A Bayes Evaluation Criterion for Decision Trees

Nicolas Voisine and Marc Boullé and Carine Hue

Abstract We present a new evaluation criterion for the induction of decision trees. We exploit a parameter-free Bayesian approach and propose an analytic formula for the evaluation of the posterior probability of a decision tree given the data. We thus transform the training problem into an optimization problem in the space of decision tree models, and search for the best tree, which is the maximum a posteriori (MAP) one. The optimization is performed using top-down heuristics with pre-pruning and post-pruning processes. Extensive experiments on 30 UCI datasets and on the 5 WCCI 2006 performance prediction challenge datasets show that our method obtains predictive performance similar to that of alternative state-of-the-art methods, with far simpler trees.

1 Introduction

Building decision trees from training data is a problem which has begun to be treated in 1963 by Morgan and Sonquist. The first method is a regression tree which predicts a numerical variable [13]. Following this seminal work, the decision trees and regression trees problem of building has long been popular in machine learning.

Decision tree is a predictive model of a categorical variable and regression tree is a predictive model of a numerical variable. We may refer to the overviews [12], [5, 14, 6] for more details about the main decision-tree methods.

Nicolas Voisine
Orange Labs, 2, avenue Pierre Marzin Lannion, France, e-mail: nicolas.voisine@orange-ftgroup.com

Marc Boullé
Orange Labs, 2, avenue Pierre Marzin Lannion, France e-mail: marc.boulle@orange-ftgroup.com

Carine Hue
GFI Informatique, 11, rue Louis de Broglie Lannion, France e-mail: chue@gfi.fr

A decision tree is a classifier expressed as a hierarchical partition of the learning space. The partitions are represented by connected nodes. A node having children is called internal node and is defined by a segmentation rule. Other nodes are called leaves and represent a decision process by assigning the majority class to each instance of the node. The two first referenced algorithms are CHAID [11] and ID3 [19]. However, the CART [5] and C4.5 [20] decision trees are the benchmark methods with the highest reported performance.

The induction of an optimal decision tree from a data set is NP-hard [15]. Thus, learning the optimal decision tree requires exhaustive search and is limited to very small data sets. As a result, heuristic methods are required to build decision trees. These methods could be divided into two groups: global and top-down. The last group has the academic preference and referenced decision trees use top-down heuristics.

There are two kinds of top-down decision trees. First ones are based on a pre-pruning procedure (cf. CHAID and ID3), partitioning at each level of the tree the training (sub) set into subsets according to a selected segmentation variable. The choice of the variable among all the variables is made according to a segmentation criterion which provides the best partition. The procedure starts at the root of the tree and stops at the terminal nodes (leaves) when the criterion can no more be improved. The choice of the variable, the number of segments and the definition of the segmentation characterize the process of segmentation. The segmentation for numerical variables is called discretization and the segmentation for categorical variables is called grouping. There is not usually a global criterion to optimize segmentation process; each node splitting is optimized regardless of the others. The main decision trees (ID3, CHAID, CART, and C4.5) exploit a criterion based on information theory or statistical decision theory for evaluating segmentation. For example, C4.5 uses the *gain ratio* measure based on entropy, CART uses the *Gini Index* based on information gain measure and CHAID uses the CHI-Squared statistic with a threshold to take the best decision. However, pre-pruning suffers from the horizon effect [5]. The issue of pre-pruning algorithms is to stop the development of nodes until the decision tree is sufficiently accurate and to limit overfitness. Since Breiman work, new algorithms based on a post-pruning (CART, C4.5) have been studied. The construction of decision trees by post-pruning consists of two steps. The first step is to build a tree by continuing the process of segmentation as deep as possible, even if the tree overfits the data. The second step prunes the decision tree by removing nodes which minimize a pruning criterion. Learning time is longer than in the case of pre-pruning algorithms, but the performance of the tree is improved (cf. C4.5). Some pruning criteria are based on the estimated error rate of classification (C4.5). Other pruning criteria are based on a validation set (CART). Both approaches need to define heuristic parameters. A third, less-used approach exploits the principle of Minimum Description Length [18], [22].

Nowadays, decision trees are a mature class of models for which is just expected slight improvement of performance. Nevertheless, the reduction of the size of the trees and the automation of learning process are still important issues.

The decision tree performance mainly depends of the structure of trees. Too small trees obtain poor performance [5]. Too large trees overfit the training dataset and their performances collapse on the test dataset. Improving decision trees requires better segmentation criterion and pruning criterion. The post-pruning methods are likely to select noisy variables which are not pruned in the pre-pruning step. This might be frequent in the case of large numbers of variables. The main issue of building decision trees is to select the best variables, to segment them correctly and to decide when to stop. The reference methods (C4.5, CART, ID3 and CHAID) use several parameters to learn their optimal tree : parameters for the choice of variables, discretization of numerical variables, grouping of categorical variables, and settings of the pruning criterion. None of these methods offers comprehensive and consistent criterion, taking into account the structure of the tree, selection of variables, segmentation and the performance of the tree. Wallace and Patrick as a result of the work of Rivest and Quinlan use a MDL approach to define a global pruning criterion taking into account the tree structure and the distribution of the classes in the leaves [22]. Their lookahead algorithm pre-prunes decision trees by selecting sub-trees which minimize MDL criterion. This MDL criterion does not provide the best pruning, Quinlan and Rivest who had given the idea have not integrated in final C4.5 decision trees. MDL criteria have been exploited both as a selection criterion of segmentation variable and post-pruning criterion. However, MDL approach is a promising way with theoretical foundation to reduce the size of decision trees and to improve learning automation. But referenced MDL approaches remain incomplete, since they do not take into account all the trees parameters.

