I N S T I T U T D E
S T A T I S T I Q U E

UNIVERSITÉ CATHOLIQUE DE LOUVAIN

D I S C U S S I O N
P A P E R

0124

HYPER-RECTANGULAR SPACE PARTITIONING CLASSIFICATION TREES : A FEW INSIGHTS

I. DE MACQ and L. SIMAR

http://www.stat.ucl.ac.be

HYPER-RECTANGULAR SPACE PARTITIONING CLASSIFICATION TREES : A FEW INSIGHTS

Isabelle De Macq and Léopold Simar[1]

June 2001

ABSTRACT. The process of computation of classification trees can be characterized as involving three basic choices: the type of splits considered in the growing process, the criterion to be optimized on the successively constructed partitions, and the way to get right-sized trees. Most implementations are ordinary binary trees, i.e. trees whose successive cuts are made by hyper-planes perpendicular to the axes, while most of the litterature concerns the various possible criteria and pruning methods. L. Devroye, L. Györfy and G. Lugosi (1996) define and consider the remarkable theoretical properties of a binary tree classifier whose prominent feature is the particular type of splits used in its construction: at a given node, partitioning is made by hyper-rectangles rather than hyper-planes. We propose a simple algorithm for the optimization problem involved. Then we compare the performance of two different implementations of our algorithm with two leading algorithms for ordinary binary trees, namely CART and C4.5 as implemented in the Splus "tree" procedure and in SAS 's Enterprise Miner respectively. For this purpose, data sets which traditionally enhance the weaknesses of classification trees are used.

KEYWORDS. Binary classification trees, XOR problem, small sample sizes, hyper-rectangles, splitting method, stopping rule.

[1]Institut de Statistique, Université Catholique de Louvain, B 1348 Louvain-la-Neuve, Belgium. Research support from "Projet d'Actions de Recherche Concertées" (No. 98/03–217) from the Belgian Government is acknowledged.

1 Introduction

Classification trees are one of the main techniques used in data mining and more precisely for "Supervised Learning", usually referred to as discrimination in the statistical litterature: they allow to establish classification rules on the basis of available correctly classified data. The space of explanatory variables is partitioned into regions in which decision is mostly taken by majority vote over the cases of the training sample, with possible adaptations for non equal misclassification costs. Like most tools of data mining, tree classifiers were developed in the perspective to be powerful on very large databases, and for problems where the computational time is a main concern. Most implementations are therefore ordinary binary classification trees, for which the partitioning method is applicable in a computationally very simple manner: the successive cuts in the space of the explanatory variables are made by hyper-planes perpendicular to the axes, so that the different attributes can be considered separately at each step of the construction. The selection of the region and of the coordinate to be split is based on the "purity" or homogeneity of the resulting partition. Various measures of purity are used, ranging from empirical error rate to more elaborated splitting criteria supplied by information theory or Bayesian probability theory. Methods differ mainly by the chosen criterion, and by the way to select a right-sized tree or equivalently to get a final tree that best captures the structure of data while avoiding overfitting. Backward pruning is prevalent for the latter purpose: a tree is grown till it is much too large and then pruned backward in a second stage. The authors of one of the leading classification tree algorithms (Breiman and al. (1984)) noted: "Within a wide range of splitting criteria the properties of the final tree selected are surprisingly insensitive to the choice of splitting rule. The criterion used to prune or recombine upward is much more important."

Despite the use of these various pruning techniques, ordinary tree classifiers are not appropriate, or give poor results when there are many regions per class, i.e. on highly fragmented distributions. The problem in its simplest setting is known as the "XOR" problem (see Figure 1). It traditionally illustrates a major drawback of the cutting method of ordinary binary trees: it does not allow to isolate small sets. At each step the cut is made by a hyperplane perpendicular to one coordinate, so that optimization is performed over halfspaces only. The extremely simple XOR structure can not be catched properly by a classifier which acts separately on the different explanatory variables even when the Bayes risk is null. Indeed no single split of the form $x_i \leq \alpha$ can lead to any substantial improvement of the purity of the partition, whatever the criterion chosen to measure it.

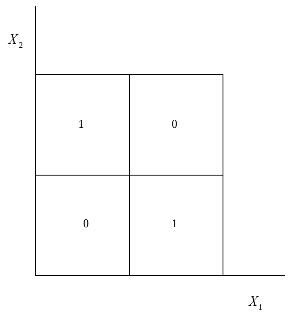

Figure 1: XOR design

In this work, we consider a binary tree classifier introduced by L. Devroye, L. Györfy and G. Lugosi ([4]). The prominent feature of this tree classifier is the particular type of splits used in its construction: at a given node, partitioning is made by hyper-rectangles rather than hyper-planes. The splitting criterion is simply the empirical error, but it is optimized over a larger set, including halfspaces as well as bounded hyper-rectangles. In particular, the method shares with ordinary binary trees the remarkable property to be invariant to monotone transformations of the axes. Accuracy and readability substantially improve on small and moderate sample sizes, as measured by the closeness of the error rate of the generated classification rules to the Bayes risk and by the complexity of the trees respectively. It is clear however that the computational order of those "Hyper-rectangular Space Partitioning" (HSP) trees will be too high to make the method practically competitive with other trees on large datasets.

