Dannegger:

Tree stability diagnostics and some remedies against instability

Sonderforschungsbereich 386, Paper 72 (1997)

Online unter: http://epub.ub.uni-muenchen.de/

Tree stability diagnostics and some remedies against instability

FELIX DANNEGGER

Institut für Medizinische Statistik und Epidemiologie

Technische Universität München

Ismaninger Straße 22, 81675 München

email: felix@imse.med.tu-muenchen.de

Abstract

Stability aspects of recursive partitioning procedures are investigated. Using resampling techniques, diagnostic tools to assess single split stability and overall tree stability are introduced. To correct for the procedure's preference for covariates with many unique realizations, corrected p-values are used in the factor selection component of the algorithm. Finally, methods to stabilize tree based predictors are discussed.

Keywords: CART, Tree–based methods, stability, diagnostic tools, predictor aggregation, p–value adjustment

1 Introduction

Recursive partitioning methods or tree-structured algorithms offer a nonparametric alternative to classic, parametric regression and classification methods. Their modular approach allows adapdation to other problems such as survival analysis. In addition, the results of tree based analyses are easily conveyed to nonstatisticians due to the intuitive binary tree structure of the predictors obtained.

Along with the increased practical use of recursive partitioning methods have come doubts concerning the reliability and stability of the procedure. One source of such instability is the method's tendency to overfit the data, and care needs to be taken to assure overfitting is objectively avoided. However not all of the instability can be assigned to too large trees. Much of the variance of these procedures is due to the often unnatural dichotomization of metric covariates. Often the defining cut-points are highly variable, and due to the hierarchical structure of tree based models, any change in a branch of a tree will certainly affect lower portions of the same branch and thus the predictor as a whole, reducing its predictive accuracy.

The aim of this paper is to introduce tools to assess such variability and offer ways to overcome unneccessarily large prediction error. In section 2, the components making up a tree-based algorithm are explained. Data from a breast cancer study are then analyzed using trees. Section 3 demonstrates what is meant by tree instability and offers simple node-level diagnostic tools based on resampling techniques to assess split stability. In section 4, the algorithm's well known preference for metric factors with many unique realizations in the training set is discussed. Correct p-values are obtained not through an adjustment of the raw p-values, but rather via a general, though computationally expensive permutation approach. Section 5 discusses stabilizing procedures for tree-structured predictors. In addition to reviewing

Breiman's "bootstrap aggregation" suggestion, a node-level stabilizing approach is introduced. Finally, a conclusion and an outlook pointing out open questions are given.

2 Recursive Partitioning

2.1 Data notation

The goal of recursive partitioning procedures is to predict an unknown quantity y_i of an individual i based upon known realizations of p covariates $x = (x_1, \ldots, x_p)_i$. In order to construct a predictor $\hat{y} := d(x)$, a learning sample or training set L consisting of a group of elements of tuples (y, x) with *both* y and x components known, is employed. Different scales of y determine the type of the problem and tree. If y is categorical, the tree produces a discriminant function assigning each individual to an estimated class, if y is metric, d is a regression function. As will be shown later, if y denotes possibly censored survival times, CART can be extended to handle these data also and produce so called survival trees.

2.2 Algorithm

The most widely used incarnations of recursive partitioning procedures in statistics go back almost exclusively to CART in Breiman, Friedman, Olshen and Stone (1984). Since then, many extensions and improvements have been suggested, altering some or all parts of the modular algorithm, but always keeping the main idea of sequential binary partitioning followed by some form of tree pruning to control overfitting. A thorough review with emphasis on biostatistics can be found in Zhang, Crowley, Sox and Olshen (1997).

Tree growth

In the first step of the algorithm, the predictor space $\mathbf{X}$ is partitionend into disjunct subspaces to either form groups of elements, called nodes, which

are homogenous with respect to the response variable of interest, or to form subgroups with maximized between group heterogeneity. This is achieved by splitting the population at a node into two subpopulations according to a simple question about one of the covariates.

Formally, one constructs a set Q of split inducing binary questions of the form 'Is $X_j \in A$?' where $j \in \{1, \ldots, p\}$ and $A \subset \mathbf{X}$. Observe that $Q = Q_1 \cup Q_2 \cup \ldots \cup Q_p$, where each Q_j is the set of binary questions concerning covariate j. For ordered covariates X_i, the set of possible questions reduces to 'Is $X_j \leq c$?', with c taking on all values of covariate realizations for elements in the current node. For unordered covariates, all possible divisions of categories into two groups must be examined. Each of these questions induces a candidate split q, sending elements belonging to A to the left sibling node, others to the right.

For every $q \in Q$, a goodness of split criterion $GS(t, q)$ is evaluated to determine the best split of a node t. Usually this criterion will measure the improvement in homogeneity of the resulting subgroups of a candidate split with respect to the response, choosing the split which produces the most homogenous sibling nodes. Similarily, goodness of split criteria have been derived which maximize heterogeneity between subgroups. Common goodness of split criteria for classification are the χ^2-test for contingency tables or the entropy measure. In the regression setting choices include the mean squared error or least absolute deviations. The value of the split criterion is recorded for each possible $q \in Q$ and the split

$$q_{opt} := \arg\max_{q \in Q} GS(t, q)$$

is selected as the optimal split. This splitting process is recursively repeated for the resulting subgroups, until it is determined that further partitioning is not warranted. Checking if further splitting is warranted or possible usually involves enforcing stopping criteria such as a minimum node size n_{min} and possibly a minimal value GS_{min} of the optimal goodness of split. Nodes which are not split again are called terminal nodes and form the final sub-

groups.