In this article, we propose a complete criterion by using a Bayesian approach according to a parsimony principle close to the Minimum Description Length approach. The aim is to transform the learning problem in a simple optimization process of one single parameter-free criterion. The MODL approach has already proved its interest in the selection of variables, the supervised discretization of numerical variables [3], grouping of categorical variables [2] and supervised classification model, with the Selective Naive Bayes [4]. Our goal is to develop a decision tree using the MODL approach, to evaluate and compare its performance with benchmark decision trees : J48(C4.5 [20]) and SimpleCART(CART [5]) from the WEKA software [9] which is an academic reference. The article is organized as follows. Section 2 summarizes the MODL approach in the case of supervised discretization. Section 3 describes the extension of this approach to decision trees. Section 4 describes optimization algorithms. Section 5 reports comparative evaluations of the method. Finally, section 6 concludes the article.

2 The MODL Approach

For the convenience of the reader, this Section summarizes the MODL approach in the case of supervised discretization of numerical variables [3].

The objective of supervised discretization is to induce a list of intervals which partitions the numerical domain of a continuous input variable, while keeping the information relative to the output variable. A trade-off must be found between information quality (homogeneous intervals in regard to the output variable) and statistical quality (sufficient sample size in every interval to ensure generalization).

In the MODL approach, the discretization is turned into a model selection problem. First, a space of discretization models is defined. The parameters of a specific discretization model are the number of intervals, the bounds of the intervals and the frequencies of the output values in each interval. Then, a prior distribution is proposed on this model space. This prior exploits the hierarchy of the parameters: the number of intervals is first chosen, then the bounds of the intervals and finally the frequencies of the output values. The prior is uniform at each stage of the hierarchy. Finally, we assume that the multinomial distributions of the output values in each interval are independent from each other. A Bayesian approach is applied to select the best discretization model, which is found by maximizing the probability $p(Model|Data)$ of the model given the data. Using the Bayes rule and since the probability $p(Data)$ is constant under varying the model, this is equivalent to maximizing $p(Model)p(Data|Model)$.

Let N be the number of instances, J the number of output values, I the number of input intervals. N_i denotes the number of instances in the interval i and N_{ij} the number of instances of output value j in the interval i. In the context of supervised classification, the number of instances N and the number of classes J are supposed to be known. A discretization model M is then defined by the parameter set $\left\{I, \{N_i\}_{1\leq i\leq I}, \{N_{ij}\}_{1\leq i\leq I, 1\leq j\leq J}\right\}$.

Using the definition of the model space and its prior distribution, Bayes formula can be used to calculate the exact prior probabilities of the models and the probability of the data given a model. Taking the negative log of the probabilities, this provides the evaluation criterion given in Formula 1.

$$\log N + \log \binom{N+I-1}{I-1} + \sum_{i=1}^{I} \log \binom{N_i+J-1}{J-1} + \sum_{i=1}^{I} \log \frac{N_i!}{N_{i1}!N_{i2}!\ldots N_{iJ}!} \qquad (1)$$

The first term of the criterion stands for the choice of the number of intervals and the second term for the choice of the bounds of the intervals. The third term corresponds to the parameters of the multinomial distribution of the output values in each interval and the last term represents the conditional likelihood of the data given the model, using a multinomial term. Therefore "complex" models with large numbers of intervals are penalized.

Once the evaluation criterion is established, the problem is to design a search algorithm in order to find a discretization model that minimizes the criterion. In [3], a standard greedy bottom-up heuristic is used to find a good discretization. In order to further improve the quality of the solution, the MODL algorithm performs post-optimizations based on hill-climbing search in the neighbourhood of a discretization. The neighbors of a discretization are defined with combinations of

interval splits and interval merges. Overall, the time complexity of the algorithm is $O(JN\log N)$.

The MODL discretization method for supervised classification provides the most probable discretization given the data. Extensive comparative experiments report high performance [3]. The case of value grouping of categorical variables is treated in [2] using a similar approach.

3 MODL Decision Trees

In this section, we apply the MODL approach to decision trees by defining explicitly a family of models and by introducing a global evaluation criterion of trees resulting from a Bayesian approach of model selection.

3.1 Definition

A decision tree is a classification model which aims at predicting a categorical output variable from a set of numerical or categorical input variables. One advantage of decision trees is that they provide understandable models, based on decision rules. The issue is to induce a tree with high predictive performance while keeping its size as small as possible. This turns into a difficult problem of finding a trade-off between the performance of the model and the complexity of the structure of the tree, in order to ensure a good generalization of the model.

The MODL approach for decision trees consists in selecting the model with the highest probability given the data from a family of decision trees. As for the case of discretization (cf. Section 2), we apply a Bayesian approach to select the decision tree with the highest posterior probability $p(Tree|Data)$, which is equivalent to maximize:

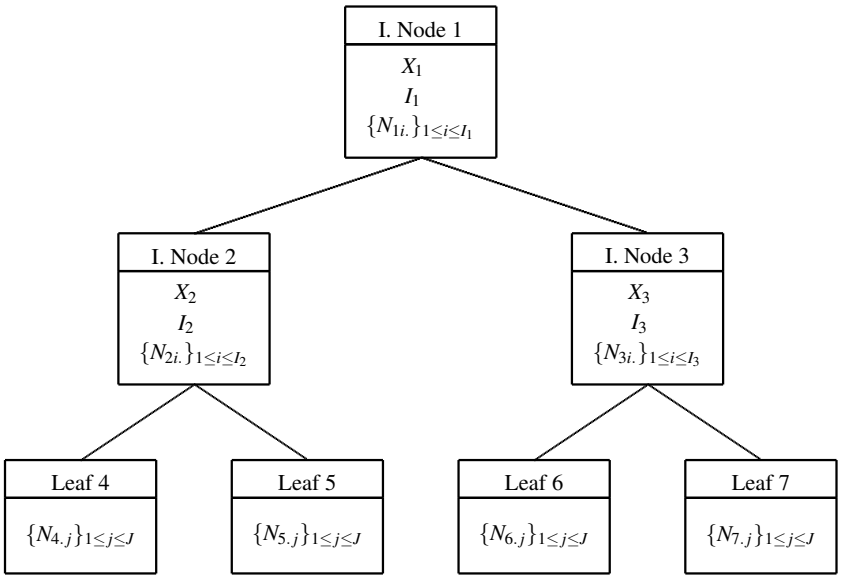
$$p(Tree)p(Data|Tree)$$

where $p(Tree)$ is the prior probability of the tree and
$p(Data|Tree)$ is the likelihood of the data given the model.

Let us introduce the following notations:

- N : number of instances,
- J : number of output values,
- T : a model of decision tree,
- $\mathbb{K}$: set of K input variables,
- $\mathbb{K}_T$: subset of K_T input variables used by tree T,
- $\mathbb{S}_T$: set of internal nodes of tree T,
- $\mathbb{L}_T$: set of terminal nodes (leaves) of tree T,
- X_s : segmentation variable of node s,
- $N_{s.}$: number of instances in node s,

Fig. 1 Example of decision tree. The internal nodes (I. Node) represent the decision rules and the leaves represent the distribution of the output values



- V_{X_s} : number of values of variable X_s in node s, in the categorical case,
- I_s : number of child nodes of node s,
- $N_{si.}$: number of instances in the i^{th} child of node s,
- $N_{l.j}$: number of instances of output value j in leaf l.

A decision tree model is defined by its structure, the distribution of the instances in this structure and the distribution of output values in the leaves (cf. Figure 1). The structure of the decision tree model consists of the set of internal nodes $\mathbb{S}_T$ (nodes with at least two children), the set of leaves and the relations between these nodes.

The distribution of the instances in this structure is defined by the partition of the segmentation variable in each internal node and by the distribution of the output values in each leaf. A decision tree model T is thus defined by:

- the subset of variables $\mathbb{K}_T$ used by model T, that is the number of selected variables K_T and the choice of the K_T variables among K,
- the number of child nodes I_s,
  - if $I_s = 1$ then the node is a leaf,
  - if $I_s > 1$ then the node is an internal node,
- the distribution of the instances in each internal node s:
  - the segmentation variable X_s,
  - the number of parts (intervals or groups of values) I_s,

  - the distribution of the instances on this parts (child nodes) $\{N_{si.}\}_{1\leq i\leq I_s}$,
- the distribution of the output values in each leaf l: $\left\{N_{l.j}\right\}_{1\leq j\leq J}$.

3.2 Evaluation Criterion

The evaluation criterion we propose here is the negative logarithm of the posterior tree probability given the data. As the data probability is constant whatever the model, the criterion is defined as

$$c(Tree) = -\log p(Tree)p(Data|Tree)$$

We choose the prior model probability $p(Tree)$ by exploiting the hierarchy of the modelization parameters. This hierarchy enables to describe dependence relationships between parameters. The prior choice comes from hierarchical extensions of the Bayesian approach. In the case of a complex parameter set, the uncertainty of high level parameters is expressed first, then, conditionally, the uncertainty on low level parameters. Bayes law enables us to express $p(Tree)$ according to a parsimony principle close to the Minimum Description Length approach and according to prior distributions for these parameters.

There are many ways to define such parameters hierarchy. The first would consist in defining the structure then the segmentations, then the class distribution for the leaves. In this article, we propose to exploit the implicit tree hierarchy by defining the model at the root level independently of its children. Then, in a recursive way, the nodes are described from the root children to the leaves. The prior probability of a MODL decision tree is thus defined as :

$$\begin{aligned} p(Tree) = p(\mathbb{K}_T)\times & \\ & \prod_{s\in\mathbb{S}_T} p(I_s)p(X_s|\mathbb{K}_T)p(N_{si.}|\mathbb{K}_T,X_s,N_{s.},I_s) \\ & \prod_{l\in\mathbb{L}_T} p(I_l)p(N_{l.j}|\mathbb{K}_T,N_{l.}) \end{aligned} \tag{2}$$

The first line in equation 2 represents the prior probability of variable selection. The second line is related to internal node probability and the last line represents leaf node probability.