In the next section, HSP trees are defined more precisely. Though the method is applicable in a more general context, we consider only discrimination problems between two populations and with continuous explanatory variables. As far as we know, no algorithm has been proposed in the litterature to handle HSP trees in practice. We propose in section 3 an algorithm with two different implementations. A few illustrative simulations are presented in section 4, where both implementations are finally compared with two leading algorithms for ordinary binary trees: CART and C4.5 implemented in the Splus "tree" procedure and SAS 's Enterprise Miner respectively. The results of some comparative simulations are presented for a few standard test designs, namely "XOR" data and "checker-board" data. For each fixed design, the behaviour of the different classifiers are compared under increasing levels of the Bayes risk.

2 Hyper-rectangular Space Partitioning trees

In a discrimination problem between two populations Π_0 and Π_1, let us denote class membership by the binary variable Y:

$$Y = \begin{cases} 1 & \text{if the case comes from } \Pi_1 \\ 0 & \text{otherwise} \end{cases}$$

and let $X \in \mathbb{R}^d$ represent d continuous explanatory variables. The learning sample is denoted by $(X_i, Y_i)_{i=1,\dots,n}$. Binary classification trees are characterized by successive binary divisions of "nodes" or sets in $\mathbb{R}^d$: if a node u represents the set A in $\mathbb{R}^d$ and its children u', u'' represent A' and A'', then we have $A = A' \cup A''$ and $A' \cap A'' = \phi$. The root of the tree, or initial node, represents the whole space $\mathbb{R}^d$ and the leaves, or terminal nodes, form a partition $\mathcal{P}$ of $\mathbb{R}^d$. Class membership decision $g_{\mathcal{P}}$ on a set $A \in \mathcal{P}$ is by majority vote: for a given $x \in A$,

$$g_{\mathcal{P}}(x) = \begin{cases} 1 & \text{if } \sum_{i:X_i \in A} Y_i > \sum_{i:X_i \in A} (1 - Y_i) \\ 0 & \text{otherwise} \end{cases}$$

An estimate of the quality of the rule $g_{\mathcal{P}}$ is given by the empirical error or percentage of misclassified cases in the learning sample. Consider the empirical error for the set $A \in \mathcal{P}$:

$$\hat{L}_n(A) = \frac{1}{n} \sum_{i=1}^{n} I_{\{X_i \in A\,,\, g_{\mathcal{P}}(X_i) \neq Y_i\}}.$$

Then we can define with some abuse of notation the empirical error of the partition $\mathcal{P}$ as

$$\hat{L}_n(\mathcal{P}) = \sum_{A \in \mathcal{P}} \hat{L}_n(A)\,,$$

.

The growing process of HSP trees is essentially the same as for ordinary binary trees: empirical error is minimized over all possible splits, but a single split is defined as a hyper-rectangle rather than a hyper-plane. Let $\mathcal{P}_t$ be the constructed partition at stage t of the process, i.e. after t cuts. Given the partition $\mathcal{P}_t$, a legal hyper-rectangle T is one for which $T \cap A = \phi$ or $T \subset A$ for all sets $A \in \mathcal{P}_t$: hyper-rectangles are not allowed to overlap. If we refine $\mathcal{P}_t$ by adding a legal hyper-rectangle T somewhere, then we obtain a partition $\mathcal{P}_{t+1}$.

The classifier proposed in Devroye and al.([4]) selects a legal hyper-rectangle T in a set $R \in \mathcal{P}_t$ which minimizes $\hat{L}_n(\mathcal{P}_{t+1})$. Since the decision functions $g_{\mathcal{P}_{t+1}}$ and $g_{\mathcal{P}_t}$ will agree except on the set R, this is equivalent to finding $T \in R$ which minimizes $\hat{L}_n(T) + \hat{L}_n(R - T) - \hat{L}_n(R)$. Starting with the trivial partition $\mathcal{P}_0 = \{\mathbb{R}^d\}$, the splitting rule just described is repeated several times, say k, leading thus to $k+1$ regions. The sequence of partitions is denoted by $\mathcal{P}_0$, $\mathcal{P}_1$,...,$\mathcal{P}_k$.

Simply stated, classification tree analysis aims to obtain a partition allowing the most accurate prediction possible; the reliability of the rule is represented by the proportion of correct classifications. The Bayes decision function, defined as

$$g^*(x) = \begin{cases} 1 & \text{if } P(Y = 1 \mid X = x) > 0.5 \\ 0 & \text{else} \end{cases}$$

provides a bound on the performance of any classifier in the sense that its probability of error $L^* = L(g^*) = P(g^*(X) \neq Y)$ is the lowest amongst all classifiers g. HSP trees are shown to be consistent classifiers provided $k = o(\sqrt[3]{n/\log n})$ and assuming X has nonatomic marginals: L_n converges to L^* with probability one (theorem 20.9, Devroye and al. (1996)).

