{e\}\}$ with any of the criteria except Twoing which is not applicable. Clearly S_{3way} cannot be obtained by splitting the classes of S_{bin}. This proves that multiple binary partitions are not equivalent to n-ary partitions and can sometime miss optimal solutions.

The merging of response values is different in nature from that of predictive attributes. Indeed, the partition of the response values does not translate into a split of the node. Considering such mergings in the optimization process merits therefore some further justification. This is given by simply extending the argument of Breiman et al. ([2], p. 105) who argue that searching for superclasses (the groups of the partitions of the response values) provides strategic information on the similarities of responses. When two or more responses, red car and blue car for example, are almost equally frequent it may be a better strategy to predict that the customer will buy a red or a blue car than explicitly a red one. Simultaneously, it may be useful to know that yellow and pink colors are much less improbable than all other non red and non blue proposed colors. There is thus no reason to limit the argument to two superclasses only. Multi-superclasses provide a more refined strategic information.

3.2 Optimal Reduction

Let $\mathbf{T}_s(C, X_j)$ denote the contingency table obtained by crossing the target variable C with the predictive attribute X_j at a given node s of the tree. From here on we shall drop the subscripts j and s when there is no ambiguity. Our objective is to seek the couple of partitions that produces the table $\mathbf{T}(C, X)$ with maximal row-column association θ. We write the association criterion as $\theta(\mathcal{C}, \mathcal{X})$ to make it clear that it varies with the partitioning of the values of C and X. Let us recall that $\mathcal{X}$ stands for the set of distinct values of X in the concerned population, and $\mathcal{C}$ for the set of distinct values of the target variable C.

We have to distinguish between ordered and unordered sets $\mathcal{X}$ and $\mathcal{C}$. In the unordered case, i.e. for nominal attributes, we denote respectively by $\mathcal{P}_x$ and $\mathcal{P}_c$ the sets of possible partitions of $\mathcal{X}$ and $\mathcal{C}$. In the ordered case, i.e. for ordinal or quantitative attributes, only the merging of adjacent categories is allowed. We denote by $\mathcal{A}_x$ and $\mathcal{A}_c$ the sets of such allowed partitions.

Table 2. A n-ary solution different from that of successive binary splits

	a	b	d	e	Total
c_1	200	100	10	1	311
c_2	10	150	150	10	320
c_3	1	10	100	200	311
Total	211	260	260	211	942

For the unordered case, assuming the criterion has to be maximized, we seek the best couple of partitions in $\mathcal{P}_c \times \mathcal{P}_x$. We have to solve 5 $\arg\max \theta(\mathcal{C}, \mathcal{X})$ for $(\mathcal{C}, \mathcal{X}) \in \mathcal{P}_c \times \mathcal{P}_x$. Finding the optimal solution requires thus to scan $|\mathcal{P}_c||\mathcal{P}_x|$ crosstables. For ordered variables, we replace $\mathcal{P}$ with $\mathcal{A}$.

The number of partitions $|\mathcal{P}|$ of a set of size m is given by Bell's formula, $B(m) = \sum_{k=0}^{m-1} \binom{m-1}{k} B(k)$ with $B(0) = 1$. In the ordinal case the number of partitions $|\mathcal{A}|$ is simply $\sum_{k=0}^{m-1} \binom{m-1}{k} = 2^{m-1}$. Thus, when both variables are nominal with $m = \ell = 10$ categories, we would have to test more than 13 billions of tables. This is indeed intractable.

3.3 Reduction Heuristic

To face the limitations mentioned above, two ways can be considered. The first, is to seek the optimal solution among reasonably sized tables only, i.e. by limiting partitions to a maximum of say 5 or 6 classes. The second approach consists in using a heuristic that would provide a solution close to the true optimum. We have considered this last case in our work on the optimal aggregation of contingency tables [10] in which we studied an algorithm first introduced in [9]. Our experiences with the heuristic led us to two main conclusions. First, the heuristic provides solutions that are most of the time very close to the true optimum. Secondly, the optimal solution is very unstable. It depends indeed strongly on the sample considered. In a learning framework, where the learned rules are intended to be applied outside the learning sample, it is then not crucial to know the exact learning optimum. A solution close to the optimum is largely sufficient. We decided therefore to adopt here the heuristic studied in [10] that we recall briefly.