Prior probability of variable selection

For the variable selection parameters, we reuse the prior introduced by [4] in the case of the selective naive Bayes classifier. We propose a hierarchic prior, by first choosing the number of selected variables and second choosing the subset of se-

lected variables. For the number K_T of variables, we propose to use a uniform prior between 0 and K variables, representing $(K+1)$ equiprobable alternatives. For the choice of the K_T variables, we assign the same probability to every subset of K_T variables. The number of combinations $\binom{K}{K_T}$ seems the natural way to compute this prior, but it has the disadvantage of being symmetric. Beyond $K/2$ variables, every new variable makes the selection more probable. Thus, adding irrelevant variables is favored, provided that this has an insignificant impact on the likelihood of the model. As we prefer simpler models, we propose to use the number of combinations with replacement $\binom{K+K_T-1}{K_T}$.It thus gives :

$$P(\mathbb{K}_T) = \frac{1}{K+1} \frac{1}{\binom{K+K_T-1}{K_T}}$$

Prior probability of an internal node

Knowing the selected variables and the parent nodes, the internal node can be defined by status (either internal node or leaf), segmentation parameters (the segmentation variable, the segmentation numbers and distribution of instances in segments. We consider that, for each internal node, the choice of the segmentation variable is independent and equal for all the selected explicative variables. To express the probability of the size of the segmentation of a given internal node, the simplest assumption of equiprobability leads to $p(I_s|K_T,X_s,N_{s.}) = \frac{1}{N_s}$ for a numerical variable and $p(I_s|K_T,X_s,N_{s.}) = \frac{1}{V_s}$ for a categorical variable. However, we obtained with such prior very cautious trees, as the higher the instances number, the lower the prior probability. That is why we propose here a prior inspired from the Minimum Description Length approach. Rissanen proposes an optimal coding of integers and gives the associated probability in [21]. This universal prior is defined so that the small integers are more probable than the large integers, and the rate of decay is taken to be as small as possible. According to Rissanen, this prior is "universal" because its resulting code length (negative log of the probability) realizes the shortest coding of large integers. This prior is attractive even in the case of finite sets of integers, because it makes small integers preferable to large integers with the slightest possible difference. The optimal length, in bits, of an integer I_s is :

$$C_{Ris}(I_s) = log_2(2.865) + log_2(I_s) + log_2(log_2(I_s)) + ...$$

We then obtain the universal prior probability of I_s segments :

$$p(I_s|K_T,X_s,N_{s.}) = 2^{-C_{Ris}(I_s)}$$

Moreover, by using the fact that an internal node has at least two children, the status of the node has not to be explicitly described. Only the number of segments, either one for a leaf or between 2 and N_s children for an internal node, are described.

For a numerical variable, the prior probability of the segmentation intervals is obtained similarly to the univariate MODL discretization (cf. section 2) :

$$\frac{1}{K_T} 2^{-C_{Ris}(I_s)} \frac{1}{\binom{N_s+I_s-1}{I_s-1}}$$

For a categorical variable, the prior probability is obtained similarly to the univariate MODL grouping method [2] :

$$\frac{1}{K_T} 2^{-C_{Ris}(I_s)} \frac{1}{B(V_{X_s}, I_s)}$$

$B(X,Y)$ is the number of divisions of the X values into Y groups (with eventually empty groups). When $X = Y$, $B(X,Y)$ is the Bell number. In the general case, B(X,Y) can be written as a sum of Stirling numbers of the second kind S(X,y) :

$$B(X,Y) = \sum_{y=1}^{Y} S(X,y)$$

$S(X,y)$ stands for the number of ways of partitioning a set of X elements into y nonempty sets.

Prior probability of a leaf

To end up, it remains to define the prior of leaves probability, that is to say the class distribution for each leaf. Assuming the distributions are equiprobable, it remains to calculate the number of multinomial distributions of $N_{l.}$ instances among J classes :

$$2^{-C_{Ris}(1)} \frac{1}{\binom{N_{l.}+J-1}{J-1}}$$

As internal nodes, the terms $2^{-C_{Ris}(1)}$ corresponds to the choice of the size of the segmentation, which is 1 for leaves.

Likelihood probability

We have now to explicit the likelihood probability of the data given the model. The data distribution depends only of tree leaves. Knowing the multinomial model defined on one leaf, we deduce the likelihood :

$$p(Data|Tree) = \prod_{l \in L} \frac{N_{l.}!}{N_{l.1}! N_{l.2}! \ldots N_{l.J}!}$$

MODL decision tree criterion

Endly, taking the negative logarithm of its posterior probability, the optimal tree cost is :

$$\begin{aligned} C_{opt}(T) = & \log(K+1) + \log\binom{K+K_T-1}{K_T} + \\ & + \sum_{s\in\mathbb{S}_{T_n}} \log K_T + C_{Ris}(I_s)\log 2 + \log\binom{N_{s.}+I_s-1}{I_s-1} + \\ & + \sum_{s\in\mathbb{S}_{T_c}} \log K_T + C_{Ris}(I_s)\log 2 + \log B(V_{X_s}, I_s) + \\ & + \sum_{l\in\mathbb{L}_T} C_{Ris}(1)\log 2 + \log\binom{N_{l.}+J-1}{J-1} + \\ & + \sum_{l\in\mathbb{L}_T} \log\frac{N_{l.}!}{N_{l.1}!N_{l.2}!\ldots N_{l.J}!} \end{aligned}$$

where $\mathbb{S}_{T_n}$ and $\mathbb{S}_{T_c}$ are the internal nodes sets using respectively numerical or categorical segmentation variable. It is noteworthy that, using Stirling's approximation, the last multinomial term of the formula is asymptotically equal to the target entropy in the leaves of the tree [7]. Thus, the whole crietrion clearly relates to an entropy-based impurity measure, with a penalization for complex trees.

