An information criterion for automatic gradient tree boosting

Berent Å. S. Lunde [*] Tore S. Kleppe [†] Hans J. Skaug [‡]

August 13, 2020

Abstract

An information theoretic approach to learning the complexity of classification and regression trees and the number of trees in gradient tree boosting is proposed. The optimism (test loss minus training loss) of the greedy leaf splitting procedure is shown to be the maximum of a Cox-Ingersoll-Ross process, from which a generalization-error based information criterion is formed. The proposed procedure allows fast local model selection without cross validation based hyper parameter tuning, and hence efficient and automatic comparison among the large number of models performed during each boosting iteration. Relative to `xgboost`, speedups on numerical experiments ranges from around 10 to about 1400, at similar predictive-power measured in terms of test-loss.

1 Introduction

This article is motivated by the problem of selecting the functional form of trees and ensemble size in gradient tree boosting (Friedman, 2001; Mason et al., 2000). Gradient tree boosting (GTB) has become extremely popular in recent years, both in academia and industry: At present, an increase in the size of datasets, both in the number of observations and the richness of the data, or number of features, is seen. This, coupled with an exponential increase in computational power and a growing revelation and acceptance for data-driven decisions in the industry makes for an increasing interest in statistical learning (Hastie et al., 2001). For these new datasets, standard statistical methods such as generalized linear models (McCullagh and Nelder, 1989) that have a fixed learning rate due to their constrained functional form with bounded complexity, struggle in terms of predictive power, as they stop learning at certain information thresholds. The interest is therefore geared towards more flexible approaches such as ensembles of learners.

[*]Department of Mathematics and Physics, University of Stavanger, 4036 Stavanger, Norway. Tel.: +47-47258605. berent.a.lunde@uis.no

[†]Department of Mathematics and Physics, University of Stavanger, 4036 Stavanger, Norway

[‡]Department of Mathematics, University of Bergen, 5020 Bergen, Norway

GTB has recently risen to prominence for structured or tabular data, and previous to this, the related random forest algorithm (Ho, 1995; Breiman, 2001) was the "off-the-shelf" machine learning algorithm of choice for many practitioners. They both perform automatic variable selection, there is a natural measure of feature importance, they are easy to combine, and simple decision trees are often easy to interpret. In fact, gradient tree boosting has dominated in machine-learning competitions for structured data since around 2014 when the `xgboost` implementation (Chen et al., 2018; Chen and Guestrin, 2016) was made popular. Recent years have seen the introduction of rivalling implementations such as `LightGBM` (Ke et al., 2017) and `CatBoost` (Dorogush et al., 2018).

A difficulty with GTB is that it is prone to overfitting: The functional form changes for every split in a tree, and for every tree that is added. Hence, it is necessary to constrain the ensemble size and the complexity of each individual tree. Standard practice is either the use of a validation set, cross-validation (Stone, 1974), or regularization to target a bias-variance trade-off (Hastie et al., 2001). Friedman (2001) suggested a constant penalisation of each split, while later implementations have also introduced L2 and L1 regularisation. All the above mentioned GTB implementations have many hyper-parameters, which must be tuned in a computationally expensive manner, typically involving cross-validation. We will collectively view these measures to avoid overfitting as solutions to a model selection problem.

In this article we take an information theoretic approach to GTB model selection, as an alternative to cross-validation. Building on the seminal work of Akaike (1974) and Takeuchi (1976) we approximate the difference between test and training error for each split in the tree growing process. This difference, known as the "optimism" (Hastie et al., 2001), is used to formulate new stopping criteria in the GTB algorithm, both for tree growing and for the boosting algorithm itself. The resulting algorithm selects its model complexity in a single run, and does not require manual tuning. We show that it is considerable faster than existing GTB implementations, and we argue that it lowers the bar for applications by non-expert users.

The following section introduces gradient tree boosting. We then discuss model selection and develop an information theoretic approach to gradient boosted trees, and comment on evaluation using asymptotic theory together with modifications of the GTB algorithm. Section 4 is concerned with validation through simulation experiments of the theoretical results in section 3. Section 5 sees applications to real-data and comparisons with competing methodologies. Proofs of the theoretical results in section 3 may be found in the Appendix.

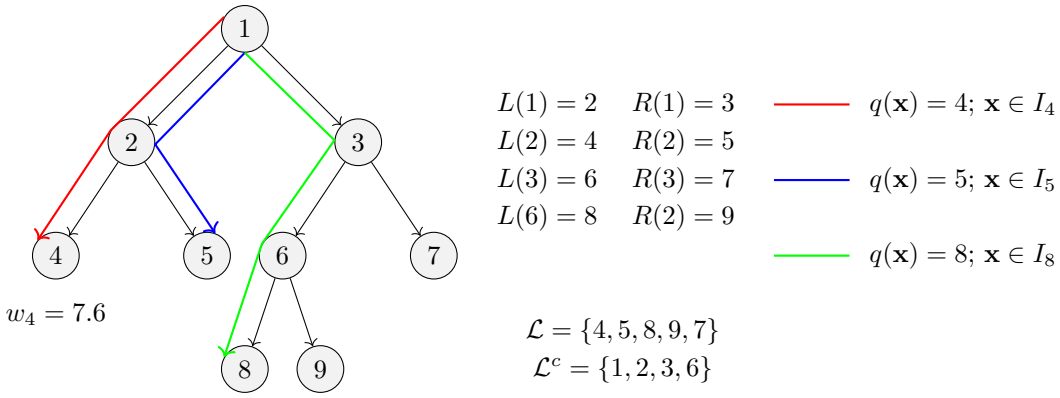

Figure 1: Example of a CART with $T = 5$ leaf nodes ($\mathcal{L}$) and 4 internal nodes ($\mathcal{L}^c$). $\mathbf{w} = (w_4, w_5, w_7, w_8, w_9)$ is the vector of possible predictions. The operator $q(\mathbf{x})$ maps different instance sets ($I_t,\ t \in \mathcal{L}$) to leaf nodes. The mappings $L(t)$ and $R(t)$ yield the left and right descendants of each internal node $t \in \mathcal{L}^c$.

2 Gradient tree boosting

Let $\mathbf{x} \in \mathbb{R}^m$ be a feature vector and $y \in \mathbb{R}$ a corresponding response variable. The objective of supervised learning in general is to determine the function $f(\mathbf{x})$ that minimises the expected loss,

$$\hat{f} = \arg\min_f E_{\mathbf{x},y}\left[l(y, f(\mathbf{x}))\right], \tag{1}$$

given a loss function $l(\cdot, \cdot)$. In practice, the expectation over the joint distribution of $\mathbf{x}$ and y must be replaced by an empirical average over a finite dataset, $\mathcal{D}_n = \{(\mathbf{x}_i, y_i)\}$, $|\mathcal{D}_n| = n$, $\mathbf{x}_i \in \mathbb{R}^m$, $y \in \mathbb{R}$. The loss, l, is a function that measures the difference between a prediction $\hat{y}_i = f(\mathbf{x}_i)$ and its target y_i. We will assume that l is both differentiable and convex in its second argument.

In GTB, f is taken to be an ensemble model, with ensemble members $f_k(\mathbf{x})$ being classification and regression trees (CARTs; see Figure 1 for notation). A prediction from f has the following form:

$$\hat{y}_i = f^{(K)}(\mathbf{x}_i) = \sum_{k=1}^{K} f_k(\mathbf{x}_i), \quad \text{where } f_k(\mathbf{x}_i) = w_{q_k(\mathbf{x}_i),k}. \tag{2}$$

Here, $q_k : \mathbb{R}^m \to \mathcal{L}_k$ (where $\mathcal{L}_k$ is the set of leaf nodes) is the feature mapping of the k'th tree, which assigns every feature vector to a unique leaf node (see Figure 1). The predictions associated with each leaf node are contained in a vector $\mathbf{w}_k = \{w_{t,k},\ t \in \mathcal{L}_k\} \in \mathbb{R}^{T_k}$, where T_k is the number of leaf nodes in the k-th tree (i.e. the cardinality of $\mathcal{L}_k$). Moreover, any internal node t (i.e. $t \in \mathcal{L}_k^c$) has exactly two descendants whose labels are denoted by $L(t)$ (left descendant) and $R(t)$ (right descendant). Figure 1 illustrates these concepts graphically for three different input feature-vectors.

Suppose an ensemble model with $k-1$ trees, $f^{(k-1)}$, has already been selected. In order to sequentially improve the ensemble prediction by adding another member f_k, the theoretical objective $E_{\mathbf{x},y}\left[l(y, f^{(k)}(\mathbf{x}))\right]$ reduces to

$$E_{\mathbf{x},y}\left[l\left(y, f^{(k-1)}(\mathbf{x}) + f_k(\mathbf{x})\right)\right], \tag{3}$$

which should be minimized with respect to the q_k and $\mathbf{w}_k$ associated with f_k. To gain analytical tractability we perform a second order Taylor expansion around $\hat{y} = f^{(k-1)}(\mathbf{x})$:

$$\hat{l}(y, \hat{y} + f_k(\mathbf{x})) = l(y, \hat{y}) + g(y, \hat{y}) f_k(\mathbf{x}) + \frac{1}{2} h(y, \hat{y}) f_k^2(\mathbf{x}), \tag{4}$$

where $g(y, \hat{y}) = \frac{\partial}{\partial \hat{y}} l(y, \hat{y})$ and $h(y, \hat{y}) = \frac{\partial^2}{\partial (\hat{y})^2} l(y, \hat{y})$.

As the joint distribution of $(\mathbf{x}, y)$ is generally unknown, the expectation in (3) is approximated by the training data empirical counterpart:

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(k-1)} + f_k(\mathbf{x}_i)\right) &\approx \frac{1}{n}\sum_{i=1}^{n}\left[l\left(y_i, \hat{y}_i^{(k-1)}\right) + g_{ik} f_k(\mathbf{x}_i) + \frac{1}{2} h_{ik} f_k(\mathbf{x}_i)^2\right] \\
&= \frac{1}{n}\sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(k-1)}\right) + \frac{1}{n}\sum_{t\in\mathcal{L}_k}\left[\sum_{i\in I_{tk}} g_{ik} w_{tk} + \frac{1}{2} h_{ik} w_{tk}^2\right] \qquad (5) \\
&=: \ell_k(q_k, \mathbf{w}_k). \qquad (6)
\end{aligned}$$

where

$$g_{ik} = g(y_i, f^{(k-1)}(\mathbf{x}_i)) \qquad \text{and} \qquad h_{ik} = h(y_i, f^{(k-1)}(\mathbf{x}_i)). \tag{7}$$

and I_{tk} is the instance set of leaf t: $I_{tk} = \{i : q_k(\mathbf{x}_i) = t\}$, (see Figure 1). Hence, ℓ_k is the *training* loss approximation of the theoretical objective (3), to be optimized in the k-th boosting iteration. This second order approximation-based boosting strategy was originally proposed by Friedman et al. (2000) and first implemented for CARTs in `xgboost` Chen and Guestrin (2016). Further, notice that for the quadratic loss $l(y, \hat{y}) = (y - \hat{y})^2$, the Taylor expansion is exact.