Constructing a predictor from the terminal nodes

After a set of terminal nodes is obtained, the final step of characterizing the elements of the terminal nodes or more precisely assigning the same estimate $\hat{y} := d(x)$ for each element of a terminal node remains. In the classification setting this will be an estimate of class membership, for regression an estimated response value is produced. The result of such an algorithm can be displayed in a binary tree structure.

Controlling tree size

Although naive predictors can be obtained by using just this first part of the algorithm, it is well known that these trees aren't flexible enough with respect to model complexity. Depending on the stopping rule used to determine whether a node is to be split again or not, the tree or some branch of it, will tend to be too large or too small, either overfitting the data or not capturing all the information contained in the learning sample. Thus a two stage procedure to determine the final tree is usually used for tree size or model selection. The first stage is as above but with sufficiently liberal stopping rules ensuring that the tree obtained is in no case too small. The second stage called *cost-complexity pruning* involves cutting down the tree to the right size in a step-wise fashion via a complexity adjusted error estimate of the tree

$$R_\alpha(T) = R(T) + \alpha|\tilde{T}|, \tag{1}$$

where $R(T)$ is the raw error measure, $|\tilde{T}|$ denotes the number of terminal nodes in the tree and α is the penalty weight. Minimizing (1) as α increases from 0 until only the root node is left for α_{max}, creates a nested sequence of cost-complexity optimal trees for a finite sequence of fixed α.

To find the globally optimal tree, cross-validation techniques are used within which auxillary trees are grown and subsequently pruned using the same sequence of α's obtained from pruning the original tree. For each α in the

sequence, the error rate of the pruned tree is estimated by the mean of the error rates of the pruned auxillary trees, for which honest estimates of error are available, since a portion of the learning sample L was left unused for each auxillary tree. The tree found to have minimal crossvalidated error is then chosen as the final tree or model.

2.3 Adaptation to survival data

One area where recursive partitioning methods have become widely used is medicine and more specifically in a survival analytic context, where clinicians are interested in predicting prognosis based upon certain risk factors, or more generally prognostic factors. Here, trees are especially appealing since in addition to stratifying study populations into subgroups with distinctly different risk expectations, they also allow simple and intuitive identification of potential prognostic factors and their possible interactions. Moreover, the suggestive, graphically intuitive structure of the predictor is a valuable tool when discussing results with clinicians.

In order to be able to handle censored survival data, certain parts of the CART algorithm need to be adapted. The construction of the set of candidate splits Q remains unchanged in the survival analysis setting. In contrast, the goodness of split criterion, the way elements of a terminal node are characterized and to some degree the pruning method need to be adapted to the survival data situation. Commonly used extensions of recursive partitioning to the survival analysis setting can be found in Ciampi, Chang, Hogg and McKinney (1987), LeBlanc and Crowley (1992) and LeBlanc and Crowley (1993). To divide the population of a node into homogenous subpopulations, the log-rank test or similar tests with prespecified weights and thus emphasis on certain time periods are commonly used. For every possible candidate split $q \in Q$, the p-value of the log-rank test used on the induced subpopulations is recorded. The best split q_{opt} is obtained for that q which has the smallest p-value. The split is then performed according to q_{opt}, if the stop-

ping criteria aren't met, otherwise the node is declared terminal.
The most pronounced alteration from regular CART occurs when assigning estimated responses to elements of a terminal node. Here a single value usually does not suffice. Instead, Kaplan-Meier estimates of cumulative survival for the populations of each terminal node, possibly along with estimates of relative risk with respect to the overall population under a proportional hazards assumption are given.
Cross-validation based pruning methods using proportional hazards martingale residuals as error measures can be used to control tree size, although enforcing a maximum optimal p-value of, 0.01 for example, is another popular way to restrict model complexity. However, this approach also suffers from the lack of flexibility mentionend earlier.

2.4 Example: Breast cancer

To demonstrate the use of tree-based methods, we employ a survival analysis example. The data come from a prospective study of 315 breast cancer patients conducted at the Technische Universität München. While the main aim of the study is to identify prognostic factors in node-negative patients (patients, where the cancer has not spread to neighboring lymph nodes), here we include all patients for our illustration. The goal remains the same: we wish to identify the main prognostic factors determining further prognosis, that is factors which allow a prediction as to whether the patient will experience a relapse. For each patient, the minimum of time under observation and time to relapse T and a censoring or event indicator δ discriminating tumor relapse ($\delta = 1$) or disease-free survival ($\delta = 0$) are recorded. In addition, the following covariates are provided on an individual basis: age of the patient at surgery (AGE), size of tumor in centimeters, number of positive removed lymph nodes ($LYPO$), progesteron and estrogen receptor states (DER and DPR), menopausal status ($MENOP$) and concentration of urokinase plasminogen activator (UPA) and its inhibitor (PAI) in the removed tumor