4 Optimization Algorithms

The induction of an optimal decision tree from a data set is NP-hard [15]. The exhaustive search algorithm is then excluded. In this article we exploit a pre-pruning algorithm 1 and a post-pruning algorithm 2. The pre-pruning algorithm starts with the root node and searches the best partition according to the criterion presented above. The leaves are segmented while the criterion is improved. For each leaf, the partition is performed according to the univariate MODL discretization or grouping methods, then the global cost of the tree is updated by accounting for this new partition. The partition is really completed if the global cost is improved. The optimum is then searched with successive local optimums at leaf levels. This algorithm is close to those used in ID3 and CHAID decision trees. The difference lies in the fact that the segmentation of two leaves is not conducted independently as the criterion is global. One leaf node can not be segmented unless it is the best choice of segmentation. In practice, the additivity of the criterion enables to update only the cost of the considered node. The algorithm does not guarantee to find the global optimum but its maximal complexity is $\mathcal{O}(KJN^2Log(N))$, in the case of an unbalanced tree. Its is $\mathcal{O}(KJN(LogN)^2)$ on average in the case of a balanced tree. This algorithm is deterministic and thus it always leads to the same local optimum.

Algorithm 1 Top-Down algorithm with pre-pruning for optimal tree search

Require: T the root tree
Ensure: the tree $\hat{T}$ which minimizes the proposed criterion
  $T^* \leftarrow T$
  while criterion improvement **do**
    $\hat{T} \leftarrow T^*$
    for all leaf l of the tree **do**
      $T' \leftarrow T^*$
      for all variable X of $\mathbb{K}$ **do**
        Search the partition rule on the leaf l according X which best improves the criterion
        $T_X \leftarrow T^* + P_X(l)$
        if $c(T_X) < c(T')$ **then**
          $T' \leftarrow T_X$
        end if
      end for
      if $c(T') < c(T^*)$ **then**
        $T^* \leftarrow T'$
      end if
    end for
  end while

Unfortunately, the pre-pruning algorithm creates small and under-fitted decision trees. To go above this *horizon effect*, we have also exploited our criterion with a post-pruning algorithm [5]. The post-pruning algorithm consists in two steps. The first step is the top-down building of the deepest tree by choosing the best univariate MODL partitions for each leaf, even if it does not lead to an improvement in the global criterion. The tree with the minimum cost is memorized during this step. This step ends when there are no more MODL informative variables left. Starting from the obtained deepest tree, the second step considers only nodes consisting of leaves, and prunes the node which leads the best improvement of the criterion. Only the internal node whose children are all leaves are candidates for pruning. Like in the first step, the tree with the minimum cost is memorized. This algorithm is also deterministic and it always leads to the same local optimum. At least, this algorithm guaranties to find a decision tree with a better cost than the tree resulting of algorithm 1. This means that the posterior probability of the tree can only be improved using the post-pruning algorithm.

5 Experiments

This section presents an experimental evaluation of our supervised decision trees methods described in the previous sections.

Algorithm 2 Top-Down algorithm with post-pruning for optimal tree search

Require: T the root tree
Ensure: the tree $\hat{T}$ which minimizes the proposed criterion
  $T^* \leftarrow T$
  while there are MODL informative variables for one leaf node **do**
    $\hat{T} \leftarrow T^*$
    for all leaf l of the tree **do**
      $T' \leftarrow T^*$
      for all variable X of $\mathbb{K}$ **do**
        Search the partition rule on the leaf l according X which best improves the criterion
        $T_X \leftarrow T^* + P_X(l)$
        if $c(T_X) < c(T')$ **then**
          $T' \leftarrow T_X$
        end if
      end for
      $T^* \leftarrow T'$
    end for
  end while
  while the tree is not reduced to its root **do**
    $\hat{T} \leftarrow T^*$
    for all internal node s of the tree whose children are all leaves **do**
      $T_s \leftarrow T^* - children(s)$
      if $c(T_s) < c(T')$ **then**
        $T' \leftarrow T_s$
      end if
      $T^* \leftarrow T'$
    end for
  end while

5.1 Experiments Setup

We conduct the experiments on two collections of data sets: 30 data sets from the repository at University of California at Irvine [1] and 5 data sets from the WCCI 2006 performance prediction challenge [10]. These data sets represent a large diversity of number of variables, instances and classes, with numerical and/or categorical variables. A summary of some properties of these data sets is given in Table 1 for the UCI data sets and in Table 4 for the challenge data sets.

We evaluate two versions of the pre-pruning algorithm 1 and the post-pruning algorithm 2, with binary trees and N-ary trees. For binary trees, we constrain the univariate partition (discretization or grouping) of each node to build at most two subparts, related to two child nodes. On the opposite, internal nodes of N-ary trees can have more than two children. For more convenience, we call our decision tree family MTrees (MODL Trees). Our evaluated methods are:

- MTp : MTree with post-pruning top-down algorithm
- MTp(2) : MTree with post-pruning top-down algorithm and a binary tree structure.
- MT : MTree with pre-pruning top-down algorithm algorithm

Table 1 UCI Data Sets. The properties of the used UCI data sets are : number of instances, number of variables, number of classes and majority accuracy

Name	Variables	Instances	Classes	Majority Accuracy
Adult	15	48842	2	0.76
Australian	14	690	2	0.56
Breast	10	699	2	0.66
Crx	15	690	2	0.56
German	24	1000	2	0.70
Glass	9	214	6	0.36
Heart	13	270	2	0.56
Hepatitis	19	155	2	0.79
HorseColic	27	368	2	0.63
Hypothyroid	25	3163	2	0.95
Ionosphere	34	351	2	0.64
Iris	4	150	3	0.33
LED	7	1000	10	0.11
LED17	24	10000	10	0.11
Letter	16	20000	26	0.04
Mushroom	22	8416	2	0.53
PenDigits	16	10992	10	0.10
Pima	8	768	2	0.65
Satimage	36	6435	6	0.24
Segmentation	19	2310	7	0.14
SickEuthyroid	25	3163	2	0.91
Sonar	60	208	2	0.53
Spam	57	4307	2	0.65
Thyroid	21	7200	3	0.93
TicTacToe	9	958	2	0.65
Vehicle	18	846	4	0.26
Waveform	21	5000	3	0.34
WaveformNoise	40	5000	3	0.34
Wine	13	178	3	0.40
Yeast	9	1484	10	0.31

- MT(2) : MTree with pre-pruning top-down algorithm and a binary tree structure.
- NMT : Naive MODL tree is a top-down building of the deepest tree by choosing the best univariate MODL partition, without any pruning.