For a *given* feature mapping q_k (and hence instance sets $I_{tk},\ t \in \mathcal{L}_k$), the weight estimates $\hat{\mathbf{w}}_k$ minimizing $\mathbf{w}_k \mapsto \ell_k(q_k, \mathbf{w}_k)$ are given by

$$\hat{w}_{tk} = -\frac{G_{tk}}{H_{tk}},\ G_{tk} = \sum_{i\in I_{tk}} g_{ik},\ H_{tk} = \sum_{i\in I_{tk}} h_{ik}. \tag{8}$$

Further, the improvement in training loss resulting from using weights (8) is given by

$$\ell_k(q_k, \hat{\mathbf{w}}) - \frac{1}{n}\sum_{i=1}^{n} l(y_i, \hat{y}_i^{(k-1)}) = -\frac{1}{2n}\sum_{t=1}^{T_k} \frac{G_{tk}^2}{H_{tk}}. \tag{9}$$

The explicit expressions for leaf weights (8) and loss reduction (9) allow comparison of a large number of different candidate feature maps q_k. Still, to consider every possible tree structure leads to combinatorial explosion, and it is therefore customary to do recursive binary splitting in a greedy fashion (p. 307 Hastie et al., 2001; Chen and Guestrin, 2016):

1. Begin with a constant prediction for all features, i.e. $\hat{w} = -\frac{\sum_{i=1}^{n} g_{ik}}{\sum_{i=1}^{n} h_{ik}}$, in a root node.

2. Choose a leaf node t. For each feature j, compute the training loss reduction

$$\mathcal{R}_t(j, s_j) = \frac{1}{2n}\left[\frac{\left(\sum_{i\in I_L(j,s:j)} g_{ik}\right)^2}{\sum_{i\in I_L(j,s_j)} h_{ik}} + \frac{\left(\sum_{i\in I_R(j,s_j)} g_{ik}\right)^2}{\sum_{i\in I_R(j,s_j)} h_{ik}} - \frac{\left(\sum_{i\in I_{tk}} g_{ik}\right)^2}{\sum_{i\in I_{tk}} h_{ik}}\right], \tag{10}$$

   for different split-points s_j, and where $I_L(j, s_j) = \{i \in I_{tk} : x_{ij} \leq s_j\}$ and $I_R(j, s_j) = \{i \in I_{tk} : x_{ij} > s_j\}$. The values of j and s_j maximizing $\mathcal{R}_t(j, s_j)$ are chosen as the next split, creating two new leaves from the old leaf t.

3. Continue step 2 iteratively, until some threshold on tree-complexity is reached.

Notice that $\mathcal{R}_t(j, s_j)$ is the difference in training loss reduction (9) between 1) a tree where t is a leaf node and 2) otherwise the same tree, but where t is the ancestor to two leaf nodes $L(t)$, $R(t)$ split on the jth feature. In particular, $\mathcal{R}_t(j, s_j)$ depends only on the data that are passed to node t.

The measures of tree-complexity in step 3 vary, and multiple criteria can be used at the same time, such as a maximum depth, maximum terminal nodes, minimum number of instances in node, or a regularized objective. Also, several alternative strategies for choosing candidate t and proposal s_js in step 2 exist, (see e.g Chen and Guestrin, 2016; Ke et al., 2017). A typical strategy is to build a very large tree, and then *prune* it back to a subtree using cost complexity pruning (Hastie et al., 2001, p. 308).

Algorithm 1 illustrates the full second order gradient tree boosting process with CART trees and several split-stopping criteria. Note an until now unmentioned hyperparameter, the "learning rate" $\delta \in (0, 1]$. The learning rate (or shrinkage (Friedman, 2002)) shrinks the effect of each new tree with a constant factor in step 2.*iv*), and thereby opens up space for feature trees to learn. This significantly improves the predictive power of the ensemble, but comes at the cost of more boosting iterations until convergence. Note how the special case of $\delta = 1$ and $K = 1$ gives a decision tree, and $\delta \to 0$ and $K \to \infty$ potentially gives a continuous

model.

Algorithm 1 Original (Hastie et al., 2001; Chen and Guestrin, 2016) and modified second order gradient tree boosting.

Input:
- A training set $\mathcal{D}_n = \{(x_i, y_i)\}_{i=1}^n$
- A differentiable loss function $l(\cdot, \cdot)$
- A learning rate $\delta \in (0, 1]$
- Number of boosting iterations K
- One or more tree-complexity regularization criteria

1. Initialize model with a constant value:
$f^{(0)}(\mathbf{x}) \equiv \arg\min_{\eta} \sum_{i=1}^n l(y_i, \eta)$
2. **for** $k = 1$ to K: **while** the inequality (29) evaluates to `false`
i) Compute derivatives (7)
ii) Determine the structure q_k by iteratively selecting the binary split that maximizes (10) until
a regularization criterion is reached. the inequality (28) evaluates to `true` for all leaf nodes t
iii) Determine leaf weights (8), given q_k
iv) Scale the tree with the learning rate
$f_k(\mathbf{x}) = \delta w_{q_k(\mathbf{x})}$
v) Update the model:
$f^{(k)}(\mathbf{x}) = f^{(k-1)}(\mathbf{x}) + f_k(\mathbf{x})$
end for **while**
3. Output the model: **Return** $f^{(K)}$

Blue background colour signifies steps unique to the original algorithm, while orange signifies steps unique to the modified algorithm proposed here.

3 Information theoretic approach to gradient boosted trees

3.1 Model selection problem

In the GTB Algorithm 1, there are two places where decisions are made with respect to the functional form of $f^{(k)}$:

- in step 2.*ii*), decisions must be made whether to perform the proposed leaf splits, i.e. sequential decisions with respect to the feature map q_k.
- in step 2.*v*) a decision must be made whether to add f_k to $f^{(k-1)}$, or otherwise to terminate the algorithm, i.e. selecting the number of boosting iterations K.

The overarching aim of this paper is to develop automatic and computationally fast methodology for performing such decisions while minimizing the generalization error. Suppose the model $f(\mathbf{x}; \theta)$ depends on

some parameters θ, and a procedure for fitting θ to the training data, say $\hat{\theta} = \hat{\theta}(\mathcal{D}_n)$ is given. Further, let $(\mathbf{x}^0, y^0)$ be a test-data realization with the same distribution as each $(\mathbf{x}_i, y_i) \in \mathcal{D}_n$, unseen in the training phase and hence independent from $\hat{\theta}$. We will use

$$Err = E_{\hat{\theta}} E_{\mathbf{x}^0, y^0} \left[l \left(y^0, f(\mathbf{x}^0; \hat{\theta}) \right) \right]. \tag{11}$$

as our measure of generalization error, as it is well suited for analytical purposes.

In GTB described above, it is not the generalization error that is used when comparing possible splits in step 2 in the greedy binary splitting procedure. Equations (9,10) are estimators (modulo errors introduced by the Taylor expansions) of *reduction in training loss*, where the training loss is given by:

$$\overline{err} = \frac{1}{n} \sum_{i=1}^{n} l(y_i, f(\mathbf{x}_i; \hat{\theta})). \tag{12}$$

As is well known, $\overline{err}$ as an estimator for Err is biased downwards in expectation, favouring complex models which leads to overfitting.

The bias of (12) relative to (11) is commonly referred to as the *optimism* of the estimation procedure (Hastie et al., 2001). The reminder of this section is devoted to deriving estimators of such optimism in the GTB context, and subsequently using these to obtain optimism-corrected estimators of $\overline{err}$.

3.2 Correcting the training loss for optimism

Define the conditional on feature j reduction in training loss

$$\mathcal{R}_t(j) = \max_{s_j} \mathcal{R}_t(j, s_j), \ j = 1, \ldots, m, \tag{13}$$

and unconditional reduction in training loss

$$\mathcal{R}_t = \max_{j \in (1, \ldots, m)} \mathcal{R}_t(j), \tag{14}$$

where the reduction in training loss $\mathcal{R}_t(j, s_j)$ for given ancestor node t, feature j and split point s_j is given in (10). A key part of our approach is to derive estimators of the generalization-loss based counterparts of $\mathcal{R}_t(j)$ and $\mathcal{R}_t$ to (12), which we denote by $\mathcal{R}_t^0(j)$ and $\mathcal{R}_t^0$, respectively. In the current section we focus on $\mathcal{R}_t^0(j)$, while $\mathcal{R}_t^0$ will be considered in Section 3.5.

The proposed estimator of $\mathcal{R}_t^0(j)$, and hence that of $\mathcal{R}_t^0$, does not rely on cross validation or bootstrapping, but rather on analytical results adapted from traditional information theory. The approach enables

learning of the feature maps q_k, and also suggests a natural stopping criterion for boosting iterations. The algorithm is terminated when splitting the root node is not beneficial. This is automatic and with minimal worries of overfitting.

As should be clear from Algorithm 1, only (local) splitting decisions on a single leaf node are performed in each step. Moreover, the splitting decisions on two distinct leaf nodes do not influence each other as different subsets of the data are passed to the respective leafs. In the presentation that follows, we therefore focus on estimating $\mathcal{R}_t^0(j)$ for a split/no-split decision of a single leaf node. To avoid overly complicated notation, we consider the *root node only*, i.e. $t = 1$, and subsequently suppress the ancestor index t. This simplification introduces no loss of generality, as the split/no-split decisions at any leaf node are exactly the same, except that they only operate on the subsets of the original data passed to that leaf node.

With the understanding that j is fixed in this section, we suppress the index j from our notation, except when strictly needed for future reference. The no-split decision involves a *root* tree, consisting of a single node with prediction $\hat{w}_1 = -\sum_{i=1}^n g_i / \sum_{i=1}^n h_i$. The do-split decision involves a *stump* tree, with two leaf nodes and parameters $\hat{\theta} = \{\hat{s}, \hat{w}_l, \hat{w}_r\}$. Here, s is the split point (for the jth feature) and $\hat{w}_l$ and $\hat{w}_r$ are the leaf weights of the left and right leaf nodes, respectively, given by (8).

The subsequent theory is derived using the 2nd order Taylor approximation $\hat{l}$, given by (4), instead of the original loss l.

In what follows, we seek an adjustment of the training loss reduction $\mathcal{R}(j)$ defined in (13) to represent $\mathcal{R}^0(j)$ in expectation over the training data. The optimism C for the constant (root) model is defined as

$$C_{\text{root}} = E_{\hat{w}} E_{y^0} \left[\hat{l}\left(y^0, \hat{w}\right) \right] - E_{\mathbf{y}} \left[\hat{l}(y_1, \hat{w}) \right], \tag{15}$$

where $\mathbf{y} = (y_1, \ldots, y_n)$ and $\hat{w} = \hat{w}(\mathbf{y}) = -\sum_{i=1}^n g_i / \sum_{i=1}^n h_i$. The use of y_1 in the last term above is justified by the fact that the $(y_i, \mathbf{x}_i)$ are identically distributed for $i = 1, \ldots, n$. Note that C_{root} does not depend j as the root model does not utilize any feature information. For the tree-stump (stump) we get

$$C_{\text{stump}}(j) = E_{\hat{\theta}} E_{\mathbf{x}^0, y^0} \left[\hat{l}(y^0, f(\mathbf{x}^0; \hat{\theta})) \right] - E_{\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{y}} \left[\hat{l}(y_1, f(\mathbf{x}_1; \hat{\theta})) \right]. \tag{16}$$

where $\hat{\theta} = \hat{\theta}(\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{y}) = \{\hat{s}, \hat{w}_l, \hat{w}_r\}$. When interpreting (16) it should be kept in mind that we are currently only using the jth component of the feature vector $\mathbf{x}$.