The heuristic is a simple greedy algorithm. It iteratively merges two rows or two columns. At each step, it merges the couple of either rows or columns that provides the greatest improvement in the criterion. The algorithm reduces thus at each step the table by one row or one column. The process stops when any additional merging would deteriorate the criterion.

Let $\mathcal{C}^k$ and $\mathcal{X}^k$ be the partitions of the values of C and X after step k. For C for example, we denote by $\mathcal{P}_c^k$ the set of partitions that can be obtained in the nominal case by grouping two classes of $\mathcal{C}^k$. When $\mathcal{C}^k$ is an ordered set, we consider the set $\mathcal{A}_c^k$ of partitions that can be obtained by grouping adjacent classes of $\mathcal{C}^k$.

Assuming the criteria θ has to be maximized, the row-column configuration achieved after step k is, for the nominal case, the solution of

$$\begin{cases} \arg\max \theta(\mathcal{C}^k, \mathcal{X}^k) & & & \\ \text{s.t.} & \mathcal{C}^k = \mathcal{C}^{(k-1)} & \text{and} & \mathcal{X}^k \in \mathcal{P}_x^{(k-1)} \\ \quad\text{or} & \mathcal{C}^k \in \mathcal{P}_c^{(k-1)} & \text{and} & \mathcal{X}^k = \mathcal{X}^{(k-1)} \end{cases} \tag{2}$$

For ordinal variables, we replace $\mathcal{P}^{(k-1)}$ by $\mathcal{A}^{(k-1)}$.

The algorithm starts from the finest partitions $\mathcal{X}^0$ and $\mathcal{C}^0$ of the values observed at the concerned node s. It seeks iteratively, the tables $\mathbf{T}_k(C, X)$

($k = 1, 2, \ldots$) corresponding to the double partition solution of problem (2). The procedure is repeated as long as we have $\theta(\mathcal{C}^k, \mathcal{X}^k) \geq \theta(\mathcal{C}^{(k-1)}, \mathcal{X}^{(k-1)})$.

4 Arbogodaï Trees

We first explain the principle of the *Arbogodaï* algorithm and, then, describe how it works on an example.

4.1 Principle of the Algorithm

Arbogodaï follows the general principle of tree growing presented in Section 2. Its specificity is an additional preparatory step before testing the attributes at a node. This step consists in optimally reducing the size of the table that crosses the target variable with every attribute. The splitting criterion is then computed using the found partitions of both the attribute and the target variable values. The splitting of the selected node is done according to the found classes of values of the selected predictive attribute.

This additional step plays a role similar to discretization. The merging of values can indeed be assimilated to some sort of discretization that works also on nominal variables. Remember, however, that the merging is done here simultaneously at each step on the target and the predictive attribute.

The reduction of the table is that for which the row-column association θ is maximized. Indeed we use the heuristic of Section 3.3 and measure the association θ with the t of Tschuprow: $t = \{n^{-1}[(\ell-1)(m-1)]^{-1/2}\chi^2\}^{1/2}$, where $\chi^2 = \sum_{i=1}^{\ell}\sum_{k=1}^{m}(nn_{i.}n_{.k})^{-1}(nn_{ik} - n_{i.}n_{.k})^2$ is the Pearson Chi-Squares statistic. Unlike some other association measures, the t of Tschuprow may increase with the merging of either rows or columns (see [10].)