We compared our methods with J48 and SimpleCART which are implementations of C4.5 and CART in open-source data mining software WEKA [9]. We take as parameters those defined by default in the software. We evaluate the accuracy (ACC), the area under the ROC curve (AUC)[8], the number of nodes (internal nodes and leaves) and the training time. Provost et al. (1998) propose to use receiver operating characteristic (ROC) analysis rather than the accuracy to evaluate induction models [17]. The ACC criterion evaluates the accuracy of the prediction, no matter whether the conditional probability of the predicted class is 51% or 100%.

The AUC criterion evaluates the ranking of the class conditional probabilities. In a two-classes problem, the AUC is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. In our experiments, we use the approach of Provost and Domingos (2001) to calculate the multi-class AUC [16].

We collect and average the four criteria owing to a stratified 10-fold cross validation, for the seven evaluated methods on the thirty five data sets. In 10-fold cross-validation, the original data set is partitioned into 10 subsamples. Of the 10 subsamples, a single subsample is retained as the test data for testing the model, and the remaining 9 subsamples are used as training data. The cross-validation process is then repeated 10 times.

5.2 UCI Results

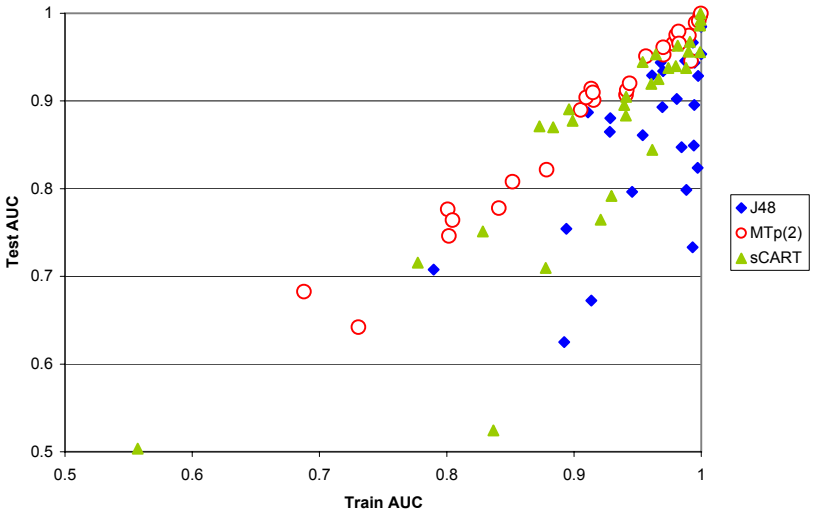
The geometric means of the four criteria for each method are summarized in Table 2. The great diversity of the data sets increases the variance of the criterion. Therefore we prefer to support our analysis on the geometric mean, which allows comparing the criterion ratios between the various methods. The full results are also reported in Table 3.

Table 2 Evaluation of the methods on UCI data sets : accuracy, AUC, size (number of nodes), training time and tree cost

Method	**Train data set**		**Test data set**		**Size**	**Time**	$\mathbf{C_{opt}(T)}$
	Acc.	**AUC**	**Acc.**	**AUC**			
MT(2)	0.845	0.914	0.819	0.889	17.5	0.5	524
MT	0.841	0.915	0.813	0.884	19.4	0.5	565
MTp(2)	0.840	0.910	0.822	0.891	17.4	0.6	508
MTp	0.834	0.905	0.817	0.890	19.5	0.6	547
NMT	0.879	0.959	0.762	0.844	142.3	0.8	1095
sCART	0.854	0.921	0.822	0.876	30.7	1.0	$\times$
J48	0.929	0.962	0.834	0.881	77.1	0.1	$\times$

Overall J48 obtains the best geometric mean of accuracy and MTp(2) obtains the best geometric mean of AUC. In most of the cases, AUC and accuracy results are close (cf. Table 3). These weak differences are not surprising, since the decision trees are a mature technology and the differences of performance are often marginal. On the other hand, the complexity of the tree structure is approximately four times less with MTree than with J48 and twice less than with SimpleCART. This property makes the interpretation of our trees considerably easier for the domain expert, and their deployment faster. Concerning training time, MTree is five times slower than J48 and twice faster than SimpleCART. Binary MTrees have better accuracy and AUC than N-ary MTrees. Constraining the algorithms to build binary trees leads to

Fig. 2 Train vs test AUC on 30 UCI data sets



a better optimization of the criterion and a better predictive performance with only a slight impact on the tree size. The test results show that the tree cost of unpruned MTrees (NMT) is twice that of pruned MTrees, while the test performances (Acc and AUC) are far worse. It is noteworthy that the criterion of binary MTrees is smaller than the one of N-ary MTrees. This shows that the performance (accuracy and AUC) of the MTree trees is clearly correlated with the value of the tree criterion. MTp(2) is slightly prone to overfitting but it overfits less the data than the other methods. Figure 2 clearly shows that the differences between train and test AUC on 30 UCI data sets are lower with MTree than with J48 or SimpleCART.