Practically a choice has to be made between different methods to determine a good size of the tree. The problem will be addressed in Section 4.

3 An algorithm

At the first stage of the construction of a HSP tree, the algorithm provides a hyper-rectangle H such that the empirical error is minimized over all possible hyper-rectangles of the entire space $\mathbb{R}^d$. The right subtree of the root is defined as the inside of H, and the left subtree as the complement of H. The same process of optimization will be repeated on both subtrees, with the additional constraint (on the left subtree) that the solution should not intersect H. Consequently, at any level of the growing process, all the sets of the constructed partition can be written as $H_0 \setminus A$, where H_0 is some hyper-rectangle in $\mathbb{R}^d$, either $A = \phi$ or $A = \cup_{i=1}^r H_i$ and H_i, $i = 1 \ldots, r$ are disjoint hyper-rectangles in H_0 (see illustration on Figure 2). When splitting some node $H_0 \setminus A$ to refine the partition, we will refer to H_0 as the search region and to $H_1, \ldots, H_r$ as the possible "constraints": given those inputs, the algorithm will provide an optimal hyper-rectangle amongst all hyper-rectangles of H_0 that do not overlap $H_1, \ldots, H_r$.

Suppose we are given m training cases to be split at a given node. A hyper-rectangle is determined by two bounds on each coordinate, and we can take the set of ordered values of

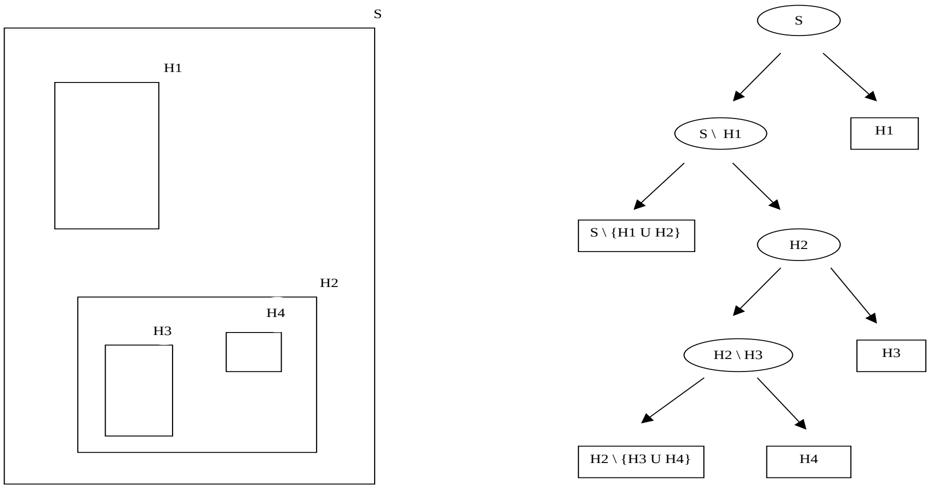

Figure 2: The growing process of HSP trees

the training cases on a coordinate axis as the set of possible bounds on this axis. Indeed, consider some fixed coordinate and denote as $\{\ell_1, \ell_2, \ldots, \ell_m\}$ the sorted values of the m considered points. Any threshold value lying between ℓ_i and ℓ_{i+1} will have the same effect of dividing the cases into those whose value of the attribute lies in $\{\ell_1, \ldots, \ell_i\}$ and those whose value is in $\{\ell_{i+1}, \ldots, \ell_m\}$. We have thus $(\frac{m(m+1)}{2})^d$ possible combinations of pairs of bounds as we include single points by allowing the inferior and superior bounds to coincide. Consequently, there are at most the same number of different groupments of points of any size that can be included in a hyper-rectangle. This allows to perform an exhaustive search for an optimal groupment but such a naive procedure is highly redundant: the same groupments of points would result of many combinations of bounds and many combinations would give empty sets.