The splitting criterion is the reduction in uncertainty (gain in purity) achieved with the columns of the reduced table as compared to its margin. The uncertainty after the split is computed for every X_j by applying formula (1) on the optimal reduced table for X_j at the considered node s. Using the * to denote quantities derived from the reduced table, the gain in uncertainty reads, with the quadratic (Gini) entropy:

$$h(C^*) - h(C^*|X_j^*) = \sum_i \big[\big(\sum_k f_{.k}^* f_{i|k}^{*2}\big) - f_{i.}^{*2}\big] \tag{3}$$

In addition, we use Laplace estimates for the proportions, i.e. the f^*'s are computed by adding a constant λ to each cell of the reduced table and of its margins. This penalizes the gain of uncertainty obtained at nodes with small counts. With very small counts, i.e. when λ represents a significant proportion of the count, a split may even deteriorate the uncertainty criterion (see [12] p.76.)

4.2 Example

We now describe the *Arbogodaï* algorithm through an example. We consider the *Flags* dataset from the UCI repository [1]. The response variable C takes 6

Table 3. Step 1 optimal crosstable and Laplace estimates of column distributions

C / X_7	$\{c,d,e,h\}$	$\{a,b,g\}$	$\{f\}$
$\{c_1\}$	33	6	0
$\{c_2,c_4,c_5,c_6\}$	2	100	1
$\{c_3\}$	17	9	26
Total	52	115	27

$f^*_{i\|k}$	$\{c,d,e,h\}$	$\{a,b,g\}$	$\{f\}$	$f^*_{i.}$
$\{c_1\}$	0.618	0.059	0.033	0.203
$\{c_2,c_4,c_5,c_6\}$	0.055	0.856	0.067	0.528
$\{c_3\}$	0.327	0.085	0.900	0.269
$f^*_{.k}$	0.271	0.581	0.148	1

nominal values $\mathcal{C} = \{c_1, c_2, c_3, c_3, c_4, c_5, c_6\}$ and there are 29 mixed categorical and quantitative predictive attributes $X_1, \ldots, X_{29}$. The dataset contains 194 cases. Figure 2 shows an extract of the two first levels of the *Arbogodaï* tree for these data.

Step 1. At the root of the tree, we have the distribution of all 194 cases among the 6 values of the response C. The 29 predictive attributes are successively tested. For every attribute, we first determine the optimal reduced crosstable with the target variable. We then select the attribute for which the gain in uncertainty computed on the reduced table is maximal. The winner is X_7, which takes 8 values: $\mathcal{X}_7 = \{a, b, c, d, e, f, g, h\}$. The two simultaneous groupings found by the heuristic of Section 3.3 are $\mathcal{X}_7^* = \{\{c, d, e, h\}; \{a, b, g\}; \{f\}\}$ and $\mathcal{C}^* = \{\{c_1\}; \{c_2, c_4, c_5, c_6\}; \{c_3\}\}$. The corresponding crosstable is shown in Table 3 together with the table of the derived conditional frequencies $f^*_{i|k}$. The latter have been computed by setting $\lambda = 1$. The marginal uncertainty is $h(C^*) = 1 - .20^2 - .53^2 - .27^2 = .61$ and the uncertainty after the split, which is the weighted average of the uncertainty of each column, is $h(C^*|X_3^*) = .31$. The gained information is thus .3. This is the maximal value achievable with any of the 29 attributes.