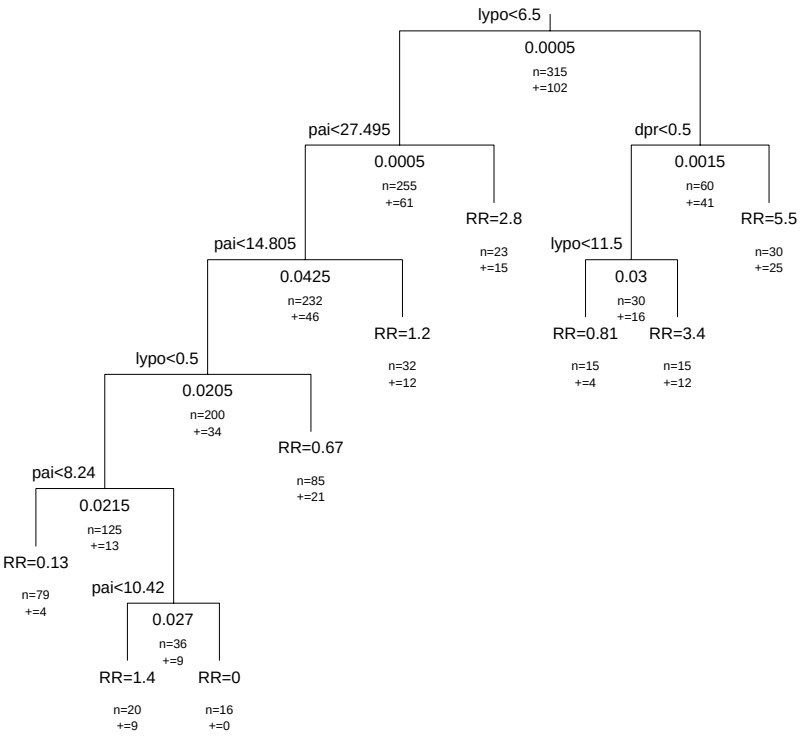

Figure 1: Survival tree for breast cancer data.

tissue. The last two substances are thought to measure body stress and are thus hoped to give an indication of the aggressiveness of the tumor.

A survival tree was grown on these data using the log-rank test as split criterion and enforcing a minimum node-size n_{min} of 15 and a maximum, optimal p-value (GS_{min}) for the log rank tests of .05. The result is depicted in figure 1. One can now follow the tree based predictor down for each present *or future* patient individually, branching off to the left or the right, according to the splits on the respective covariates at the determined cut-points, until

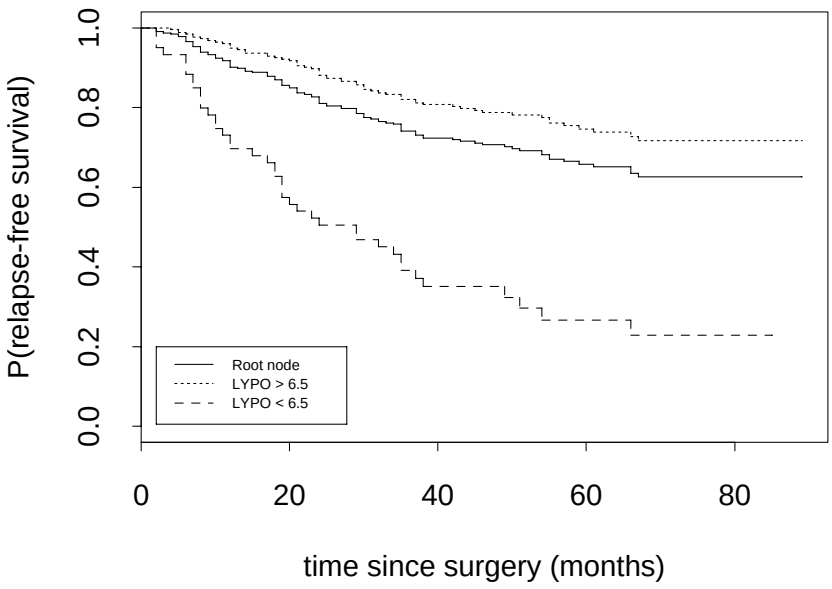

Figure 2: Kaplan-Meier estimates for the subpopulations induced by the root-node split on lymph node status.

the individual finally arrives at a terminal node. For these nodes (as for the nonterminal nodes), estimates of relative risks with respect to the total learning sample and Kaplan-Meier estimates of cumulative survival can be computed. Of course, any other suitable measure describing terminal node populations could also be added. Relative risk estimates are shown for the terminal nodes. Figure 1 also depicts remaining sample sizes and number of events for each of the terminal nodes. Notice how PAI plays the most important role in further predicting prognosis in the left subbranch of the tree. This influence however, seems ill-captured by the repetitive binary structure of the tree, as four splits are necessary to depict the relationship.

Figure 2 contrasts the survival expectation of the root node (the whole training set) with those of the two subpopulations induced by the split on

the dominating prognostic factor, the number of positive removed lymph nodes. Observe the clear separation of the two groups, with patients with six or less affected lymph-nodes having a markedly better prognosis than their counterparts with seven or more affected nodes. By now, one has gained alot of information: lymph-node status ($LYPO$) is the dominating factor determining prognosis, so much so, that a stratified analysis seems called for. Supporting this approach are the tree's strong suggestions at interactions: while PAI is important in the left branch of the tree, it plays no role in determining outcome in the poor prognosis group (right branch) of the tree. Finally, one worries that modelling PAI by a series of dichotomous factors may not be appropriate.
We will try to shed some light on this and other problems in the next sections.