Redundancy can be easily avoided, however. Let us illustrate this for the case $d = 2$. Let x_1 and x_2 be the two available attributes, and let $\{u_1, \ldots, u_m\}$ and $\{v_1, \ldots, v_m\}$ be the sets of ordered values of x_1 and x_2 respectively. A possible rectangle will be determined by two bounds chosen amongst the u_j and two bounds chosen amongst the v_j. The bounds themselves are included in the resulting rectangle. Let u_i and u_s be the fixed inferior and superior bounds on x_1, and let $p(u_i)$ and $p(u_s)$ be the coordinates of the corresponding sample points (see the example illustrated on Figure 3). Only those points whose first coordinate lies between u_i and u_s will be considered as possible bounds on x_2, so that the latter bounds will be chosen among at most $s-(i-1)$ values. On our example, this would be the set $\{v_{\ell+2}, v_{\ell+3}, v_{\ell+5}, v_{\ell+6}\}$. Moreover, we can exclude from this set those points whose second coordinate lies between $p(u_i)_2$ and $p(u_s)_2$. Indeed, only those rectangles for which u_i and u_s are the actual bounds on x_1 are considered, i.e. only the rectangles including $p(u_i)$ and $p(u_s)$. Practically, only the bounds inferior to $\min(p(u_i)_2, p(u_s)_2)$ or superior to $\max(p(u_i)_2, p(u_s)_2)$ are taken into account as inferior and superior bounds respectively. So in the example, only the combinations $(v_{\ell+2}, v_{\ell+5})$ and $(v_{\ell+2}, v_{\ell+6})$ will be considered for the chosen (u_i, u_s). For $d>2$, the argument is inductive: we restrict in the same way the set of possible bounds on x_j once all pairs of bounds are fixed up to x_{j-1} for $j=2,\ldots,d$.

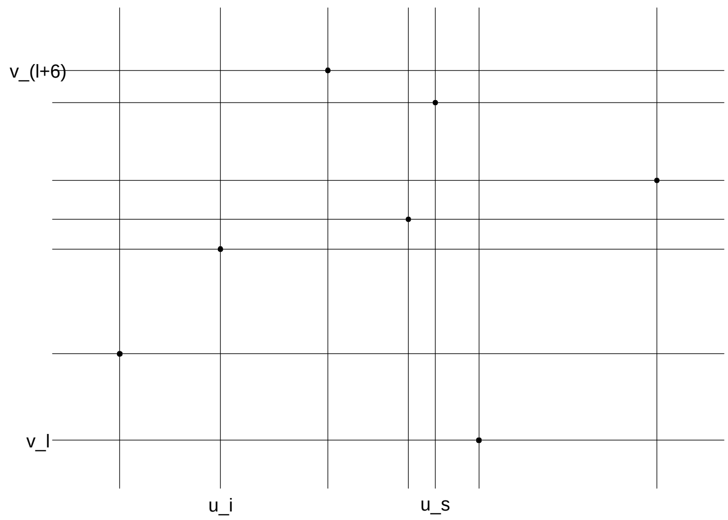

Figure 3: The dots represent the sample points in the plane (x_1, x_2). For u_i and u_s fixed on x_1, only $(v_{\ell+2}, v_{\ell+5})$ and $(v_{\ell+2}, v_{\ell+6})$ will be considered on x_2.

On the other hand, many combinations may be naturally dropped during the minimization process as they cannot lead to any improvement of the attained smallest value of the criterion minimized, namely the number of misclassified observations. This can be seen as follows. Consider a minimization process on some region $H_0 \setminus A$, and let $e(t)$ be the smallest value of the number of misclassified observations on $H_0 \setminus A$ at some stage t of the process. Let H be some hyper-rectangle in $H_0 \setminus A$. Denote H^c as the complement of H in $H_0 \setminus A$, n_H and n_{H^c} as the number of points inside H and H^c, n_0 and n_1 as the number of points from Π_0 and Π_1 in $H_0 \setminus A$, and $n_{min} = \min(n_0, n_1)$. Then the following condition is necessary for H to give a smaller value of the criterion than $e(t)$: either n_H or n_{H^c} should lie in the interval $]n_{min} - e(t), n_{min} + e(t)[$. Indeed, for a fixed value of n_H, the number of misclassified points can not be smaller than $\min(| n_H - n_{min} |, | n_{H^c} - n_{min} |, n_{min})$. This can be seen by observing that the minimum is attained when all points of either the minority group or the majority group are in the same subset. In our implementation, combinations isolating less than $\min(n_0, n_1) - e(t)$ or more than $\max(n_0, n_1) + e(t)$ are not considered after stage t of the process.

Finally, the possible constraints on a search region also reduce the number of possible cuts. However, the computational order will not decrease so much as to make the method practically competitive with ordinary binary trees on large datasets. In the most favorable case for which all observations are situated on a same line (sothat the actual dimension reduces to one), any two bounds on some coordinate corresponds to exactly one groupment of points. The number of possible groupments in this case still amounts to $\frac{m(m+1)}{2}$.