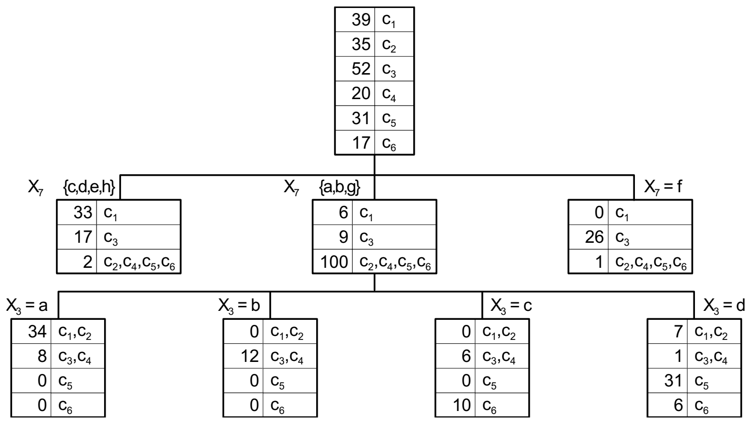

Fig. 2. Example of an Arbogodaï tree

Table 4. Crosstable for splitting the two leftmost leaves with X_{29}

	$\{a, b, d, e\}$	$\{f\}$
$\{c_1, c_2, c_4\}$	38	1
c_3	0	3
other	0	0

	$\{a, d, e\}$	$\{b, f\}$
c_3	2	2
c_4	8	0
other	0	0

Step 2. The process is repeated on every terminal node of the previously obtained tree. Notice that we try to merge the original set of values $\mathcal{X}$ and $\mathcal{C}$ and not the set of previously merged classes. In our example, the next best split occurs at the middle node ($X_7 \in \{a, b, g\}$). The attribute selected for splitting this node is X_3. The 6 values of the target C were merged to form 4 target classes. However, no merging of the attribute could improve the association between X_3 and the target C. The node is therefore split in 4 new classes corresponding to the 4 values of X_3. This leads to the tree with 6 leaves shown in Figure 2.

Following steps. In our example, the tree growing process is stopped after step 2. Without explicit stopping rules, the growing continues until the criterion can no longer be improved. At step 3, *Arbogodaï* would scan the 6 leaves of the previously grown tree.

Two further remarks should be made: (i) At a same level, nodes that do not result from a same parent may have different partitions of the set $\mathcal{C}$ of response values. (ii) When the same attribute is used as the splitting variable at more than one node, its values are not necessarily partitioned the same way for each split. For example, growing the tree of Figure 2 one level further leads to split each of the two left most leaves of level 2 with the same attribute X_{29}. The corresponding crosstables are given in Table 4. It can be seen that the values of C are once partitioned as $\{\{c_1, c_2, c_4\}, \{c_3\}, \{c_5, c_6\}\}$ and once as $\{\{c_3\}, \{c_4\}, \{c_1, c_2, c_5, c_6\}\}$. Likewise, attribute X_{29} is used once with the partition $\{\{a, b, d, e\}, \{f\}\}$ and once with $\{\{a, e, d\}, \{b, f\}\}$.

5 Induces Rules and their Accuracy

Arbogodaï can generate two types of classification rules: (i) Classical rules by disregarding the merged classes of response values in the final leaves. (ii) Multiple conclusion rules for leaves with merged response values. This Section specifies the nature of these rules, defines error rates adapted for them and presents experimentation results.

We give hereafter the multiple conclusion rules generated by the tree of Figure 2. Each path joining the root to a leaf defines the premise of a rule. The conclusion is drawn from the distribution in the leaf, i.e. the rule predicts for cases falling in the leaf the modal value in the leaf, or modal class of values when some are merged. The tree has 6 leaves giving rise to the 6 following rules (the value between parentheses is the confidence of the rule for the training data). Clearly, when the majority class contains only one value we get classical rules.

Here, only R_3 and R_4 provide multiple conclusions in the form of "either c_1 or c_2."

R_1 : **If** $X_7 \in \{c, h, e, d\}$ **then** $C = c_1$ (33/52)
R_2 : **If** $X_7 = f$ **then** $C = c_3$ (26/27)
R_3 : **If** $X_7 \in \{a, b, g\}$ **and** $X_3 = a$ **then** $C \in \{c_1, c_2\}$ (34/42)
R_4 : **If** $X_7 \in \{a, b, g\}$ **and** $X_3 = b$ **then** $C \in \{c_3, c_4\}$ (12/12)
R_5 : **If** $X_7 \in \{a, b, g\}$ **and** $X_3 = c$ **then** $C = c_6$ (10/16)
R_6 : **If** $X_7 \in \{a, b, g\}$ **and** $X_3 = d$ **then** $C = c_5$ (31/45)

5.1 Error Rates

The accuracy of the learned rules is usually assessed with the misclassification error rate or equivalently with the classification success rate. For classical rules, the misclassification rate reads $err = 1 - \sum_{s \in S} f_s f_{\max|s}$ where f_s is the proportion of cases in leaf s and $f_{\max|s} = \max_i f_{i|s}$ is the frequency of the modal response in leaf s.