3 Node-level stability diagnostics

Tree based predictors produce suggestive results. The hierarchical ordering of the included factors lends itself to assessments of factor importance, the general notion being that the closer to the root node a factor appears, the more important is its influence on the response. In addition, the dichotomization of factors allow convenient interpretations of individuals falling below or above a certain cutpoint. In a medical setting, factor realizations below the cutpoint could be considered within the normal range and others in the elevated or pathological range.
While these interpretations are tempting, they are routinely used without any idea about the variability involved. This however may be just as dangerous and misleading as ignoring available confidence regions within other statistical estimation procedures. What one needs then, is some measure of reliability and stability of the partitions chosen by the algorithm. Since direct, analytical inference about the distributions involved is difficult if not impossible due to the hierarchical dependence structures, resampling techniques such as the bootstrap may offer a simple alternative.

When CART partitions the population at a node into two sibling nodes, the choice of the cutpoint determines which elements of the original node are branched to the left, and which ones to the right. Any change in sibling populations of course has an effect on future partitions. In some cases it might be argued that such a cutpoint will be naturally given, such as when the relationship between the factor and the response involves some sort of threshold below which all values of the factor have roughly the same influence on the response, and above which the relationship is essentially different. In most situations however the relationship between a factor and the response will be much more complex, making the process of finding a cutpoint or indeed determining whether an adequate cutpoint exists at all, difficult. While recursive partitioning algorithms generally do a good job at the former, they are usually not at all concerned with the latter. Instead, their data-driven, black-box behaviour all but ensures that a critical reflection upon the choices made is never done. This eventhough simple diagnostic tools can be made available with little effort, as will be seen in the following section.

3.1 Graphing the split criterion

To start out, looking at a graph of the test-statistic plotted against realizations of the corresponding factor provides a first impression of the adequacy of the chosen factor and its cutpoint. Figure 3 shows such a plot for the factor $LYPO$ for the root node of the tree from figure 1. The chosen cutpoint of 6.5 lymph nodes dominates the graph, and the choice appears obvious. A rugplot of factor realizations further aids in assessing the resulting proportions in the sibling nodes for a potential cutpoint. In contrast, figure 4 depicts the same graph, but for the covariate PAI at node 2. Two seperated modes are visible and it is not at all clear, that the choice of 27.5 as used in the tree displayed in figure 1 is reasonably stable or if a stable choice can be found.

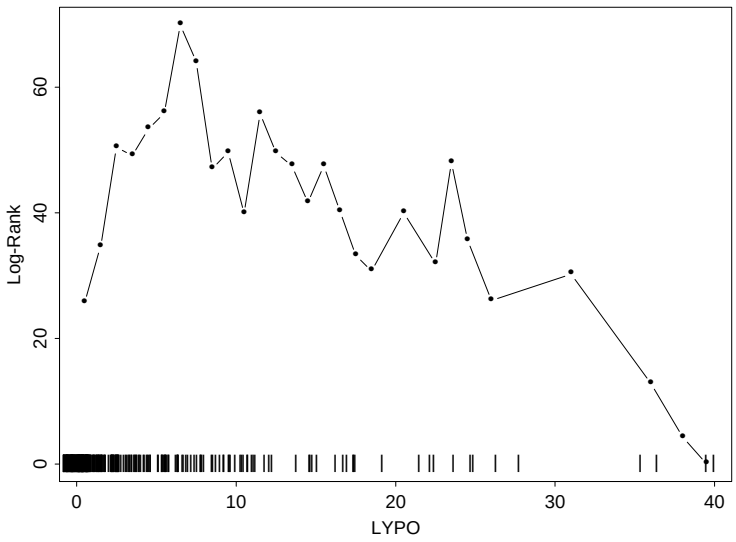

Figure 3: Graph of the splitcriterion at the root level for the factor "number of positive, removed lymph nodes ($LYPO$)".

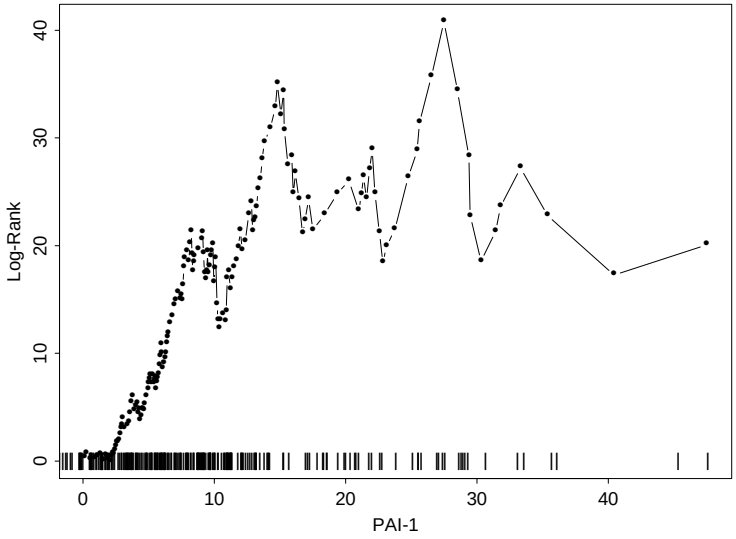

Figure 4: Graph of the split criterion at node 2 for PAI.