A detailed analysis of the results (cf. Table 3) shows that MTree accuracy are worse with data sets having correlated variables such as Letter or image segmentation. On the other hand for marketing data sets such as Adult, MTree is slightly better while having ten times less nodes than J48.

5.3 Prediction Challenge Results

This section reports the results obtained by our method on the performance prediction challenge of Guyon et al. [10]. The purpose of the performance prediction challenge is "*to stimulate research and reveal the state-of-the-art in model selection*". Five data sets are used in the challenge (cf. Table 4). The ada data set comes from the marketing domain, the gina data set from handwriting recognition, the hiva

Table 3 Test accuracy, AUC and tree size of post-pruned decision tree on UCI data sets, using ten-fold cross validation

	Accuracy				AUC				Tree Size			
Data Set	**MTp(2)**	**MTp**	**sCart**	**J48**	**MTp(2)**	**MTp**	**sCart**	**J48**	**MTp(2)**	**MTp**	**sCart**	**J48**
Adult	0.864	0.862	0.863	0.860	0.910	0.911	0.888	0.886	124.8	176.1	120.2	1099
Australian	0.852	0.852	0.857	0.852	0.904	0.903	0.878	0.881	6.2	6.2	5.8	46.2
Breast	0.937	0.957	0.949	0.946	0.965	0.969	0.948	0.948	9.6	8.7	15.8	23.4
Crx	0.861	0.861	0.852	0.861	0.914	0.914	0.866	0.894	7	7	3.6	27.1
German	0.692	0.692	0.750	0.739	0.682	0.682	0.722	0.692	3.2	3.2	19.4	140.6
Glass	0.597	0.649	0.705	0.659	0.778	0.811	0.848	0.793	7.8	8.2	20	47
Heart	0.733	0.719	0.785	0.767	0.808	0.810	0.792	0.755	7.2	7.7	14.2	33.8
Hepatitis	0.806	0.806	0.786	0.838	0.642	0.642	0.598	0.697	3	3	9.6	17.8
HorseColic	0.843	0.843	0.875	0.878	0.822	0.822	0.861	0.864	5.6	5.6	10	19.6
Hypothyroid	0.992	0.992	0.992	0.992	0.979	0.976	0.957	0.95	5.8	10.4	10.8	11.8
Ionosphere	0.898	0.889	0.898	0.915	0.901	0.908	0.896	0.895	5.2	7.9	8.8	27.4
Iris	0.953	0.933	0.953	0.960	0.975	0.963	1.000	0.990	5	4	8	8.4
LED	0.705	0.705	0.725	0.729	0.920	0.920	0.916	0.920	29	29	110	62.2
LED17	0.735	0.735	0.735	0.722	0.951	0.951	0.957	0.891	75.6	75.6	123.8	890
Letter	0.768	0.738	0.869	0.879	0.975	0.966	0.965	0.964	461.2	531.8	2091.2	2321.6
Mushroom	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	12.6	14	13.4	29.8
PenDigits	0.945	0.903	0.963	0.966	0.991	0.984	0.991	0.992	170.6	247	363.4	375.6
Pima	0.741	0.740	0.751	0.738	0.764	0.767	0.725	0.749	8.4	6.9	16.2	37.4
Satimage	0.853	0.852	0.868	0.873	0.972	0.971	0.981	0.979	76	72.3	165.2	551.4
Segmentation	0.938	0.938	0.958	0.971	0.989	0.989	0.998	0.998	31.2	46.1	76.6	82.6
SickEuthyroid	0.978	0.979	0.977	0.979	0.961	0.956	0.96	0.941	11.4	12.5	14	26.2
Sonar	0.735	0.735	0.712	0.712	0.746	0.746	0.722	0.735	4.8	4.8	14.2	29.2
Spam	0.912	0.911	0.922	0.935	0.953	0.955	0.94	0.943	44.8	54.1	131.2	192.2
Thyroid	0.995	0.992	0.996	0.997	0.997	0.996	0.996	0.99	14.4	23	22.4	30.6
TicTacToe	0.923	0.866	0.932	0.851	0.965	0.921	0.963	0.899	42.8	49.1	67.2	135.4
Vehicle	0.676	0.661	0.701	0.726	0.890	0.877	0.926	0.932	24.4	28.2	104.8	136
Waveform	0.756	0.745	0.777	0.759	0.912	0.904	0.903	0.845	72.4	90.4	136.6	541.8
WaveformNoise	0.744	0.751	0.767	0.751	0.907	0.906	0.903	0.847	70.4	84.4	121.4	580.4
Wine	0.917	0.917	0.894	0.939	0.946	0.951	0.963	0.975	8.6	7.4	9.2	9.8
Yeast	0.565	0.542	0.309	0.503	0.776	0.779	0.501	0.749	16.8	22.8	1.4	96.6
Ar. Mean	0.830	0.825	0.837	0.843	0.896	0.895	0.885	0.886	45.5	54.9	127.6	254.4
Geo. Mean	0.822	0.817	0.822	0.834	0.891	0.890	0.876	0.881	17.7	19.9	30.7	77.1
Mean Rank	2.6	2.8	2.3	1.9	2.0	2.2	2.6	2.8	1.3	1.8	2.7	4.0

data set from drug discovery, the nova data set from text classification and the sylva data set from ecology.

The detailed results of our evaluation are presented in Table 5. Unfortunely, we cannot report the results of simpleCART and J48 on all the data sets, since some of these data sets are too large given the Weka implementation of simpleCART and J48. Overall, MTree with post-pruning is the best method. It comes first on most of the data sets for the AUC, accuracy and tree size criteria.