Equations (15) and (16) may be combined to get an equivalent representation of $\mathcal{R}^0(j)$ expressed in terms

of expected reduction in training loss (i.e. (13) in expectation), namely

$$E_{y^0,x^0}[\mathcal{R}^0(j)] = E_{\mathbf{x},y}[\mathcal{R}(j)] + C_{\text{root}} - C_{\text{stump}}(j). \tag{17}$$

Under the assumption that the j-th feature is independent of the response y, each term on the right hand side of (17) may be estimated efficiently and consequently allows us to correct the training loss reduction. In practice, $E_{\mathbf{x},y}[\mathcal{R}(j)]$ is estimated using the observed training loss reduction $\mathcal{R}(j)$. Hence, similarly to conventional hypothesis testing, if the estimated version of (17) is negative we retain root model, whereas if the estimated (17) as a consequence of a large training loss difference is positive, we opt for the stump model. The next few sections are devoted to derive approximations for the optimisms C_{root} and $C_{\text{stump}}(j)$, and constitute the main methodological contribution of the paper.

3.3 Optimism for loss differentiable in parameters

In the standard case where some loss function l is differentiable in its parameters, say η, and adhere to the regularity conditions in A the optimism may be estimated by (Burnham and Anderson, 2003, Eqn. 7.32):

$$\tilde{C} = \text{tr}\left(E_{\mathbf{x},y}\left[\nabla^2_{\eta_0} l(y, f(\mathbf{x};\eta_0))\right] Cov\left[\hat{\eta}\right]\right), \tag{18}$$

where $\eta_0 = \lim_{n\to\infty}\hat{\eta}$. If $Cov\left[\hat{\eta}\right]$ is estimated using the Sandwich Estimator (Huber et al., 1967; White, 1982) one obtains the network information criterion (NIC) (Murata et al., 1994). The training loss of the stump model is discontinuous in s_js for finite n, and hence (18) is not applicable in the s_j-direction.

Again taking the local perspective, we omit dependence on being in node t. We start off with considering an optimism approximation for C_{root}, say $\tilde{C}_{root}$, which subsequently will be used in (18). Moreover $\tilde{C}_{root}$ constitute a building block for our approximation to C_{stump}. The root-model does not involve any split-points, and hence when (18) is applied we obtain:

$$\tilde{C}_{\text{root}} = E_{\mathbf{x},y}\left[\frac{1}{n}\sum_{i=1}^{n} h_i\right] Var\left(\hat{w}_1\right). \tag{19}$$

Turning to the stump-model, suppose momentarily that the split-point s_j is given a-priori. Then we may compute the optimism approximation (19) for the tree-stump model (conditioned on s_j), say $\tilde{C}_{\text{stump}}(j|s_j)$, which is given by

$$\tilde{C}_{\text{stump}}(j|s_j) = \tilde{C}_{\text{root},L} P(x_j \leq s_j) + \tilde{C}_{\text{root},R} P(x_j > s_j). \tag{20}$$

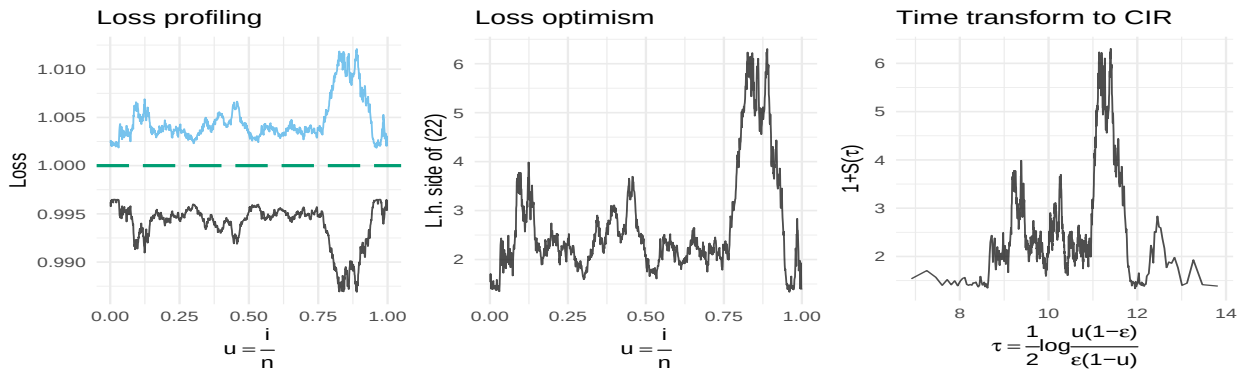

Figure 2: Left: Training loss $\hat{l}(y, f(x;\hat{\theta}))$ (black) and generalization loss $E_{y^0,x^0}\left[\hat{l}(y^0, f(x^0;\hat{\theta}))\right]$ (blue) as a function of $u = \frac{i}{n}$, defined from the sorted order of x_j. Green long-dashed line is the expected loss-value at $\theta_0 = \lim_{n\to\infty}\hat{\theta}_n$ if $n \to \infty$ for the training data, constant as there is no information in x_j for this instance. Right plot: The transformation of distance between generalization loss and training loss into a CIR process, with $\epsilon = 10^{-7}$ which is used throughout. In this case with no information in feature x_j, choosing the value of s giving the smallest value of training loss in the left plot induces an optimism at the value of 1 plus the expected maximum of the CIR-process, illustrated in the right plot.

Here $\tilde{C}_{\text{root},L}$ and $\tilde{C}_{\text{root},R}$ are as in (19) but computed from sub-datasets corresponding to the child leaf nodes L (i.e. $i : x_{i,j} \leq s_j$) and R (i.e. $i : x_{i,j} > s_j$)) respectively. $\tilde{C}_{\text{stump}}(j|s_j)$, however, cannot be substituted in (17) directly, since the optimism induced by optimizing over s_j is not accounted for in (20). Consequently $\tilde{C}_{\text{stump}}(j|s_j)$ will be downward biased relative to $C_{\text{stump}}(j)$. The next section attempts to account for this bias by providing an approximate correction factor.

As a side-note, notice that (18) may be applied more generally to a full tree, if the structure q is given a-priori. In this case, the optimism of the full tree may be approximated by

$$\sum_{t=1}^{T} \tilde{C}_{root,t} P(q(x) = t) \tag{21}$$

Again, $\tilde{C}_{root,t}$ is computed as in (19), based on the data that is passed in leaf-node t, i.e. $i \in I_t$. Of course, (21) is also biased downward relative to the optimism of the tree when q is learned from training data.

3.4 Optimism from greedy-splitting over one feature

In order to resolve $C_{\text{stump}}(j)$ appearing in (17), this section provides an approximation $\tilde{C}_{\text{stump}}(j)$ which in general is biased upward relative to $C_{\text{stump}}(j)$. Consequently, the approximation of $\mathcal{R}^0(j)$ resulting from substituting $C_{\text{stump}}(j)$ with $\tilde{C}_{\text{stump}}(j)$ is biased downward, in practice favouring the constant model. However, it is illustrated in the simulation experiments in Section 4 that this bias is rather small. In order to construct $\tilde{C}_{\text{stump}}(j)$, we first assume that y is independent of $x_j = x_{\cdot,j}$. This assumption appears to be

necessary to get an asymptotic approximation to the joint distribution of the difference in test and training loss (which in expectation over training data is the conditional optimism) for different values of split points. This distribution obtains as the limiting distribution of an empirical process. The argument leading to this limiting distribution has similarities to the one originally presented in Miller and Siegmund (1982) regarding maximally selected chi-square statistics, and generalized with refinements in Gombay and Horvath (1990).

Suppose the training data $(y_i, x_{i,j})$ has been sorted in ascending order over i according to the j-th feature. If x_j contains repeated values, the ordering (in i) of observations with identical $x_{i,j}$ is arbitrary. Further, define $u_i := i/n$, and let $f(\cdot; \hat{w}_l(u_i), \hat{w}_r(u_i))$, $i = 1, \ldots, n-1$, be the tree stump with left node containing $x_{1:i,j}$, right node containing $x_{(i+1):n,j}$ (and hence split point $s = x_{i,j}$). Notice that $\lim_{n\to\infty} u_i = p(x_{\cdot,j} \leq x_{i,j})$. Under the independence assumption, the difference between generalization loss and training loss as a function of i converges in distribution as

$$n\left[E_{y^0,x^0}\left[\hat{l}(y^0, f(\mathbf{x}^0; \hat{w}_l(u_i), \hat{w}_r(u_i)))\right] - \hat{l}(y, f(\mathbf{x}; \hat{w}_l(u_i), \hat{w}_r(u_i)))\right] \quad \xrightarrow[n\to\infty]{D} \quad n\hat{C}_{\text{root}}\left(1 + S(\tau(u))\right), \quad (22)$$

where $n\hat{C}_{\text{root}} = O(1)$. Here $S(\tau)$ is defined through the stochastic differential equation

$$dS(\tau) = 2(1 - S(\tau))d\tau + 2\sqrt{2S(\tau)}dW(\tau). \quad (23)$$

Moreover, $W(\tau)$ is a Wiener process with time τ following $\tau = \frac{1}{2}\log\frac{u(1-\epsilon)}{\epsilon(1-u)}$, $u = \min\left\{1-\epsilon, \max\left\{\epsilon, \frac{i}{n}\right\}\right\}$ and $0 < \epsilon << 1$. The diffusion specified by (23) is recognized as a Cox-Ingersoll-Ross process (Cox et al., 1985), with unconditional mean $E[S_\tau] = 1$. Appendix A gives the details underlying this result.

Figure 2 is included to illustrate this result for a known distribution on $(y, \mathbf{x})$, $m = 1$, $y \perp \mathbf{x}$, and a simulated training data set of size $n = 1000$. The left hand side panel displays both the training loss $\hat{l}(y, f(\mathbf{x}; \hat{w}_l(u_i), \hat{w}_r(u_i)))$ (black) and the test loss $E_{y^0,x^0}\left[\hat{l}(y^0, f(\mathbf{x}^0; \hat{w}_l(u_i), \hat{w}_r(u_i)))\right]$ (blue, resolved approximately using 100000 Monte Carlo simulation from the true data generating process) as functions of $u = i/n$. Also included in the left panel is the asymptotic limit (green, dashed) which coincides for both types of loss, and is constant as the feature is uninformative w.r.t. to the response. The paths of the training- and expected test-loss are almost mirror images about the asymptotic line, and asymptotically they are indeed exactly that. This becomes clear upon inspection of (22): The only source of randomness in the expected test-loss, is the estimator based upon the (random) training data – the source of randomness for the training loss. The middle panel shows the difference in losses (left hand side of (22) scaled with conditional optimism), also as a function of $u = i/n$. Finally, the right hand side panel depicts the same curve as the middle panel, but with transformed horizontal axis conforming with the "time" τ of (23).