3.2 Bootstrap confidence intervals for the cutpoints

To further this discussion we employ bootstrap techniques. Consider the population $L_t \subset L$ of node t. By drawing B (B large) bootstrap samples $L_t^{(b)}$ with $b = 1, \ldots, B$, we can repeat the optimization process for all replicate samples. As usual, we obtain bootstrap confidence intervals for the cutpoint by selecting the appropriate quantiles of the empirical distribution F_{cut} of the bootstrapped cutpoints. Thus, a confidence interval at level γ for a cutpoint would be given by $[F_{cut}(\frac{\gamma}{2}), F_{cut}(1 - \frac{\gamma}{2})]$.

This method was employed for PAI at node 2. The results are shown in figure 5, with a density smother having been applied to the bootstrap distribution of the cutpoints, and the 90% confidence interval from $[12.8, 34.8]$ included. At once, one can appreciate how highly variable a cutpoint for *pai* is in this situation, as the confidence interval easily encompasses the two modes. In addition, the seemingly optimal cutpoint will only slice off a disproportionately small subgroup, in the process not revealing much structure.

3.3 Assessing factor importance rankings

So far, we have assumed the choice of the factor to be fixed in our discussion of tree stability. We will now shift focus, and concentrate on the stability of the process governing the choice of the covariate which is used to split a node.

To illustrate the problem, we will leave the survival analysis setting for the moment and instead simulate a binary classification problem. The training set includes 5 metric covariates, each uniformly and independently dis-

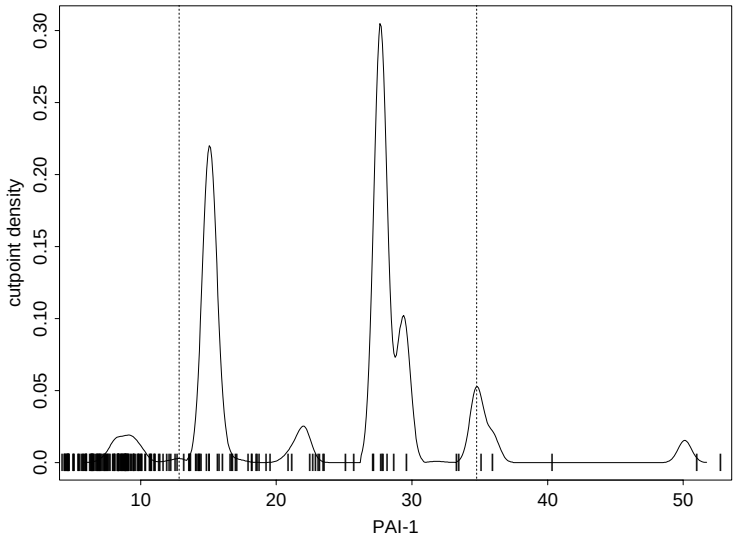

Figure 5: Smoothed density of bootstrapped cutpoints (solid) along with 90% confidence region (dashed) and rugplot of PAI realizations in node 2.

tributed on $[-1, 1]$. The response y is determined by

$$y = \begin{cases} 1, \text{ if } & (X_1 < 0 \wedge X_2 < 0) \vee \\ & (X_1 > 0 \wedge X_2 > 0) \\ 0, \text{ else.} \end{cases}$$

Thus, only two covariates have a systematic influence on the response while the remaining three are added as noise to distract the algorithm.

The complete tree is not of interest here, we will return to it in section 5. For now, we concentrate on the root node split instead, that is on the partitioning of the complete training set. Even without added noise variables, this is a tough problem for recursive partitioning to handle, as there is no way for it to uncover the structure of the boolean operators within a single split. To see what happens, we again use node-level bootstrap replicates, this time using $B = 100$ replicates.

Table 1 shows the number of times each covariate was used to split the root node. As one might expect, the two factors actually involved in the boolean

Table 1: Results for one simulation (100 bootstrap samples).

Covariate	number of times used
X_1	40
X_2	36
X_4	13
X_3	7
X_5	4

combination appear roughly the same number of times. The disturbing observation is the fact, that for almost 25% of the replicate samples, a factor which has no systematic influence whatsoever on the response is determined as being most important. Nonsensical splits like these which depend completely on idiosyncrasies in the data, remain largely invisible when just a single split is generated. In contrast, employing simple diagnostic tools as described above in a thorough fashion, allows detection of splits which are based on chance rather than structure.

In the next two sections, we will discuss methods, which reduce or at least average out some of the variability uncovered in this section, and thus allow for the construction of predictors with improved predictive accuracy.

4 How to account for the multiple testing situation

One difficulty with tree-based predictors is their bias towards selecting metric covariates when partitioning the test sample. Due to the optimization process which is started anew at every node when trying to find the best binary split of elements in the sample, it is clear that continuous covariates have a better chance of providing homogenous subgroups within the learning sample, than categorical ones with only a few unique realizations. While this

behaviour is justified to a certain extent as metric covariates do have the potential of carrying more information, there is also the danger of locking into chance idiosyncrasies in the training sample. One way this phenomenon may show up in tree-structured classifiers is the so called end-cut preference of split criteria. Often, structure in a factor is seemingly revealed by splitting of one disproportionately small subpopulation whose size is near the minimum nodesize limit, and one subpopulation still carrying the overwhelming number of cases of the original node to be split. While modified split criteria have been proposed to reduce this tendency by penalizing splits of unequal proportions, the type of penalty function remains arbitrary and introduces yet another algorithmic tuning parameter.