Table 4 WCCI Challenge Data Sets. The properties of the used UCI data sets are : number of instances, number of variables, number of classes and majority accuracy

Name	Variables	Instances	Classes	Majority Accuracy
ada	48	4562	2	0.75
gina	970	3468	2	0.51
hiva	1617	4229	2	0.96
nova	16969	1929	2	0.72
sylva	216	14394	2	0.94

Table 5 Test accuracy, AUC and tree size of post-pruned decision tree on prediction challenge data sets, using ten-fold cross validation

	Accuracy				AUC				Tree Size			
Data Set	MTp(2)	MTp	sCart	J48	MTp(2)	MTp	sCart	J48	MTp(2)	MTp	sCart	J48
ada	0.847	0.847	0.842	0.846	0.887	0.890	0.860	0.860	22.0	23.6	28.1	224.0
gina	0.881	0.863	0.894	0.867	0.923	0.913	0.918	0.862	47.8	49.1	64.4	247.7
hiva	0.966	0.966	-	0.955	0.622	0.622	-	0.659	6.0	6.0	-	64.4
nova	0.866	0.866	-	-	0.817	0.817	-	-	17.6	17.6	-	-
sylva	0.989	0.989	0.991	0.990	0.991	0.991	0.981	0.954	26.2	41.4	41.0	105.2

6 Conclusion

The Bayesian criterion presented in this article gives a complete criterion to evaluate a decision tree by taking into account the structure of the tree, the choice of the explanatory variables, the segmentation in each internal node and the distributions of the classes in each leaf. This corresponds to an exact analytic evaluation of the posterior probability of a tree given the data and results in a parameter-free evaluation criterion. We have tested two optimization algorithms. The first one is a pre-pruning heuristic and the second one is a post-pruning heuristic which leads to a better optimization and obtain the better performance. Evaluations on 30 UCI data sets show that MTrees obtains state of the art performance while being much less complex. The evaluation on prediction challenge data sets show that our method gets the best results and builds the less complex decision trees. It is also noteworthy that binary trees are better on average than N-ary trees. Therefore designing new algorithms is a promising direction to get better performance. Another direction of research is to use MODL trees with random forest or Bayesian Model Averaging.

References

1. Blake, C., Merz, C.: UCI repository of machine learning databases (1996). Http://www.ics.uci.edu/mlearn/MLRepository.html

2. Boullé, M.: A Bayes optimal approach for partitioning the values of categorical attributes. Journal of Machine Learning Research **6**, 1431–1452 (2005)
3. Boullé, M.: MODL: a Bayes optimal discretization method for continuous attributes. Machine Learning **65**(1), 131–165 (2006)
4. Boullé, M.: Compression-based averaging of selective naive Bayes classifiers. Journal of Machine Learning Research **8**, 1659–1685 (2007)
5. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification And Regression Trees. Chapman and Hall, New York (1984)
6. Breslow, L.A., Aha, D.W.: Simplifying decision trees: A survey. Knowl. Eng. Rev. **12**(1), 1–40 (1997). DOI http://dx.doi.org/10.1017/S0269888997000015
7. Cover, T.M., Thomas, J.A.: Elements of Information Theory. John Wiley & sons (1991)
8. Fawcett, T.: ROC graphs: Notes and practical considerations for researchers. Tech. Rep. HPL-2003-4, HP Laboratories (2003)
9. Garner, S.R.: Weka: The waikato environment for knowledge analysis. In: In Proc. of the New Zealand Computer Science Research Students Conference, pp. 57–64 (1995)
10. Guyon, I., Saffari, A., Dror, G., Bumann, J.: Performance prediction challenge. In: International Joint Conference on Neural Networks, pp. 2958–2965 (2006). Http://www.modelselect.inf.ethz.ch/index.php
11. Kass, G.: An exploratory technique for investigating large quantities of categorical data. Applied Statistics **29**(2), 119–127 (1980)
12. Kohavi, R., Quinlan, R.: Decision tree discovery. In: in Handbook of Data Mining and Knowledge Discovery, pp. 267–276. University Press (2002)
13. Morgan, J., Sonquist, J.A.: Problems in the analysis of survey data, and a proposal. Journal of the American Statistical Association **58**, 415–435 (1963)
14. Murthy, S.K.: Automatic construction of decision trees from data: A multi-disciplinary survey. Data Mining and Knowledge Discovery **2**, 345–389 (1998)
15. Naumov, G.E.: Np-completeness of problems of construction of optimal decision trees. Soviet Physics **34**(4), 270–271 (1991)
16. Provost, F., Domingos, P.: Well-trained pets: Improving probability estimation trees. Tech. Rep. CeDER #IS-00-04, New York University (2001)
17. Provost, F., Fawcett, T., Kohavi, R.: The case against accuracy estimation for comparing induction algorithms. In: Proceedings of the Fifteenth International Conference on Machine Learning, pp. 445–553 (1998)
18. Quinlan, J., Rivest, R.: Inferring decision trees using the minimum description length principle. Inf. Comput. **80**(3), 227–248 (1989)
19. Quinlan, J.R.: Induction of decision trees. Machine Learning **1**, 81–106 (1986)
20. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo (1993)
21. Rissanen, J.: A universal prior for integers and estimation by minimum description length. Annals of Statistics **11**(2), 416–431 (1983)
22. Wallace, C., Patrick, J.: Coding decision trees. Machine Learning **11**, 7–22 (1993)