Notice that the method described above provides the smallest hyper-rectangle that minimizes the empirical error. Indeed, to isolate a given groupment of points, the bounds are always taken on the extreme values of the cloud of points even when the same groupment could be isolated in a much larger hyper-rectangle. Two different implementations of the algorithm will be considered. In the first one, the smallest hyper-rectangles resulting from the optimization process directly define the final partition of $\mathbb{R}^d$. In the second one, the hyper-rectangles have been maximally enlarged, while containing the same sets of sample cases. Finally, for both versions, if u_i and u_s are the inferior and superior values that determine a groupment of points on some coordinate, the traditionally chosen "midpoints" define the solution: the bounds are taken as $\frac{u_{i-1}+u_i}{2}$ and $\frac{u_s+u_{s+1}}{2}$. We will see in Section 4 the relative advantages of both implementations. [1]

4 Simulations on some common data designs

We present in this section results observed on some typical data designs on which ordinary binary trees mostly fail; however the simulated distributions all have decision boundaries perpendicular to the axes so that tree methods should not be discarded at first sight. The very poor results obtained with the classical algorithms are due to the construction method

[1]The algorithm is intended for continuous variables so that the probability of encountering ties is null; however in practical applications, some very small random noise has been added to some data.

of the tree, which uses a one step lookahead (performance is optimized split by split, but not on the whole tree). For each design, both versions of the algorithm for HSP trees (HSPL and HSPS, standing for large and small hyper-rectangle solutions respectively) are compared with traditional tree algorithms like CART or C4.5 .

4.1 Right-sized trees

During the construction of the tree, the empirical error rate typically decreases as the different subsets of the trees become more and more homogeneous -it is always possible to grow a tree till empirical error decreases to 0-, whereas the actual error rate mostly decreases, then becomes quasi stationary before increasing again. So the finest tree, or perfectly homogeneous tree, is obviously not the best one, and we come to one major issue of these methods: how to determine the size and structure of a "best" tree? The idea is to remove parts of the tree that do not contribute to classification accuracy on unseen cases. Pruning is the approach proposed by most well-known implementations of ordinary binary trees: decision trees are usually simplified by discarding one or more subtrees and replacing them with leaves, according to some estimation of the predicted error rate other than the highly biased resubstitution error. This leads to two families of techniques. The first family predicts the error rate of the tree using a new set of cases that is distinct from the training set. The drawback associated with this family of techniques is simply that some of the available data must be reserved for the separate set, what can lead to inferior trees when data is scarce. One way around this problem is to use a cross-validation technique, which characterizes the second family of techniques. In essence, the available cases are divided into b equal-sized blocks and, for each block, a tree is constructed from cases in all other blocks and tested on cases in the "holdout" block. For moderate values of b, the assumption is made that the tree constructed from all but one block will not differ much from the tree constructed from all data.

The various pruning methods used for the few comparative simulations that follow will be precised in each section.

4.2 Highly fragmented distributions

The first data design is the two-dimensional "4×4 checker-board" distribution. Though tree methods are best suited for high-dimensional problems, this allows to visualize on simple graphical representations the decision rules identified by the different types of tree classifiers. For fixed sizes ranging from 100 to 4000 and for fixed increasing levels of the Bayes error L^*, 100 samples were drawn from each distribution. For each sample, HSPL, HSPS and S+ trees were maximally grown on 80% of the data, while the remaining part was used as test sample to determine a best subtree. The accuracy of the resulting tree classifier was measured by the difference between its probability of error and the Bayes error. The results for moderate sample sizes and completely separable distributions (i.e. when $L^* = 0$) are summarized in Table 1; Figure 4 represents the mean values of the probability of error of the rule generated by the best subtree $L(T_{BEST})$ versus the size of the training sample for S+ trees (solid lines), HSPS trees (dashed lines) and HSPL trees (dotted lines). The latter trees clearly and substantially outperform both other methods for the accuracy criterion; nevertheless the difference decreases as the size increases.

	HSP small	HSP large	S+
$n = 400$	0.138	0.085	0.155
$n = 800$	0.064	0.039	0.066
$n = 2000$	0.024	0.014	0.021

Table 1: Mean attained error rates for checker-board data, $L^* = 0$

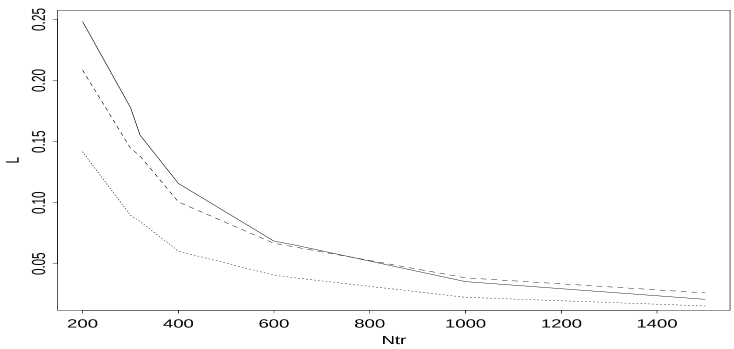

Figure 4: Mean error rate versus size of training samples for HSPL, HSPS and S+ trees

Similar results are observed for non zero values of L^*. The prediction accuracies for the three methods are closer to each other as the Bayes error increases, but HSP trees possess outstanding features like stability of the solutions and ability to recover perfectly the underlying structure for values of L^* till 0.15 (see examples on Figure 5). Both features are a natural consequence of the proper way HSP trees work, notably by directly isolating and therefore identifying "exception" zones with respect to a class label in a bigger space. On large sample sizes, ordinary trees globally and roughly come to the same classification rule, but the numerous steps needed to get a right partition differ from one sample to another. So the tree structure is very unstable and the actual data structure is hidden, making interpretation difficult without graphical visualization, i.e. as soon as the dimension is greater than 2. More precisely and to simplify, if we consider ideally pruned trees, each of the regions corresponding to any group will appear as the union of numerous leaves varying from sample to sample; HSP trees on the other hand will identify stable blocks.