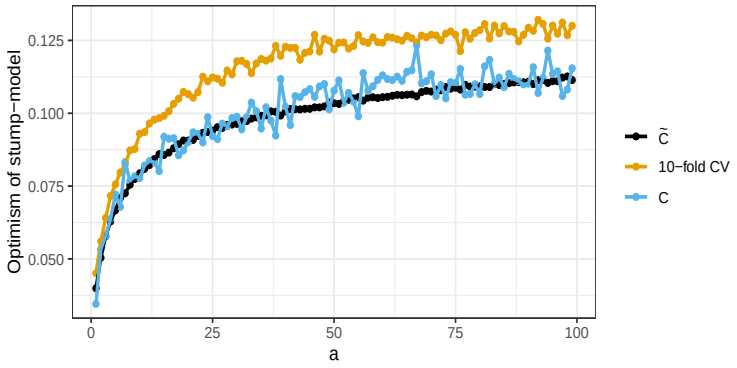

Figure 3: Illustration of Equation (24) by repeatedly simulating $n = 100$ observations in training data, for each value of a possible split-points, under the assumption $y \perp x$, $y \sim N(0,1)$, for which $\tilde{C}_{stump}$ is asymptotically exact. The feature x is constructed to have the same expected number of observations in each group, and each group is also guaranteed to have at least one observation. The simulation experiment is repeated 1000 times, and the average values are reported. The black line shows the value of $\tilde{C}_{stump}$, which aligns with the intuition that more number of split-points should correspond to increased optimism. The blue line is the Monte-Carlo estimated generalization loss C_{stump}, using the average of 1000 test-loss datasets. It verifies the above intuition and values of $\tilde{C}_{stump}$ as it fluctuates mildly about the optimism approximation. Also included is the 10-fold CV optimism (orange), that retains the shape of C and $\tilde{C}_{stump}$, but is upward biased, resulting from using 9/10'ths of the data in the estimator fitting procedure.

Now suppose x_j takes $a+1$ distinct values, then there are a different split-points s_k, $k = 1, \ldots, a$, which are compared in terms of training loss during the greedy profiling procedure. These correspond to the is such that $i/n = u_k = P(x_j \leq s_k)$ and $\tau_k = \tau(u_k)$ for the right hand side. Equation 22 provides the joint distribution of the differences in test and training loss in terms of the joint distribution of $\{\hat{C}_{\text{root}}(1 + S(\tau_k))\}_{k=1}^{a}$. Consequently, the expected maximum of $\{\hat{C}_{\text{root}}(1 + S_{\tau_k})\}_{k=1}^{a}$ is upward biased relative to $C_{\text{stump}}(j)$. In the proceeding, we will use this expected maximum, i.e.

$$\tilde{C}_{\text{stump}}(j) = \hat{C}_{\text{root}} \left(1 + E\left[\max_{1 \leq k \leq a} S(\tau_k)\right]\right) \tag{24}$$

as the (somewhat conservative in favor of the root model) approximation of $C_{\text{stump}}(j)$. As shown in in Appendix B, under assumption $y \perp x$, $\tilde{C}_{\text{stump}}(j)$ converges to $C_{\text{stump}}(j)$ as $n \to \infty$. The corresponding finite-sample behaviour, for different numbers of split-points a, is illustrated in Figure 3. In the Figure, the exact value of $C_{\text{stump}}(j)$, estimated using Monte-Carlo simulations of the true data-generating process, slightly fluctuates (due to being a simulation estimate) about the value of $\tilde{C}_{\text{stump}}(j)$. The optimism implied by 10-fold CV has the exact same shape, but is upward biased relative to $C_{\text{stump}}(j)$ and $\tilde{C}_{\text{stump}}(j)$ as it only employes 9/10'ths of the data in its fitting procedure.

The scaling factor $E\left[\max_{1 \leq k \leq a} S(\tau_k)\right]$ depends on the nature of the j-th feature. In particular for a feature taking only two values, e.g. one-hot encoding, we have $\tilde{C}_{\text{stump}}(j) = \tilde{C}_{\text{root}}(1 + E(S_{\tau_1})) = 2\tilde{C}_{\text{root}}$

as no optimization over the split point is performed, which agrees with AIC-type criteria when the number of parameters are doubled. At the other extreme, for a feature with absolutely continuous marginal distribution, the scaling factor converges to the expected maximum of a CIR process over the "time"-interval obtained by applying $\tau(u)$ to each $u \in \left(\max\left\{\frac{1}{n}, \epsilon\right\}, \min\left\{\frac{n-1}{n}, 1-\epsilon\right\}\right)$. Setting $\epsilon = 10^{-7}$ gives $E\left[\max_{0 \leq \tau \leq \frac{1}{2}\log\frac{(1-\epsilon)^2}{\epsilon^2}} S(\tau)\right] \approx 7.5$ as $n \to \infty$. In general, the expectation is bounded as long as $\epsilon > 0$, as the CIR process is positively recurrent. Linetsky (2004) give an exact analytical expression for its distribution in terms of special functions, but is not applied here as evaluation is computationally costly, generally not straight forward, and would only apply to continuous features when $a = n - 1$.

3.5 Optimism over several features

In general, the greedy binary splitting procedure profiles both over features j and within feature split-point s_j. Let $B_j = \tilde{C}_{\text{root}} \max_{1 \leq k \leq a_j}(1 + S_j(\tau_k))$ where $\{\tau_k\}$ correspond to the potential split points on the j-th feature with a_j possible split-points, so that $E_{S_\tau}(B_j) = \tilde{C}_{\text{stump}}(j)$. Following a similar logic as leading to (24), an upward biased approximation of the unconditional (over feature j) optimism C_{stump} obtains as

$$E\left[\max_{j \in \{1,\ldots,m\}} B_j\right] \tag{25}$$

However, for typical values of $m >> 1$, characterization of the dependence structure among the B_js appears difficult. Hence, in order to get a practical approximation to (25), we calculate as if the B_js are independent, to get the approximation

$$\tilde{C}_{\text{stump}} = \int_0^\infty 1 - \prod_{j=1}^{m} P\left(B_j \leq z\right) dz, \tag{26}$$

where the integral is over a single dimension and hence efficiently calculated numerically. In Section 4.2, the errors incurred by using the independence simplification on data sets with correlated features are studied.

3.6 Applications to gradient tree boosting

Returning attention to the application of the above theory in the GTB context, the ancestor node subscript t is re-introduced. All quantities, e.g. $\mathcal{R}_t$ and $\tilde{C}_{\text{stump},t}$ are calculated as if node t was the root node in the above theory, and in particular based only on the data passed to node t. The previous sections gives us the needed approximation to adjust the training loss reduction $\mathcal{R}_t$ according to the unconditional (over j)

counterpart to (17), namely

$$\tilde{\mathcal{R}}_t^0 = \mathcal{R}_t + \tilde{C}_{\text{root},t} - \tilde{C}_{\text{stump},t}. \tag{27}$$

The approximation of generalization loss reduction $\tilde{\mathcal{R}}_t^0$ has at least two important applications to the tree boosting algorithm. Firstly, it provides a natural criterion on whether to split a node or not, with the stopping criterion for splitting a leaf node t becomes

$$\tilde{\mathcal{R}}_t^0 < 0. \tag{28}$$

If no leaf node t in the tree f_k has positive $\tilde{\mathcal{R}}_t^0$, the tree building process in boosting iteration k is stopped. Note that due to the usage of an upward biased optimism approximation for the stump model, this criterion will slightly favour less complex models. In principle, (28) can be augmented to read $\tilde{\mathcal{R}}_t^0 < \rho$ where ρ is a tuning parameter controlling individual tree complexity in a coherent manner. However, this option is not pursued further as the default $\rho = 0$ produces good results in practice.

Further, the proposed approximate optimism may also be applied within a stopping-rule for the gradient boosting iteration – often referred to as "early stopping". When a tree-stump, scaled by the learning rate δ, no longer gives a positive reduction in approximate generalization loss relative to the previous boosting iterate, we terminate the algorithm. Care must be taken as the learning rate δ scales the training loss and the optimism differently. Recalculating the training loss (9), with δf_k as the predictive function, we obtain that the training loss associated with f_k should be scaled with a factor $\delta\,(2-\delta)$. The optimism, on the other hand scales linearly, as is seen from expressing optimism as a covariance, $C = \frac{2}{n}\sum_{i=1}^{n} Cov(y_i, \hat{y}_i)$, (Hastie et al., 2001, p. 229) and recalling that $\hat{y}_i$ is linear in δ. The boosting stopping criterion hence becomes (with ancestor index $t = 1$):

$$\tilde{\mathcal{R}}_\delta^0 = \delta(2-\delta)\mathcal{R}_1 + \delta\left(C_{\text{root},1} - \tilde{C}_{\text{stump},1}\right) > 0. \tag{29}$$

When (29) evaluates to true, there is no more information left in data for another member added to the ensemble $f^{(k-1)}$ to learn, in the generalization error sense, using the boosting iteration of Algorithm 1.

Algorithm 1 with orange markers (and not blue markers) gives the proposed modified algorithm. The early stopping criterion saves one hyperparameter. The adaptive tree complexity on the other hand alleviate the need for the multiple hyperparameters typically used to fine-tune the tree complexities. E.g. the popular `xgboost` implementation has 4 such hyperparameters: a constant minimum reduction in loss, a maximum depth, a minimum child weight and a maximum number of leaves. These computational-reductions stemming

from not having to tune the original algorithm are explored and measured in more detail in Section 5.3.

3.7 Implementation

Recall that the basic building block of the above theory is the root optimism approximation (19). However, this approximation also depends on moments (Expected loss Hessian and parameter variance) which must be estimated empirically in the numerical implementation. As previously mentioned, (19) is a special case of theoretical optimism of Murata et al. (1994). Further, Murata et al. (1994) estimated the parameter variance using the conventional Sandwich Estimator (see e.g. van der Vaart, 1998, Section 5.3), as the estimated leaf weights (8) are M-estimators. This approach is also taken here, and results in the root optimism estimator:

$$\hat{C}_{\text{root},t} \approx \frac{\sum_{i \in I_t} (g_i + h_i \hat{w}_t)^2}{n_t \sum_{i \in I_t} h_i} \tag{30}$$

where $n_t = |I_t|$ is the number of observations passed to leaf t.

The same estimator is also used for evaluating conditional stump optimisms $\hat{C}_{\text{stump},t}(j|\hat{s}_j)$ in (20), but of course then based on the on the subsets of data falling into the left and right child nodes of t. The probabilities in (20) are simply estimated as the corresponding relative frequencies in the training data.

When (19) is evaluated using (30), adding evaluation of $\hat{\mathcal{R}}^0$ to the greedy-binary-splitting procedure does not change the computational complexity of the overall algorithm, as the only cost is to keep track of sum of squares and cross multiplication among the g and h vectors.