For multiple conclusion rules, two kinds of error rates can be defined:

$$\text{superclass error} \qquad serr = 1 - \sum_{s \in S} f_s f^*_{\max|s}$$

$$\text{weighted superclass error} \qquad werr = 1 - \sum_{s \in S} f_s f^*_{\max|s} \big(\sum_{i \in \mathcal{C}_{\max,s}} \hat{p}_{i|\max,s} \; f_{i|\max,s} \big)$$

where $\mathcal{C}_{\max,s}$ is the set of response values in the modal superclass at leaf s, $f_{i|\max,s}$ the frequency of response c_i in that superclass and $\hat{p}_{i|\max,s}$ an estimation of the probability of c_i in the superclass. We get resubstitution error rates when the frequencies are those of the learning sample and generalization error rates when the frequencies are obtained from validation data. The estimations $\hat{p}_{i|\max,s}$'s are in any case computed on the training data. To get more reliable estimates, we use the marginal distribution at the parent node. This can be justified as follows. Values are merged when their distributions among the values of the split attribute are similar. Hence, their distributions inside the superclass are similar too and therefore similar to the marginal distribution.

The *superclass error*, $serr$, is computed as for classical rules but with the superclass frequencies $f^*_{i|s}$ instead of the single response frequencies $f_{i|s}$. Doing so, we do not care indeed of classification error inside the modal superclasses. This may have sense independently for each rule. We cannot compare, however, the error rate of a rule that predicts for instance c_1 with that of a rule that predicts c_1 or c_2. Hence, the global superclass error does not make much sense.

The *weighted superclass error*, $werr$, takes the uncertainty inside the majority class into account. It assumes that each case falling in a leaf is randomly assigned to a value in the modal superclass. The supposed random assignment is done according to the learned distribution inside $\mathcal{C}_{\max,f}$. For instance for our example tree, a case $(X_7 = a, X_3 = a, C = c_2)$ is correctly classified in the modal superclass of leaf 3. In that leaf, the estimated proportion of cases taking $C = c_2$

in the superclass is 85%. Thus, we weight this correct classification and count it as a .85 correct classification. In resubstitution, if we use $\hat{p}_{i|\max,s} = f_{i|\max,s}$, this is equivalent to weighting down the success rates with the Gini uncertainty of the distribution inside the superclass: $werr = \sum_s [1 - (1 - serr_s)\text{Gini}(C_{\max,s})]$, where $serr_s$ is the superclass error for rule s.

It is well known that the learning error rate suffers from an optimistic bias. It underestimates the generalization error rate. For validation, it is then common to compute the classification error rate on a separate dataset not used for learning. Alternatively, and perhaps more frequently, a cross-validation error rate is computed. A 10 folds cross-validation (10CV), for instance, consists in splitting the learning sample into 10 approximately equally sized parts. Dropping each time a different part we get 10 learning datasets from which 10 trees are induced. For each of them we compute the error rate on the dropped out data. The cross-validation error rate is the mean values of the 10 resulting error rates.

5.2 Experimentation

We have experimented our approach on 8 benchmark datasets. Table 5 gives the cross-validation success rates obtained for each dataset with *Arbogodaï* and, for the sake of comparison, with CART and CHAID. For *Arbogodaï*, we give the rate derived from both the classical and the weighted superclass error. *Arbogodaï* ranks first for 5 of the 8 datasets whatever error is considered. Unsurprisingly, its superiority is mostly significant when the number of values of the target variable is large.