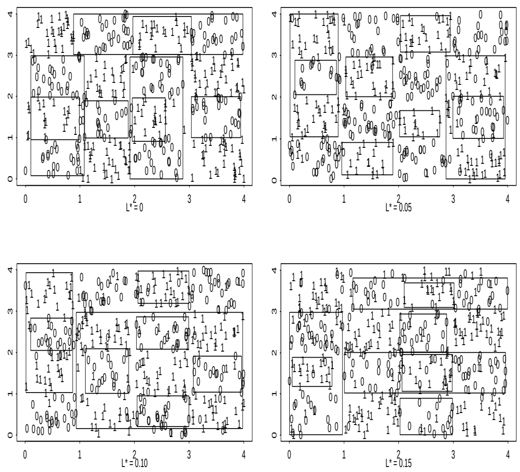

Figure 5: HSPS trees for checker-board data under increasing values of L^*

For small sample sizes (100 to 400), the following simplified cross-validation technique has been used to improve accuracy. HSP trees were grown in order of decreasing value of the resubstitution error, till all regions were perfectly homogeneous. The choice of a best tree was then over the same complete sequence of trees: the final tree classifier was defined as the tree T in the sequence for which $\hat{L}_n^{cv}(T)$ was minimal, where $\hat{L}_n^{cv}(T)$ is the estimation of the error rate obtained by cross-validation for the trees with the same number of leaves as T.

Obviously the tree obtained after k steps during the construction of a tree is not necessarily the best subtree with k final nodes that we can extract from the finest tree of the whole sequence. The purpose of most well-known pruning rules is to extract some sequence of trees in order of decreasing complexity and such that each tree of the sequence would be a "best candidate" for a fixed number of leaves. The method used here involved neither any sequence of best candidates for fixed numbers of leaves nor any complexity minimization in the final choice.

For $L^* = 0.15$, some random sample was drawn and the cross validation technique just described was applied with holdout blocks amounting 10% of the sample size. The curve of the error rates averaged over the different iterations for fixed number of steps in the growing process clearly indicates a minimum as a peak value for $k = 8$ (Figure 6). On the other hand, the same curves for the S+ trees do not show any peak value but because of the instability of the trees mentioned above, they remain around the minimum over some interval before increasing again. This justifies the final choice in the sequence in most ordinary implementations as the simplest tree such that $\hat{L}_n^*(T)$ is inferior to $\min \hat{L}_n^*(T) + a *$ std err of $\hat{L}_n^*(T)$, $a \in [0, 1]$.

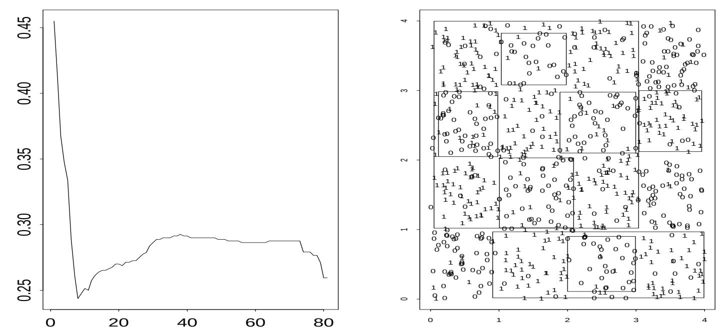

Figure 6: Error rate versus size for cv iterations and best corresponding classifier (k=8)

The results of the comparisons were similar on small sample sizes as well as for different levels of the Bayes risk.

4.3 Effect of noisy variables

While HSP trees clearly provide a solution to the XOR problem (see Figure 1) or to related problems by allowing and even forcing simultaneous cuts on all explanatory variables, the advantage of this feature may turn into a drawback in presence of noisy variables if we do not include infinite hyper-rectangles. Let us illustrate this on a simple example.

Let X_1 be a random variable that completely determines group membership:

$$\begin{aligned} X_1 \mid Y = 1 &\sim U[-1, 0] \text{ and} \\ X_1 \mid Y = 0 &\sim U[0, 1]. \end{aligned}$$