The expected maximums over CIR processes (26) are resolved based on a combination of Monte Carlo simulations and approximating the B_j-s by a parametric distribution. First of all, (25) is approximated by assuming independence, obtaining (26). We then *only* need knowledge of the CDF of the maximum of the CIR process observed on time-points associated with the split-points of feature j. Linetsky (2004) gives expressions for the maximum of the CIR on an interval, however, the expressions are not easily calculated and comes to a non-negligible computational cost, and would also penalize non-continuous features too much. We therefore consider an alternative approach: For the case with only one possible split, the Gamma distribution with shape 0.5 and scale 2 is used, which is exact. For the cases with more than one split a Monte Carlo simulation procedure is used to simulate the expected maximum of the CIR over the split-points on feature j. In principle we could simulate indefinitely to obtain exact estimates of the CDF. However, this quickly becomes infeasible when the number of features grows large, and (26) will be concerned with the tail-behaviour of the CIR maximums. We therefore do an asymptotic approximation, by fitting the CIR to the Gumbel distribution, which it is in the maximum domain of attraction of, as it has a Gamma stationary distribution. The approximation is asymptotic in the number of observations-points, and will be expected

DGP	$y \sim N(0,1)$		$y \sim N(0,5^2)$		$y \sim N(\lfloor x \rceil, 1)$		$y \sim N(\lfloor x \rceil, 5^2)$		$y \sim N(x,1)$		$y \sim N(x,5^2)$	
	E	P	E	P	E	P	E	P	E	P	E	P
						$a+1=2$						
$\mathcal{R}$	0.969	1	24.1	1	25.6	1	48.4	1	26	1	47.8	1
$\mathcal{R}^0$	-0.966	0.016	-24.1	0.024	24	1	.212	0.684	23.9	1	.47	0.691
$\tilde{\mathcal{R}}^0$	-1.03	0.154	-25.5	0.157	22.7	0.998	-2.82	0.332	22.9	1	-2.65	0.35
10-fold CV	-1.16	0.165	-29.9	0.162	24	0.998	-4.93	0.342	24.3	1	-3.68	0.365
100-fold CV	-1.06	0.159	-26.3	0.157	24.1	0.999	-2.17	0.335	24.4	1	-2.03	0.352
						$a+1=10$						
$\mathcal{R}$	2.99	1	72.4	1	26.6	1	95	1	12.3	1	84.9	1
$\mathcal{R}^0$	-2.99	0	-72.5	0	22.5	0.996	-52.5	0.185	4.86	0.947	-61.9	0.046
$\tilde{\mathcal{R}}^0$	-2.82	0.086	-75	0.084	17.8	0.992	-53.3	0.164	4.4	0.763	-62.3	0.136
10-fold CV	-3.46	0.202	-90.3	0.199	22.7	0.982	-65.7	0.266	4.54	0.682	-73.7	0.243
100-fold CV	-3.18	0.316	-81.2	0.286	23.1	0.98	-60.3	0.37	4.77	0.727	-60.6	0.364
						$a+1=100$						
$\mathcal{R}$	4.58	1	115	1	28	1	136	1	12.9	1	124	1
$\mathcal{R}^0$	-4.58	0	-115	0	20.7	0.995	-96.9	0.052	2.46	0.799	-106	0
$\tilde{\mathcal{R}}^0$	-4.73	0.048	-116	0.057	14	0.957	-103	0.092	.582	0.489	-112	0.061
10-fold CV	-5.41	0.158	-141	0.157	20.3	0.975	-119	0.21	2.14	0.586	-142	0.183
100-fold CV	-5.31	0.226	-130	0.233	20.7	0.965	-111	0.301	2.53	0.672	-127	0.258

Table 1: Single feature root versus stump loss reduction simulation study with $n = 100$ observations. The contending methods are cross validation (CV) and the proposed test loss reduction estimator $\tilde{\mathcal{R}}^0$. In addition, the test loss $\mathcal{R}^0$ and training loss $\mathcal{R}$ were included for reference. Columns E give the expected loss reduction, multiplied by a factor 100 for readability, for the different estimators, and columns P give the probability of a positive loss reduction (i.e. probability of choosing the stump model). In all cases, the feature was simulated on $(0, 1)$ and with $a + 1$ distinct values. The results are based on 1000 simulated data sets in each case. The test loss $\mathcal{R}^0$, was estimated using 1000 simulated test responses for each simulated data set.

to perform increasingly well in the number of split points.

4 Simulation experiments

The theory developed in the previous section involves multiple approximations. This section studies the performance of the proposed training loss reduction estimator when the data generating process is known a-priori. All computations involving the proposed methodology were done using the associated R-package `aGTBoost` which can be downloaded from https://github.com/Blunde1/aGTBoost, and scripts that re-create the below results can be found at the same place. `aGTBoost` is written mainly in C++, and computing times are therefore directly comparable to those of e.g. `xgboost`.

4.1 Simulations in the single feature case

In the first batch of simulation experiments, the single feature estimator of the test loss reduction in the root versus stump situation, developed in Section 3.4 is considered. The results are summarized in Tables 1 and 2 for $n = 100$ and $n = 1000$ respectively. In the experiments, the test loss reduction estimator $\tilde{\mathcal{R}}^0$

DGP	$y \sim N(0,1)$		$y \sim N(0,5^2)$		$y \sim N(\lfloor x \rceil, 1)$		$y \sim N(\lfloor x \rceil, 5^2)$		$y \sim N(x,1)$		$y \sim N(x,5^2)$	
	E	P	E	P	E	P	E	P	E	P	E	P
$a+1=2$												
$\mathcal{R}$	0.904	1	23.3	1	251	1	280	1	253	1	277	1
$\mathcal{R}^0$	-0.903	0.023	-23.3	0.022	249	1	226	1	249	1	222	0.998
$\tilde{\mathcal{R}}^0$	-1.09	0.138	-26.7	0.15	248	1	229	0.974	250	1	226	0.956
10-fold CV	-1.22	0.157	-29.7	0.166	250	1	228	0.962	252	1	225	0.947
100-fold CV	-1.1	0.134	-27.3	0.146	250	1	231	0.971	252	1	227	0.957
$a+1=10$												
$\mathcal{R}$	2.89	1	75.3	1	251	1	301	1	84	1	174	1
$\mathcal{R}^0$	-2.9	0	-75.3	0	249	1	162	0.935	71.9	1	14.3	0.709
$\tilde{\mathcal{R}}^0$	-2.94	0.087	-70.6	0.104	242	1	152	0.828	76.1	1	25.8	0.52
10-fold CV	-3.4	0.204	-81.4	0.226	250	1	166	0.784	71.7	1	5.04	0.471
100-fold CV	-2.97	0.379	-75.1	0.383	250	1	169	0.797	71.4	0.992	15.4	0.608
$a+1=100$												
$\mathcal{R}$	4.58	1	114	1	252	1	328	1	74.6	1	207	1
$\mathcal{R}^0$	-4.57	0	-114	0	248	1	137	0.872	58.8	1	-39.9	0.382
$\tilde{\mathcal{R}}^0$	-4.73	0.051	-119	0.041	238	1	90.9	0.694	62.1	1	-28.7	0.322
10-fold CV	-5.52	0.176	-139	0.176	248	1	107	0.675	57.7	0.999	-43.5	0.362
100-fold CV	-5.08	0.325	-129	0.33	249	1	120	0.726	58.9	0.982	-28.4	0.542
$a+1=1000$												
$\mathcal{R}$	5.7	1	143	1	254	1	365	1	75.6	1	220	1
$\mathcal{R}^0$	-5.71	0	-143	0	246	1	117	0.833	57.4	1	-64.6	0.284
$\tilde{\mathcal{R}}^0$	-5.78	0.03	-144	0.033	236	1	71.9	0.626	60.3	1	-71.3	0.208
10-fold CV	-6.57	0.16	-162	0.149	246	1	109	0.668	57.2	1	-82.9	0.316
100-fold CV	-6.1	0.288	-154	0.298	247	1	131	0.732	57.7	0.985	-78.3	0.455

Table 2: Single feature root versus stump loss reduction simulation study with $n = 1000$ observations. The contending methods are cross validation (CV) and the proposed test loss reduction estimator $\tilde{\mathcal{R}}^0$. In addition, the test loss $\mathcal{R}^0$ and training loss $\mathcal{R}$ were included for reference. Columns E give the expected loss reduction, multiplied by a factor 1000 for readability, for the different estimators, and columns P give the probability of a positive loss reduction (i.e. probability of choosing the stump model). In all cases, the feature was simulated on $(0, 1)$ and with $a+1$ distinct values. The results are based on 1000 simulated data sets in each case. The test loss $\mathcal{R}^0$, was estimated using 1000 simulated test responses for each simulated data set.

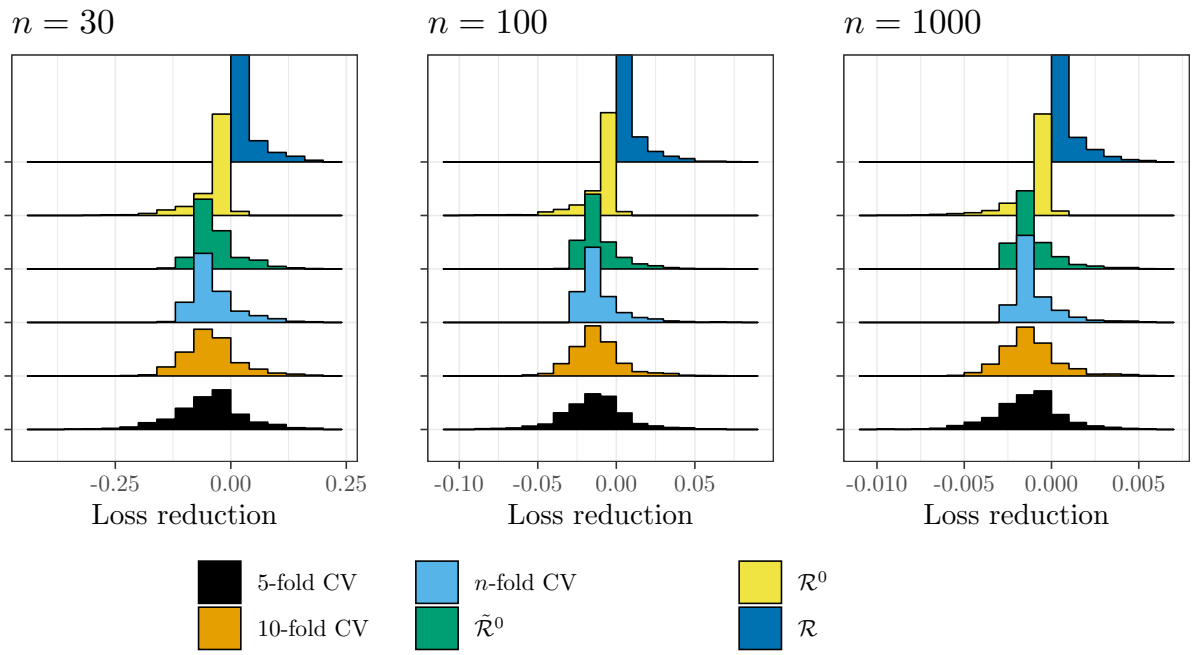

Figure 4: Histograms of single feature root versus stump loss reduction, with $n = \{30, 100, 1000\}$, $a = 1$ and the DGP being $y \sim N(0, 1)$. The results are based on 1000 simulation replica in each case. The two values taken by the feature were simulated uniformly on $(0, 1)$.

is compared to two fidelities of cross validation, and in addition test loss and training loss are provided as references. For both sample sizes, a range of numbers of potential split points a are considered, including binary feature ($a = 1$) and continuous feature ($a + 1 = n$). In the tables, "E" corresponds to the mean the loss reductions, and P is the probability of rejecting the root model.