Another way of looking at this situation is by studying the distribution of the split criterion, as is done naturally for test-based split criteria such as the log-rank test in the survival analysis setting. The two sample log-rank test is asymptotically χ^2 distributed with one degree of freedom, but when looking for the best split one does not perform one log-rank test on two prespecified, fixed populations, rather one shifts the two subpopulations through all possible combinations until the pair producing the largest test statistic is found. Obviously, for this maximally selected test statistic, the χ^2 distribution will not hold, drawing into question the utility of comparing the raw, unadjusted p-values between covariates with unequal numbers of unique realizations. This is just another way of saying, that one expects a metric covariate to do better when modelling a cutpoint in a flexible, completely datadriven fashion. However, since covariates of different scales need to be compared to assess the goodness of splits and for the algorithm to work, adjusted or simply correct p-values are required.

Several methods for adjusting p-values have been proposed, Hilsenbeck and Clark (1996) give a comparative review of possibilities for adjusting log-rank test p-values. Here, we wish to demonstrate the use of permutation techniques as proposed by LeBlanc and Crowley (1993) to obtain correct p-values without actually basing these on the raw p-values. Although this procedure

is computationally demanding, it naturally transfers to most any other split criterion both within and outside the survival analysis setting.

Assume that optimal splits for node t have been obtained for every covariate resulting in maximized teststatistics $GS_{max}(j,t)$ for $j = 1,\ldots,p$. To estimate correct p-values for the optimal split of each covariate, we will obtain an estimate of the distribution of each of the teststatistics. We do this by deliberately breaking up the structure between the response y and the predictor variables x. Since computational burdens usually don't allow generation of all possible recombined node t populations and the calculation of the exact distribution of the test-statistics is thus not feasible, a large number K of random permutations are used instead. Note that the process of breaking up the structure between y and x components of the training set is very general and thus flexible and that in contrast to commonly used p-value adjustment approaches, no further assumptions are neccessary.

Specifically for survival data, we permute the response component $y = (T,\delta)$ with the explanatory covariates x of the elements in t. Now, for each of these permuted node populations, we repeat the process of finding an optimal split in the exact same fashion as for the original data. Thus, one receives an optimal split-criterion $GS^k_{max}(j,t)$ for each covariate in the k-th permutation, with $k = 1,\ldots,K$. We can now obtain an estimate of the teststatistic distributions through their empirical distributions and use

$$\pi_{adj}(j) = \frac{\sum_{k=1}^{m}\{I_{\{GS^k_{max}(j,t)\geq GS_{max}(j,t)\}}\}+1}{m+1}. \qquad (2)$$

as an estimate for the p-value of $GS_{max}(j,t)$. Then covariate j^* with

$$j^* = \arg\min_{j\in\{1,\ldots,p\}}\{\pi_{adj}(j)\},$$

is chosen to split node t at the cutpoint obtained from optimizing in the original data. If this split meets some sort of minimum improvement criterion (GS_{min}) such as $\pi_{adj} \leq .05$, the split is performed, otherwise t is declared terminal. Note that in order to receive correct p–value estimates of adequate

resolution, K must be chosen sufficiently large, otherwise it will be impossible to distinguish between important factors with similarily small, raw p-values. The correction term in (2) assures that the estimate will be conservative and always at least equal to $1/(K+1)$.

To appreciate how p-value corrections or adjustments affect not only estimates of effect size, but also directly influence tree structure, consider table 2 where the results of the split optimizations are depticted for node 3 of the breast cancer tree from figure 1. Columns 2 and 3 show the apparent factor importance rankings along with the raw p-values based on the χ^2_1 distribution assumption. Columns 4 and 5 impressively demonstrate the change in importance rankings when estimates of correct p-values are used instead. The seemingly most important factor PAI drops to third place and progesteron receptor status, a binary factor, is determined as most important factor. Notice how the overoptimism in effect size (raw p-values) is drastically corrected particularily for metric covariates, in addition to the change in rankings.

Table 2: Comparison of raw and corrected factor importance rankings and p-values.

	raw		corrected	
Factor	rank	p-value	rank	p-value
PAI	1	0.000559	3	0.01
DPR	2	0.000604	1	0.0015
AGE	3	0.005873	6	0.06200
DER	4	0.006897	2	0.006
DHORM	5	0.010612	4	0.0125
MENOP	6	0.012251	5	0.019
UPA	7	0.048851	8	0.312
LYPO	8	0.050768	7	0.2495
TUMOR	9	0.37069	9	0.8925

5 Stabilizing predictors

While adjusting for different scalings of the factors studied with methods such as the one described in the previous section helps reduce the number of splits caused by artefacts in the training set, this approach can't on its own remove all or even most of the undesired variability of tree based prediction rules. Consider for example the root node split characteristics of our first simulation experiment in section 3.3: there simply is no single perfect or correct split in this situation. Without some sort of averaging or stabilizing, some structure (here: one part of the boolean relation) will inevitably be missed.

To further demonstrate the need for such measures we conduct another simulation experiment, this time within a survival analysis framework. The basic ideas for the simulations are taken from LeBlanc and Crowley (1993). We simulate three models generating exponentially distributed failure times for individuals with 5 covariates each. All covariates are drawn iid. from the uniform distribution on $[0, 1]$. Their influence on the intensity paramter λ of the failure time generating exponential distributions can be seen from table 3. Model A has a constant intensity of $\lambda = 1$, thus none of the 5 covari-

Table 3: Construction of the three simulation models.