Let $g_{n,j}$ be the rule generated by a HSP tree, smallest version, constructed on a sample of size n drawn from $(X, Y) = (X_1, X_2, \ldots, X_j, Y)$ where $X_2, \ldots, X_j$ represent pure noise, e.g. independant uniformly distributed variables on $[0, 1]$. We denote by $L_j = L(g_{n,j})$ the probability of error of such a rule: $L_j = P(g_{n,j}(X) \neq Y)$. Suppose that for a given sample, the bounds on X_1 isolate the 0 cases (see Figure 7). Then, if we denote by T_i the first order statistic on X_i in the subsample of the 0-cases and by S_i one minus the last order statistic on X_i in the same subsample, we have

$$\begin{aligned} L_1 &= S_1 + T_1 \\ L_k &= L_{k-1} + (1 - L_{k-1})(S_k + T_k) \end{aligned}$$

or equivalently

$$L_k = (S_1 + T_1) + \sum_{i=2}^{k} (S_i + T_i) \prod_{j=1}^{i-1} (1 - (S_j + T_j))$$

Consequently, we have $EL_k = 2\,ET_1 \sum_{i=1}^{k} (1 - 2\,ET_1)^{i-1}$ and clearly $EL_k > EL_{k-1}$: the error rate of the rule increases with the number of variables without discriminatory power, whereas the Bayes risk is fixed to 0. This "curse of dimensionality" problem is avoided by ordinary binary trees as well as by HSP trees, large version.

However, if we replace (X_1, X_2) of the previous example by a XOR design so that X_1 and X_2 together completely separate the distribution whereas the remaining variables $X_3, \ldots, X_j$ do not contribute substantially to discriminate between both groups, only HSP trees are expected to provide a satisfactory solution. Indeed, a single cut on any variable with poor discriminatory power may increase the purity of the partition more than a single cut on X_1 or X_2 due to the chosen XOR design. And to a certain extent, if we increase the discriminatory power of the variables $X_3, \ldots, X_j$, than the global rule of ordinary trees is expected to perform even poorlier. The various simulations realized confirm this weakness: ordinary trees give extremely complex and unstable solutions, with bad prediction accuracy. Table 2 contains the results of simulations performed on samples of size 100 drawn from the

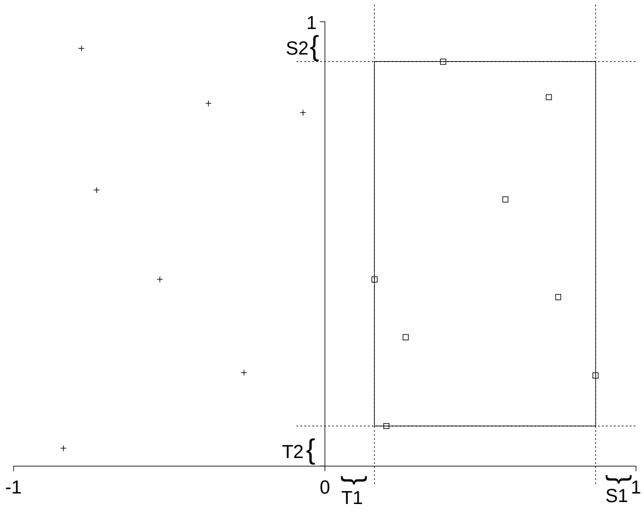

Figure 7: Increase of error when adding X_2 to X_1 for HSP tree small version

distribution described above with two noisy variables. The levels of the Bayes risk for the marginal distribution (X_3, X_4) are fixed to 0.50, 0.45 or 0.40, i.e the variables X_3 and X_4 have slightly increasing discriminatory power.

	HSP small	HSP large	S+
$L^*(X_3, X_4) = 0.50$	0.156 (0.044)	0.044 (0.034)	0.209 (0.147)
$L^*(X_3, X_4) = 0.45$	0.151 (0.039)	0.044 (0.036)	0.251 (0.139)
$L^*(X_3, X_4) = 0.40$	0.154 (0.037)	0.049 (0.036)	0.282 (0.123)

Table 2: Mean attained error rates (and standard errors) for XOR with noise data

It should be noticed that in this section and in what follows, the pruning method was based on a test sample used to select the best subtree amongst all possible subtrees of the finest tree produced on the train sample.

5 Application to the "Pima Indians Diabetes data"

We present in this section an application of the algorithm on a data set available in the public domain: the Pima Indians Diabetes data (http://www.ics.uci.edu/$\sim$mlearn/MLRepository.html). The class labels of the Pima data are 1 for patients who tested positive for diabetes (268 cases) and 0 for those who tested negative (500 cases). There are 8 predictor variables for the 768 patients, all females, at least 21 years old, and of Pima Indian heritage. The predictor variables and their definitions are shown in Table 3.