Six data generating process (DGP) cases were considered. For the former two DGPs ($y \sim N(0, \sigma^2)$ with $\sigma = 1, 5$) the feature is un-informative with respect to y. The rejection rate of the (true) root model for non-binary features is around 5-10 % for $n = 100$ observations and around 5 % for $n = 1000$. It is seen from Tables 1, 2 the proposed methodology does on par (the $a = 1$ case) or better (the $a > 1$ cases) than cross validation. For binary features ($a = 1$), the expectation of $\tilde{\mathcal{R}}^0$ is very close to that of $\mathcal{R}^0$, but the root model rejection rate is higher. To better understand this phenomenon, the $a = 1$, $y \sim N(0, 1)$ case is further explored in Figure 4. As expected in the $a = 1$ case, the training losses and test losses are close to being mirror images around the asymptotic loss reduction, which in this independent response case of course is 0. This effect is a consequence of the training- and test loss empirical processes (see left panel of Figure 2) themselves are close to being symmetric around zero loss reduction. Specifically, in the $a = 1$ case, these losses obtains as evaluations of the empirical processes at single point on the horizontal axis, which gives rise to the symmetry. It is also seen that the shapes of the right hand side tails of $\mathcal{R}$ and $\tilde{\mathcal{R}}^0$ are very similar,

but with $\tilde{\mathcal{R}}^0$ shifted to have expectation close to that of $\mathcal{R}^0$ (see Tables 1 and 2). In this case, the heavy right hand side tail of $\tilde{\mathcal{R}}^0$ leads to non-negligible rate of rejection of the (appropriate) root model, even if the mean is essentially that of $\mathcal{R}^0$.

In the next two DGPs ($y \sim N(\lfloor x \rceil, \sigma^2)$ with $\sigma = 1, 5$), the stump model with split point $s = 0.5$ is the true model. In the high signal-to-noise ratio case $\sigma = 1$, both the proposed estimator $\tilde{\mathcal{R}}^0$ and cross validation select the true model almost perfectly in both sample sizes. Interestingly, in the $\sigma = 5$ case, the test loss reduction $\mathcal{R}^0$ selects the root model rather often, and the proposed test loss reduction estimator and cross validation largely follow this behavior.

Finally, the last two DGPs ($y \sim N(x, \sigma^2)$ with $\sigma = 1, 5$), the feature is also informative with respect to the response, but the dependence is linear rather discontinuous. It is seen also in this case that the proposed test loss reduction estimator and cross validation estimators behaves similarly to the test loss with respect to rejection probabilities.

Recall that stump optimism estimator (24) was derived based on an independence assumption between the feature and response. The simulation studies do not provide evidence that optimism estimators calculated from informative features somehow overwhelms the reduction in training loss. Further, notice in the $a = 1$ case where no optimization over split-points is performed, it is still not expected that $\tilde{\mathcal{R}}^0$ is exactly equal to $\mathcal{R}^0$. This is as there is approximation error in (18), and that involved moments are estimated from data in $\tilde{\mathcal{R}}^0$.

The initial conclusion to be drawn from from these simulations is that the proposed estimator has small sample performance at least on par with cross validation in the root vs stump situation with one feature and mean squared error losses, but at a much lower computational cost.

4.2 Simulations of the multiple feature case

This subsection explores the performance of the proposed methodology in the presense of more than one feature. Recall from Section 3.5 that $\tilde{\mathcal{R}}^0$ in this case is derived under the assumption that the (multiple) features are mutually independent and also independent of the response. Figure 5 depicts average $\tilde{\mathcal{R}}^0$, along with simulated test loss reduction $\mathcal{R}^0$ for different numbers of uninformative and independent features and standard normal responses. Also included in the Figure is the corresponding training loss reduction, $\mathcal{R}$.

It is seen that the $\tilde{\mathcal{R}}^0$ and $\mathcal{R}^0$ have very similar behavior. However, small deviations still exist, stemming both from the deliberate (downward) bias introduced in equations 24, 25 for $a > 1$, and also the simulation based algorithm used to estimate the expected maxima of (25). Still, it does not seem that these approximations introduces an undue amount of bias towards the root model in this particular setting.

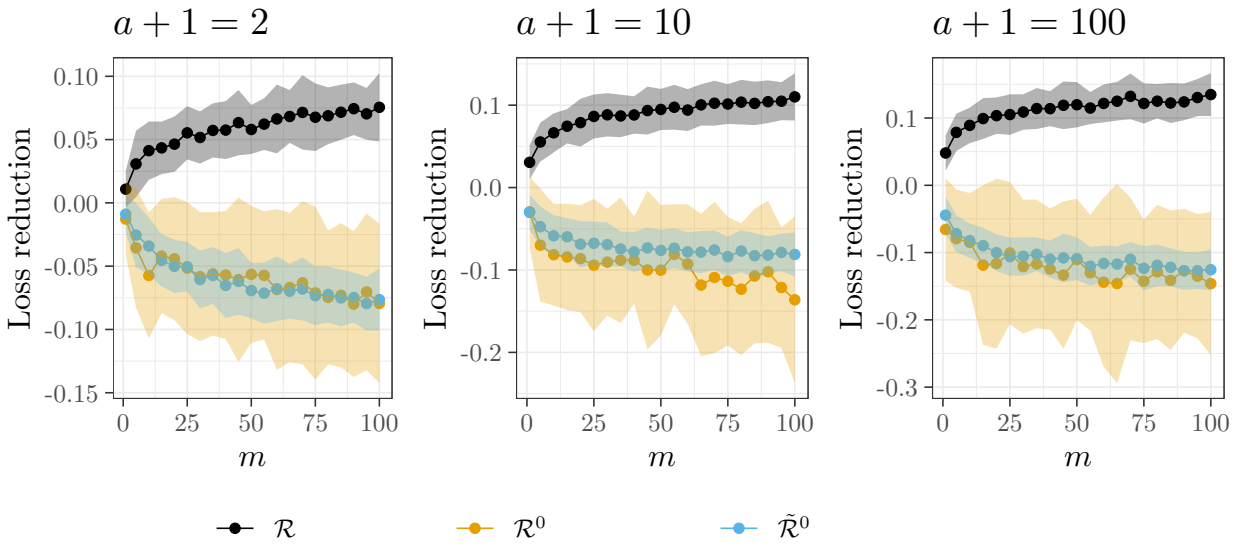

Figure 5: Root versus stump loss reduction as a function of the number of features m, when all features are uninformative. The DGP is $y_i \sim N(0, 1)$, and the m features are simulated conditional on the given number a of potential split points. Dots give the average loss reductions over 100 simulated data sets of size $n = 100$ for each considered m, and shaded areas are the averages $\pm$ one standard deviation. The test loss $\mathcal{R}^0$ was obtained from a single simulated test set for each simulation replica.

Method	Case 1 ($m = 1$)			Case 2 ($m = 10000$)			Case 3 ($m = 10000$)		
	Loss	K	CPU-Time	Loss	K	CPU-Time	Loss	K	CPU-Time
linear model	0.977		0.0293	1.01		16	1.07		43
`aGTBoost`	1.01	365	0.162	1.05	294	723	1.04	348	821
`xgboost`: cv	1.11	275	4.28	1.07	357	3447	1.08	370	3908
`xgboost`: val	1.16	311	0.507	1.16	371	258	1.09	249	171

Table 3: Test losses, (where relevant) number of trees K and associated computing times for the linear model (31) with the different cases corresponding to different design matrices described in Section 4.2. Single core CPU-times are measured in seconds. Test losses are evaluated on a test data set of size $n = 1000$. As a reference, a constant model corresponds to a test loss of ≈ 2.34.

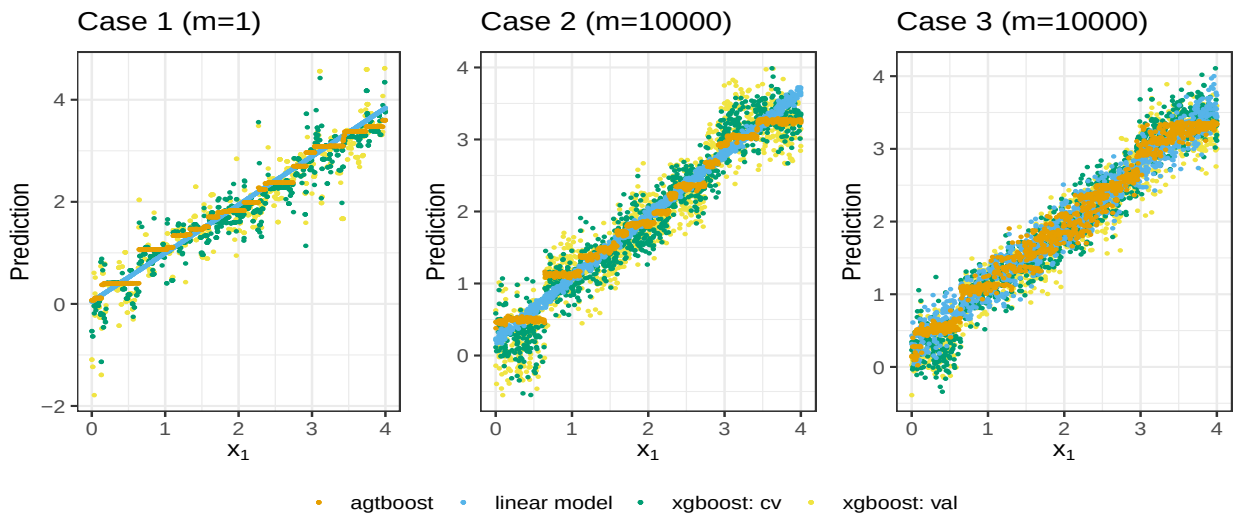

Figure 6: Predictions for the linear model (31) with the different cases corresponding to different design matrices described in Section 4.2. The predictions are evaluated on the test data set features and plotted as function of x_1.