Model	$\theta_i = log(\lambda_i)$	No. of risk levels
A	$\theta_i = 0$	1
B	$\theta_i = I_{\{x_{1i} \leq 0.5 \cap x_{2i} > 0.5\}}$	2
C	$\theta_i = 3x_{1i} + x_{2i}$	continuous

ates contain any useful information regarding survival expectation. Model B produces failure times originating from one of two survival distributions with $\log(\lambda)$ either 0 or 1 depending on a boolean combination of two of the five covariates. Lastly, model C uses exponential distributions with continuously varying intensity parameters depending on x_1 and x_2 in an additive

fashion. Populations of sample size $n = 500$ were generated for each model. The training sets were then used to construct survival trees using the log-rank test as a split criterion, enforcing a minimum node size of $n_{min} = 25$ and pruning the trees via ten-fold cross-validation. The whole process was repeated 1000 times for each model, and the resulting number of terminal nodes was recorded for every tree. Table 4 shows the relative frequencies obtained for the 1000 runs on each of the tree models.

As hoped for with model A data, the tree realizes there is nothing to split

Table 4: Relative frequencies of terminal nodes for 1000 trees for each model.

	relative frequency of terminal nodes					
Model	1	2	3	4	5	≥ 6
A	96.0	1.2	2.8	0	0	0
B	32.0	2.8	38.0	14.4	6.4	6.4
C	0	0	3.6	12.8	14.0	69.6

the data on in more than 95% of the runs. Still, structure is suggested in 4% of the simulations eventhough none is present. As for model B, the difficulty recursive partitioning procedures have with boolean combinations becomes apparent in the second row of table 4. One would hope for a large number of trees with three terminal nodes, accepting the fact that the breakup of the boolean combination requires one additional split and thus produces an extra terminal node. Although this does occur in about 40% of the runs, the tree still comes up with an unsatisfactory answer in the rest of the simulations. On half of these, the tree doesn't detect any structure in the data at all after pruning. Finally, the results for model C show the tree's clumsy attempts to sequentially dichotomize a linear relationship in a stepwise fashion.

Thus, there is clearly a large amount of variability contained in tree-based predictors and one is worried that this instability will degrade predictive accuracy in addition to hindering correct conclusions about factor importance

and adequate cutpoints. In what follows we describe two attempts to capture this variability and to stabilize tree-based predictors by averaging or aggregating.

5.1 Tree-based predictor aggregation via the bootstrap

Demonstrating that trees are high variance, low bias procedures, Breiman (1996) suggests growing numerous trees using a series of training sets and then suitably aggregating these to form a single, stabilized predictor.
More formally, starting off with a series of learning samples $L^{(r)}$ with $r = 1, \ldots, R$, using $R = 50$ for example, Breiman constructs R trees $d^{(r)}$ using identical growing and pruning parameters which he then combines. The exact method of aggregation depends on the response type. For the regression case, the arithmetic mean is used, so that the aggregated predictor d_A is simply $\frac{1}{R}\sum_{r=1}^{R} d^{(r)}$, while for the classification case a simple voting procedure is invoked, so that d_A assigns that class $\hat{y}$ to an element which was predicted most often in the R single predictors.
Using a bias–variance decomposition of prediction error and employing Jensen's inequality, Breiman is able to show that the prediction error of the aggregated predictor possesses the prediction error of the single predictor as an upper bound. In other words, the aggregated predictor is always at least as good as the single one.
For this to work, one needs a series of training sets drawn independently from the population Ω. Since this luxury is never available save for simulations, Breiman uses bootstrap replicate training sets drawn form the original training set L as a substitute for repeatedly drawing from Ω. Each of these replicate samples are then used to grow trees, which are aggregated as above. Eventhough now the prediction error of the single predictor need no longer be worse than that of the bootstrap aggregated predictor, bagging performs remarkably well in most situations. Only in cases where the single predictor

is extremely stable will bagging decrease performance.

To demonstrate bootstrap aggregation, we return to our survival simulations from section 5. Survival trees were grown on data generated from models A, B and C. The trees were pruned using ten-fold cross-validation. Afterwards, bagged predictors were constructed using $R = 50$ bootstrap replicates for each model. To compare predictor performance, the mean squared error between predicted and observed failure times was computed. Table 5 shows the results for the three situations. As expected for model A, bagging slightly decreases tree performance, whereas for models B and C, bootstrap aggregation is able to adequately average out the variability of single tree–based predictors, resulting in moderate improvements.

Predictor aggregation procedures can be improved upon by invoking an adaptive resampling approach, where attention on stabilization is focused on those regions of the training set that led to poor or variable predictions in the single predictor. Freund and Schapire (1996) discuss some possibilities.

Table 5: Comparison of single, tree predictors and bagged predictors for survival data.

Model	$MSE_{single}(t, \hat{t})$	$MSE_{aggregated}(t, \hat{t})$	Change
A	1.004	1.017	+1.3%
B	0.883	0.861	−2.5%
C	0.80	0.74	−8.1%

There are drawbacks to bagging or predictor aggregation in general. In the tree–based context, bagging requires individual (subject) specific estimates of the response. While these are readily available in classification and regression settings, it is not immediately clear how this concept can be transferred to a general survival setting with censored data, where population averaged estimates and characteristics are usually provided.