Table 5. Cross-Validation classification success rates (in percents)

	CART		ChAID		Arbogodaï			
Dataset	$1-err$	stdev	$1-err$	stdev	$1-err$	stdev	$1-werr$	stdev
Iris (3 cl.)	95.11	0.08	94.81	0.08	98.35	0.11	95.50	0.08
Flags (6 cl.)	75.14	0.40	75.21	0.40	78.83	0.41	83.37	0.34
Breast (2 cl.)	97.54	0.17	97.19	0.15	98.17	0.13	98.08	0.17
Car (4 cl.)	83.47	0.32	93.62	0.23	86.75	0.32	87.81	0.31
Ionosphere (2 cl.)	92.10	0.19	89.68	0.20	89.34	0.3	93.36	0.25
Pima (2 cl.)	84.44	0.38	83.55	0.38	81.39	0.38	81.20	0.40
Wine (3 cl.)	97.71	0.19	97.99	0.19	98.09	0.07	95.21	0.20
Zoo (7 cl.)	87.57	0.22	85.99	0.26	88.61	0.12	94.04	0.16

6 Conclusion

To conclude, we would like to point out that the *Arbogodaï* method is well suited for mixed nominal and ordinal multi-valued attributes since the merging of any or only adjacent values can be set on the fly. It is also able to handle similarly nominal and ordinal, hence quantitative, target variables. Thus, *Arbogodaï* could be seen as some sort of regression tree. The originality is that, unlike for instance CART that generates point predictions for each leaf, *Arbogodaï* would provide interval predictions. The multi-conclusion of an *Arbogodaï* rule can hence be seen as a generalized interval for qualitative responses. Finally, let us mention that we are presently designing further experiments for comparing *Arbogodaï* with other tree methods and especially CHAID and CART. This aspect requires a careful investigation. Indeed, the parameterization of the trees (depth, pruning, stoping rules,...) plays a crucial role on the classification performance. We are trying, therefore, to set up rigorous conditions that would ensure more fair, hence more useful, comparison results. We also plan to investigate the relationship to the minimal description length (MDL) principle [8], as the optimally reduced tables can be seen as theories that best describe, locally at each node, the relevant knowledge about the relation between attributes and the target variable.

References

1. Blake, C., Merz, C.: UCI repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html (1998)
2. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: *Classification And Regression Trees.* Chapman and Hall, New York (1984)
3. Hastie, T., Tibshirani, R., Friedman, J.: *The Elements of Statistical Learning.* Springer, New York (2001)
4. Kass, G.V.: An exploratory technique for investigating large quantities of categorical data. *Applied Statistics* **29** (1980) 119–127
5. Morgan, J.N., Sonquist, J.A.: Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association* **58** (1963) 415–434
6. Quinlan, J.R.: Induction of decision trees. *Machine Learning* (1986) 81–106
7. Quinlan, J.R.: *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo (1993)
8. Rissanen, J.: A universal prior for integers and estimation by minimum description length. *The Annals of Statistics* **11** (1983) 416–431
9. Ritschard, G., Nicoloyannis, N.: Aggregation and association in cross tables. In Zighed, Komorowski, Zytkow, eds.: *Principles of Data Mining and Knowledge Discovery.* Springer-Verlag, Berlin (2000) 593–598
10. Ritschard, G., Zighed, D.A., Nicoloyannis, N.: Maximisation de l'association par regroupement de lignes ou colonnes d'un tableau croisé. *Revue Mathématiques Sciences Humaines* **39** (2001) 81–97
11. Zighed, D.A., Auray, J.P., Duru, G.: *SIPINA : Méthode et logiciel.* Editions A. Lacassagne, Lyon (1992)
12. Zighed, D.A., Rakotomalala, R.: *Graphes d'induction: apprentissage et data mining.* Hermes Science Publications, Paris (2000)