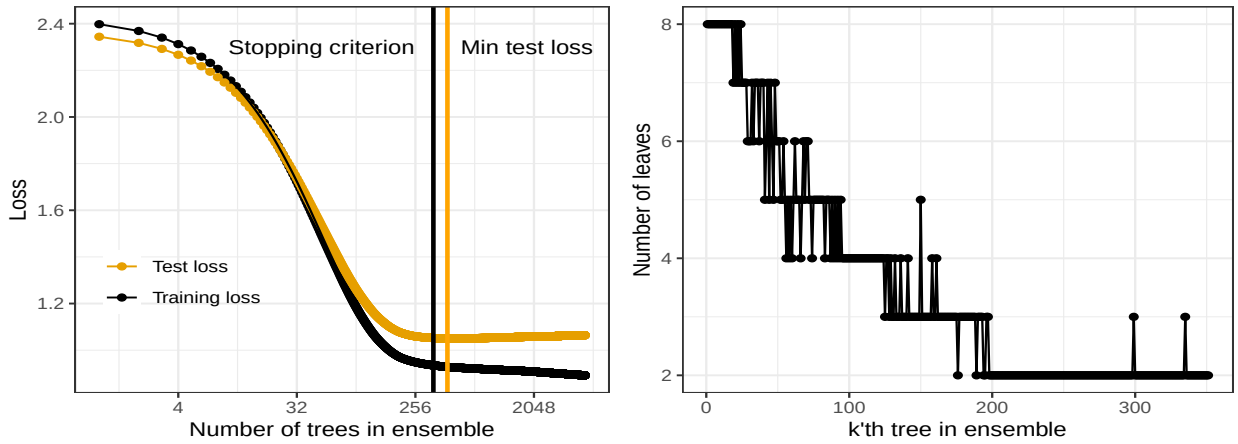

Figure 7: (left) Training loss (black) and test loss (orange) plotted versus the number of boosting iterations or the number of trees in the ensemble (note the $\log_{10}$ axis). Included is a vertical line (orange) representing the iteration number that the stopping criterion (29) terminates the procedure and another (black) representing the minimum test-loss. (right) The number of leaves for each tree in the ensemble until termination by the stopping criterion. The data is simulated with a linear relationship between the response and the first feature, $y_i \sim N(x_{i,1}, 1)$ and additional 9 noisy Gaussian features. The informative feature $x_{\cdot,i}$ is sampled uniformly on $[0, 4]$. Both training and test loss consists of $n = 1000$ examples, the ensemble had a learning rate of 0.01.

As more realistic, but still simulated situation, we considered $n = 1000$ training data observations with data generating process being

$$y_i \sim N(x_{i,1}, 1),\ i = 1, \ldots, n,\ x_{i,1} \sim \text{ iid } U(0, 4). \tag{31}$$

This situation tests the recursive usage of the proposed methodology in a full application of gradient tree boosting, including tree building- and boosting iteration termination criteria. Three cases, with appropriate linear model benchmarks, where considered:

Case 1: $m = 1$, where $x_{\cdot,1}$ is the only feature. The benchmark linear model was an un-regularized linear regression model.

Case 2: $m = 10000$ with $x_{i,k},\ i = 1, \ldots, n,\ k = 2, \ldots, m$ being iid $U(0, 4)$ noise independent of $x_{\cdot,1}$. The benchmark linear model was the Lasso regression with regularization determined by 10-fold cross validation, implemented in the `glmnet` R-package.

Case 3: $m = 10000$ with dependent features $x_{i,k} = \frac{m-k}{m} x_{i,k-1} + N(0, (k/m)^2),\ i = 1, \ldots, n,\ k = 2, \ldots, m$. The benchmark linear model was the Ridge regression with regularization determined by 10-fold cross validation, implemented in the `glmnet` R-package.

As additional benchmarks, gradient boosted tree ensembles were obtained using `xgboost`. Default settings were used, and number of boosting iterations were learned using cross validation (`xgboost`:CV) and a 30% validation set (`xgboost`:VAL).

The linear model (31) constitutes a substantial model selection challenge for tree-based predictors, as a rather complex tree ensembles are required to faithfully represent the linear functional form. Table 3 provides test losses for the proposed methodology and the benchmarks obtained from test data sets with 1000 observations.

From the Table, it is seen that `aGTBoost` provides better test losses than the `xgboost`-based benchmarks, and also better test loss than for Ridge regression in Case 3. Further, in all cases, the test loss obtained by `aGTBoost` is quite close to the benchmark linear models, indicating a close to optimal behavior given that the linear functional form cannot be represented exactly by finite tree ensembles. Further, `aGTBoost` produces marginally better test losses than `xgboost`:CV, whereas `xgboost`:Val is not competitive. The computing time associated with `aGTBoost` is about an order of magnitude smaller than that of `xgboost`:CV.

Figure 6 gives a graphical illustration of the predictions made by the contending methods. It is seen that `aGTBoost` produces substantially more parsimonious fits than both `xgboost` methods. In particular in Case 2, the `aGTBoost` boosting iterations stop criterion is meet before the algorithm starts utilizing the noise

features $x_{\cdot,k},\ k = 2, \ldots, m$. This is in contrast to the Lasso regression, which as can be seen from the noisy predictions in the plot, assigns non-zero predictive power to some of the noise features. In Case 3, some of the dependent noise features $x_{\cdot,k},\ k = 2, \ldots, m$ are used by `aGTBoost`, but the fit is still substantially less variable than for the contenting tree boosting methods.

The left panel of Figure 7 depicts the test- and training losses of `aGTBoost` as function of the boosting iterations in Case 1. Also indicated with an orange vertical line is the boosting iteration where stop criterion (29) becomes negative. More precisely, the `aGTBoost` results reported in Table 3 and Figure 6 are based on the boosting iterate immediately before the vertical line (but more iterations were carried out for the purpose of Figure 7). It is seen that the stop criterion becomes active very close to the global minimum of the training loss (also indicated by black vertical line in the Figure).

From the right panel of Figure 7, it is seen that `aGTBoost` builds deep trees (relative to stumps) at early iterations. As information is learned by the ensemble, subsequent trees become smaller until they are stumps, and the algorithm terminates shortly thereafter.

To summarize; the application of the proposed methodology in actual gradient tree boosting results in highly competitive tree ensemble fits in the example model cases 1-3. This appears to be a consequence of both the adaptive selection of the number of leaf nodes in each individual tree, and also that such adaptive features enable the (automatic) selection of quite few (and hence computationally cheap) boosting iterations.

5 Comparisons on benchmark datasets

To further illustrate the validity of the modified boosting algorithm implemented in aGTBoost, we test it on all regression and classification datasets in Hastie et al. (2001) and James et al. (2013). These datasets represent a relatively broad spectrum of model-types (Table 4).

5.1 Algorithms

Our algorithm is compared against the `xgboost` implementation. Our hypothesis is that the two algorithms will give similar predictions, but will differ in computation time and ease of use. To ensure comparability, we avoid L1 and L2 regularization of the loss and stochastic sampling in `xgboost`. In addition, we include random forest and generalized linear models in the comparisons. Lastly, we include a version of our proposed algorithm restricted to a single ($K = 1$) unscaled ($\delta = 1$) tree, and a CART tree learned with CV and cost complexity pruning. This gives additional validation of the root-stump criterion (28).

Dataset	$n \times m$	Loss function	train vs test	Source packages
Boston	506×14	MSE	$50 - 50$	MASS
Ozone	111×4	MSE	$50 - 50$	ElemStatLearn
Auto	392×311	MSE	$70 - 30$	ISLR
Carseats	400×12	MSE	$70 - 30$	ISLR
College	777×18	MSE	$70 - 30$	ISLR
Hitters	263×20	MSE	$70 - 30$	ISLR
Wage	3000×26	MSE	$70 - 30$	ISLR
Caravan	5822×86	Logloss	$70 - 30$	ISLR
Default	10000×4	Logloss	$70 - 30$	ISLR
OJ	1070×18	Logloss	$70 - 30$	ISLR
Smarket	1250×7	Logloss	$70 - 30$	ISLR
Weekly	1089×7	Logloss	$70 - 30$	ISLR

Table 4: All regression and classification datasets from the books Hastie et al. (2001); James et al. (2013), their dimensions, loss functions (MSE corresponds to regression, Logloss to classification), the percentage split to training and test, and source. Dimensions are after using the R function `model.matrix()`, which performs one-hot encoding on the data, and remove NA values. See Table 1.1 in James et al. (2013) for further descriptions of the datasets.

5.2 Computation

Computations are done in `R` version 3.6.1 on a Dell XPS-15 computer running 64-bit Windows 10, utilizing only a single core for comparability of algorithms. We run `xgboost` 0.90.0.2, `randomForest` 4.6-14 and `tree` 1.0-40 which contain the CART algorithm. GLM algorithms are found in the base-`R` `stats` library, through the functions `lm()` for linear regression, and `glm()` with specified `family=binomial` for logistic regression. For `randomForest` we use the default parameter values. The same is the case for `lm` and `glm`, while `tree` is trained using pruning on a potentially deep tree.

For the results in Table 5, `xgboost` is trained with a learning rate of $\delta = 0.1$, the same as aGTBoost, and importantly, L2 regularization are removed from the boosting objective by setting the (by-default non-zero) `lambda` parameter to zero. The number of trees, K, for `xgboost` models are found by 10-fold CV, where we check if the 10 consecutive trees improve overall CV-loss, selected by setting `early_stopping_rounds`=10. The configuration of `xgboost` in Table 6 is identical to Table 5, except for the learning rate set to $\delta = 0.01$ (same as for aGTBoost). The different variants of `xgboost` in Table 6 differ in the CV profiling over the hyperparameters `max_depth` and `gamma`. Also, a variant using 30% of the training data as a validation set for selecting K is included.

Each dataset is split randomly into a training set and a test set (see Table 4). All algorithms train on the same training set, and report the loss over the test set. This is done for 100 different splits, and the mean and standard deviation of relative test loss (to `xgboost`) is calculated across these 100 datasets.

Dataset	xgboost	aGTBoost	random forest	glm	CART	gbtree
Boston	1 (0.173)	1.02 (0.144)	0.877 (0.15)	1.3 (0.179)	1.55 (0.179)	1.64 (0.215)
Ozone	1 (0.202)	0.816 (0.2)	0.675 (0.183)	0.672 (0.132)	0.945 (0.225)	1.13 (0.216)
Auto	1 (0.188)	0.99 (0.119)	0.895 (0.134)	11.1 (14.6)	1.45 (0.185)	1.45 (0.201)
Carseats	1 (0.112)	0.956 (0.126)	1.16 (0.141)	0.414 (0.0433)	1.84 (0.212)	1.9 (0.195)
College	1 (0.818)	1.27 (0.917)	1.07 (0.909)	0.552 (0.155)	1.46 (0.881)	1.71 (1.08)
Hitters	1 (0.323)	0.977 (0.366)	0.798 (0.311)	1.21 (0.348)	1.23 (0.338)	1.21 (0.408)
Wage	1 (1.01)	1.39 (1.64)	82.5 (21.4)	290 (35.5)	109 (6.78)	2.41 (1.91)
Caravan	1 (0.052)	0.983 (0.0491)	1.3 (0.167)	1.12 (0.115)		
Default	1 (0.0803)	0.926 (0.0675)	2.82 (0.508)	0.898 (0.0696)		
OJ	1 (0.0705)	0.966 (0.0541)	1.17 (0.183)	0.949 (0.0719)		
Smarket	1 (0.00401)	0.997 (0.00311)	1.04 (0.0163)	1 (0.0065)		
Weekly	1 (0.00759)	0.992 (0.00829)	1.02 (0.0195)	0.995 (0.0123)		

Table 5: Average relative test-loss and standard deviations (parentheses) across 100 random splits of the full datasets into training and test, for the datasets in 4. The reported values are relative to the average `xgboost` test-loss values. `aGTBoost` is the modified boosting algorithm 1, gbtree is a regression tree stopping according to (28), CART is from the R package "tree", GLM uses a linear regression model for MSE-loss and logistic regression for classification. Random forest uses the default settings in the "randomForest" R package, while `xgboost` is trained deterministically with CV on the number of trees with maximum depth 6 but no L1 or L2 regularisation. The learning rate, δ, is set to 0.1 for both `aGTBoost` and `xgboost`.

	aGTBoost	xgboost				
Variant		30% Validation	K	K, gamma	K, max depth	K, gamma, max depth
Runtime (seconds)	1.46	1.3	8.55	190	90.6	2033
Test loss	0.3792	0.4229	0.3985	0.3839	0.3743	0.3983

Table 6: CPU computations time in seconds for the training of `aGTBoost` versus different variants (Section 5.2) of `xgboost` for the "OJ" dataset. `gamma` takes values on integers from 0-9, and max depth takes values on integers 1-10. Also reported is the loss on 30% test data. The naive test loss (constant prediction) is 0.662.