Variable	description
PRGN	number of times pregnant
PLASMA	plasma glucose concentration at two hours in an oral glucose tolerance test
BP	diastolic blood pressure (mm Hg)
THICK	triceps skin fold thickness
INSULIN	two hour serum insulin (μ U/ml)
BODY	body mass index (weight in kg/(height in m)^2)
PEDIGREE	diabetes pedigree function
AGE	age (years)
DIABETES	class variable (1 if diabetes, 0 otherwise)

Table 3: Predictor variables for the Pima Indians Diabetes data

We worked on a reduced data set of size 392, avoiding cases with missing or badly recorded values (e.g. BODY = 0), and without the first predictor (discrete variable). Among those 392 remaining cases were 130 patients who tested positive for diabetes so that the error rate of the tree consisting of the root only (decision in favor of the majority group whatever the values of the predictors) amounts to 130/392 i.e. 33%. Various trials were realised with the C4.5 algorithm (SAS 's Enterprise Miner tree tool) applied to the data after random partitioning of into train (40%), test (30%) and validation sets (30%). For all the resulting trees, the variable PLASMA appeared at the top of the tree, and for half of them no further split was done, indicating that tree methods may not be optimal here. After the first cut on PLASMA, the tree structure was not stable; the predictors mostly used were AGE and INSULIN. The best single cut on PLASMA for a tree grown on the 392 data provides a resubstitution error of 0.24, which is also the mean level attained when entering all variables. Notice that using the CART algorithm under the same conditions, we obtained an error rate of 0.26.

Using the same proportion of the cases as training set (namely 40%), HSP trees were grown on the three fore-mentioned variables, and the same average error rate was attained on the test samples, i.e. 0.24. For those trees as well as for the trees grown on the 7 predictors (on smaller training sizes) the structure depends mainly on the group isolated in the first hyper-rectangle. By dropping the predictors that are taken on their whole range, it appears that the non diabetic cases are characterised by PLASMA and AGE while for the diabetic cases the most effective cuts are made on PLASMA and BODY. This characterisation, not revealed by ordinary trees, could also be enhanced by the modified splitting criteria proposed in A. Buja and Y.-S. Lee ([2]). Notice also that by using only PLASMA and AGE to grow HSP trees on 80% of the data, we get an average error rate of 0.21 (estimated by cross-validation on the 20% remaining data).

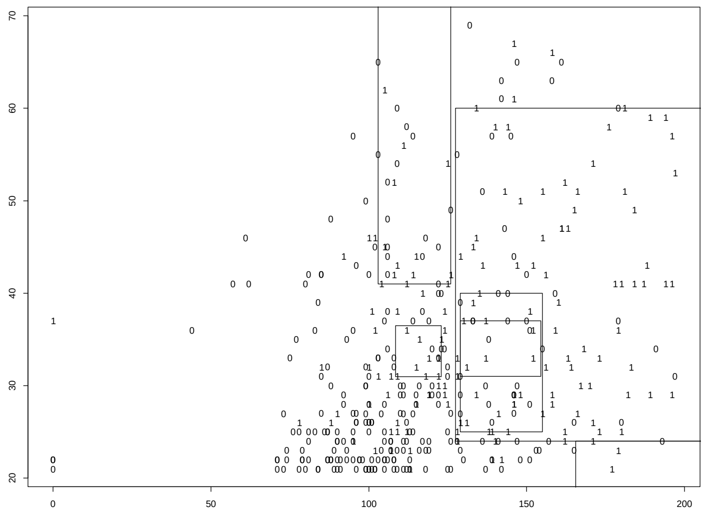

Figure 8: HSP tree classifier on PLASMA and AGE for Pima Indians Diabetes data

6 Conclusion

The problem for ordinary binary trees to provide good classifiers in the data distributions presented in the previous sections comes from their growing process allowing only a one step ahead examination of the possible splits. The consequences of any candidate split at a given stage of the optimization process are never checked out, as this avoids a combinatorial explosion over future choices. The implementation that we proposed however involves optimization over the set of hyper-rectangles (HSP trees) instead of over the set of hyper-planes (OB trees). As expected it provides a solution to the typical problematic data designs presented in this paper: for small sample sizes, HSP trees have the advantage to combine higher accuracy, stability, simplicity and to give a precise representation of the underlying structure. To perform more valuable comparisons, many topics should be investigated and developed further like better adapted rules for stopping and/or pruning, different splitting criteria, a specific treatment of the non continuous predictors, a generalisation to the multi-class problem...

References

[1] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth International, Belmont, CA, 1984.

[2] A. Buja and Y.-S. Lee. Data mining criteria for tree-based regression and classification. Manuscript, AT & T labs, 1999.

[3] L. Devroye. *Non-uniform Random Variate Generation.* Springer-Verlag, Berlin, New York, 1990.

[4] L. Devroye, L. Gyorfi, and G. Lugosi. *A probabilistic theory of pattern recognition.* Springer-Verlag, New-York, 1996.

[5] D. J. Hand. *Construction and assessment of classification rules.* John Wiley (London), 1997.

[6] D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine Learning, neural and statistical classification.* Ellis Horwood, 1994.

[7] J. R. Quinlan. *C4.5: Programs for Machine Learning.* San Mateo, CA: Morgan Kaufmann, 1993.

[8] StatSci. *S-PLUS 5 for Unix, Guide to Statistics.* Seattle: MathSoft, Inc., 1998.

[9] W. Venables and B. Ripley. *Modern Applied Statistics with S-Plus.* New-York, NY: Springer-Verlag, 1994.