The main problem with predictor aggregation is the loss of an intuitively

structured, simple predictor. This consequence becomes especially apparent for trees, as there is simply no tree to display for the aggregated predictor. Thus, while prediction is usually improved, understanding how the predictor reaches its conclusions is hindered. This is a drastic drawback in disciplines such as medicine, where improving understanding which prognostic factors influence prognoses is one of the main goals.
To alleviate this problem, Wernecke, Possinger and Kalb (1996) suggest what amounts to counting and weighting specific split occurrences over numerous (cross-validated) tree replicates, but at the cost of only allowing dichotomous factors in the first place, thus reducing recursive partitioning to a variable selection and interaction detection method.
In the next section we outline an approach that can potentially reduce predictive error by stabilizing individual splits, while at the same time keeping the simple structure of a single tree–based predictor.

5.2 Node-level stabilization via the bootstrap

Stabilizing tree–based predictors by aggregating several slightly modified versions leads to marked improvements in their predictive accuracy. The loss of simple structure however can be anything from a nuisance to a major problem, especially when focus is on understanding as well as on making decisions. As was demonstrated in section 3, alot of the variability of trees stems from the instability at the node–level, when selecting the factor and cutpoint to split a population on. Thus if more stable splits could be found, one could reduce variability while keeping the single tree.
Here we try an approach along these lines for the binary classification simulation of section 3.3. Using 100 bootstrap replicates of any node population, a simple voting procedure is used to determine which factor is actually used to split the node. Accordingly, the optimal cut-point is determined by taking the median of the replicate cut-points for the chosen factor. Table 6 contrasts estimates of average classification errors for a single tree, a bootstrap

aggregated predictor and a tree based on node-level stabilization measures described above. Numbers shown reflect averages over 50 independent runs of the same procedure. Both optimistically biased resubstitution estimates and errors for an independently drawn validation sample are shown. We rely

Table 6: Average misclassification rates (50 runs).

Method	Resubstitution estimate	validation sample
single tree	0.18333	0.26067
bagging	0.01667	0.03533
node resampling	0.04333	0.06333

on the validation sample to compare performance. Note how the bootstrap aggregated predictor leads to a drastic reduction in prediction error from 26% to less than 4%, but that node-level stabilization also reduces the error to a competitive 6%. In addition, the node-level stabilized tree has kept its simple, single tree structure and thus can easily be interpreted and communicated.

It must be mentionend that in contrast to bagging, node-level stabilization can also lead to marked increase in errors in some situations, so care needs to be taken when using this approach. More work needs to be done to determine an optimal stabilized split, perhaps by making increased use of information obtained by the diagnostic tools introduced earlier. Still, this example demonstrates that approaches to stabilize trees without sacrificing their simple structure are feasible.

6 Conclusion

We have supplied and reviewed methods for assessing and improving tree performance. The simple diagnostics tools introduced allow thorough analyses of split and factor importance stability. The general concept of obtaining p-values via permutation techniques has proven to be flexible and computationally feasible.

Predictor aggregation works well for improving performance, but the loss of simplicity is a severe problem, at least in the medical setting. The concept of "representer trees" in Breiman and Shang (1996), wherein trees are used to produce structurally understandable representations of arbitrarily complex predictors may reduce this problem, but was not analyzed here.

Node-level stabilizing procedures appear to have potential but haven't been extensively studied yet. Another alternative to reduce variability caused by artificial dichotomizations of metric factors could be to allow non-binary splits, that is instantaneous partitions of a node into more than two subgroups, when the data indicate such a split to be called for. Here as with node-level stabilizing procedures, there appears to be a need to extend recursive partitioning procedures to look ahead more than one split at a time. While computationally demanding, this approach should be able to better determine whether a certain split really is preferable over another one in the "long run" or further down in the tree, eventually leading to better prediction.

References

Breiman, L. (1996). Bagging Predictors, *Machine Learning* **26**, 123–140.

Breiman, L. and Shang, N. (1996). Born again trees, *Technical report*, University of California, Berkeley, California.

Breiman, L., Friedman, J., Olshen, R. and Stone, C. (1984). *Classification and Regression Trees*, Chapman and Hall, New York.

Ciampi, A., Chang, C. H., Hogg, S. and McKinney, S. (1987). Recursive Partition: A versatile method for exploratory data analysis, *Festschrift in Honor of Professor V.M. Joshis's 70th Birthday*, Vol. V: Biostatistics, D. Reidel Publishing Company, Dordrecht.

Freund, Y. and Schapire, R. (1996). Experiments with a new boosting algorithm, *Machine Learning: Proceedings of the Thirteenth International Conference*. To appear.

Hilsenbeck, S. G. and Clark, G. M. (1996). Practical p-value adjustment for optimally selected cutpoints, *Statistics in Medicine* **15(1)**, 103–112.

LeBlanc, M. and Crowley, J. (1992). Relative Risk Trees for Censored Data, *Biometrics* **48**, 411–425.

LeBlanc, M. and Crowley, J. (1993). Survival Trees by Goodness of Split, *Journal of the American Statistical Association* **88**, 457–467.

Wernecke, K. D., Possinger, K. and Kalb, G. (1996). Zur Validierung von Klassifikationsbäumen, *Technical report*, Humboldt-Universität Berlin.

Zhang, H., Crowley, J., Sox, H. C. and Olshen, R. A. (1997). Tree-structured Statistical Methods, *Encyclopedia of Biostatistics*. To appear.