5.3 Results

Consider first the two rightmost columns in Table 5, reporting the results from the CART and gbtree single-tree models. These constitute the building blocks of `xgboost` and `aGTBoost`, respectively, and might therefore indicate an explanation for potential differences in the results of `xgboost` and `aGTBoost`. Overall, the results are fairly similar with a slight advantage for CART, but well within the standard deviations of Table 5, except for the Wage data. The fundamental difference of the CART trees and gbtree lies in the tree-building method of CART which performs consecutive splitting, also after encountering the first split giving a negative reduction in loss, until a pre-defined depth is reached and then a pruning process is initiated. The gbtree method, on the other hand, and by extension aGTBoost, stops splitting immediately when encountering the first split giving a negative loss reduction in approximate generalization loss. Most of the results favour slightly the cost-complexity pruning done by CART. However, the wage data strongly favour gbtree, showing that the adaptiveness of gbtree has other advantages than just speed and ease-of-use. The CART trees are constrained by their default setting for tree-depth, which is likely to cause the inferior performance for this dataset. The adaptive gbtrees, on the other hand, are able to build rather deep trees. Overall, the results are so similar that we would be hard pressed to attribute potential large differences in `xgboost` and `aGTBoost` to their individual tree building algorithms.

We then turn to the comparison of `xgboost` and `aGTBoost` in Table 5. aGTBoost outperforms `xgboost` on 9 out of 12 datasets, although the average test losses are within the Monte-Carlo (permutation) uncertainty of each other. The results for the other methods, random forest and GLM, gives an additional perspective on difference between `xgboost` and `aGTBoost`. For some datasets the GLM and random forest have slightly lower test-loss, but for other significantly higher test-loss.

Having demonstrated similar performance as regularized un-penalized `xgboost`, the vantage point of `aGTBoost` is its automatic properties and as a consequence, speed. Table 6 tells a story of computational benefits to this adaptivity: What took 1.46 seconds for `aGTBoost` took a regularized `xgboost` (K, `gamma`, `max_depth` variant) 2033 seconds. Furthermore, this adaptivity does not only have computational benefits, but also decreases the threshold for users that are new to tree-boosting: By eliminating the need to set up a search grid for the `gamma` and the `max_depth` hyperparameters in `xgboost`, `aGTBoost` lowers the bar to employ gradient tree boosting as an off-the-shelf method for practitioners. Notice also that of all the different variants of `xgboost`, only one (tuning K and maximum depth), slightly outperformed `aGTBoost` in terms of test-loss. A final observation is that simultaneously tuning K, `gamma` and `max_depth`, gives higher test-loss than only tuning K and `max_depth` in `xgboost`. This is likely due to the high variation inherent in CV.

6 Discussion

This paper proposes an information criterion for the individual node splits in gradient boosted trees, which allows for a modified and more automatic gradient tree boosting procedure as described in Algorithm 1. The proposed method (`aGTBoost`), and its underlying assumptions, were tested on both simulated and real data, and were seen to perform well under all testing regimes. In particular, the modifications allow for significant improvements in computational speed for all variants of `xgboost` involving hyperparameters. Additionally, `aGTBoost` lowers the bar for employing GTB as an off-the-shelf algorithm, as there is no need to specify a search grid and set up k-fold CV for hyperparameters.

One potential problem with `aGTBoost` is the tendency of early trees being too deep in complex datasets, as illustrated in Figure 7. This is because `aGTBoost` does not have a global hyperparameter for the maximum complexity of trees (`max_depth` as in `xgboost`, or a maximum number of leaves hyperparameter). The problem of too deep trees in GTB was first noted in Friedman et al. (2000), who suggested to put a bound on the number of terminal nodes for all trees in the ensemble.

The leading implementations of GTB come with options to modify the algorithm with stochastic sampling and L1 and L2 regularization of the loss, modifications that often improve generalization scores. This differ from the deterministic un-penalized GTB flavour discussed in this paper, and which the theory behind the information criterion assumes. Further work will try to accommodate these features, and allow for automatic tuning of sampling-rates and severity of loss-penalization.

References

Akaike, H. (1974). A new look at the statistical model identification. *IEEE transactions on automatic control 19*(6), 716–723.

Breiman, L. (2001). Random forests. *Machine learning 45*(1), 5–32.

Burnham, K. P. and D. R. Anderson (2003). *Model selection and multimodel inference: a practical information-theoretic approach*. Springer Science & Business Media.

Chen, T. and C. Guestrin (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794. ACM.

Chen, T., T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, R. Mitchell, I. Cano, T. Zhou, M. Li, J. Xie, M. Lin, Y. Geng, and Y. Li (2018). *xgboost: Extreme Gradient Boosting*. R package version 0.71.2.

Cox, J. C., J. E. Ingersoll, and S. A. Ross (1985). A theory of the term structure of interest rates. *Econometrica 53*(2), 385–407.

Dorogush, A. V., V. Ershov, and A. Gulin (2018). Catboost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*.

Friedman, J., T. Hastie, R. Tibshirani, et al. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics 28*(2), 337–407.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189–1232.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis 38*(4), 367–378.

Gombay, E. and L. Horvath (1990). Asymptotic distributions of maximum likelihood tests for change in the mean. *Biometrika 77*(2), 411–414.

Hastie, T., R. Tibshirani, and J. Friedman (2001). *The elements of statistical learning*. Springer series in statistics New York, NY, USA:.

Ho, T. K. (1995). Random decision forests. In *Document analysis and recognition, 1995., proceedings of the third international conference on*, Volume 1, pp. 278–282. IEEE.

Huber, P. J. et al. (1967). The behavior of maximum likelihood estimates under nonstandard conditions. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Volume 1, pp. 221–233. University of California Press.

James, G., D. Witten, T. Hastie, and R. Tibshirani (2013). *An introduction to statistical learning*, Volume 112. Springer.

Ke, G., Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu (2017). Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pp. 3146–3154.

Linetsky, V. (2004). Computing hitting time densities for CIR and OU diffusions: Applications to mean-reverting models. *Journal of Computational Finance 7*.

Mason, L., J. Baxter, P. L. Bartlett, and M. R. Frean (2000). Boosting algorithms as gradient descent. In *Advances in neural information processing systems*, pp. 512–518.

McCullagh, P. and J. A. Nelder (1989). *Generalized linear models*, Volume 37. CRC press.

Miller, R. and D. Siegmund (1982). Maximally selected chi square statistics. *Biometrics*, 1011–1016.

Murata, N., S. Yoshizawa, and S.-i. Amari (1994). Network information criterion-determining the number of hidden units for an artificial neural network model. *IEEE Transactions on Neural Networks 5*(6), 865–872.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the royal statistical society. Series B (Methodological)*, 111–147.

Takeuchi, K. (1976). Distribution of information statistics and validity criteria of models. *Mathematical Science 153*, 12–18.

van der Vaart, A. (1998). *Asymptotic Statistics*. Cambridge University Press, New York.

White, H. (1982). Maximum likelihood estimation of misspecified models. *Econometrica: Journal of the Econometric Society*, 1–25.

Online Appendix to "An information criterion for automatic gradient tree boosting" by Lunde, Kleppe and Skaug

A Derivation of Equation 22

This section derives the CIR limit of stump optimism, as function of split point s. All equation references < 32 are for equations in the main paper.

The derivation relies on results for M-estimators. These results rely on certain regularity conditions, which may be found in van der Vaart (1998) for Theorem 4.21 page 52, but are restated here for convenience. The parameter vector θ is assumed finite-dimensional and to take values in an open subset of Euclidian space, $\theta \in \Theta \subset \mathbb{R}^d$, further, assume $z_1, \ldots, z_n$ to be a sample from some distribution P. The loss function $l(z, \theta)$ needs to be twice continuously differentiable, and we denote its first derivative, the score, as $\psi_\theta(z_i) = \nabla_\theta l(z_i, \theta)$. Parameter estimates, $\hat{\theta}$, are assumed to solve the following estimating equations

$$\frac{1}{n}\sum_{i=1}^{n} \psi_{\hat{\theta}}(z_i) = 0$$

and further consistency with $\hat{\theta}_n \underset{p}{\to} \theta_0$, where θ_0 is the population minimizer, i.e. $E\left[\psi_{\theta_0}(Z)\right] = 0$. Finally we impose conditions on the score. First a Lipschitz condition: For all θ_1 and θ_2 in a neighbourhood of θ_0 and a measurable function H with $E[H^2] \leq \infty$, we assume

$$\left\|\psi_{\theta_1}(z) - \psi_{\theta_2}(z)\right\| \leq H \left\|\theta_1 - \theta_2\right\|.$$

Lastly that $E[\|\psi_{\theta_0}\|^2] < \infty$, and that the map $\theta \mapsto E[\psi_\theta]$ is differentiable at θ_0 with a nonsingular derivative matrix (van der Vaart, 1998).

Note that it is possible to loosen these conditions and still obtain asymptotic normality (needed in Section A.3 and A.4), for example with regards to the differentiability of the score function, the estimating equation need not be exactly zero, but $o_p(n^{-\frac{1}{2}})$, the Lipschitz condition is too stringent, and θ need not be finite dimensional.

However, the gradient boosting approximate loss function we work with, $\hat{l}$, is appropriately differentiable, and allows solutions $\hat{w}$ that are exact zeroes of the estimating equations. While the set of score functions $\{\psi_{\theta_0}(z),\ -\infty < s < \infty\}$ can be established to be a Donsker class (van der Vaart, 1998).

A.1 Insights behind AIC/TIC/NIC

When parameter estimates $\hat{\theta}$ satisfy the regularity conditions in Section A, importantly, the loss l is differentiable in θ and estimates are found by minimizing the loss over data

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{n} l(y_i, f(x_i; \theta)),$$

then the Akaike Information Criterion (AIC) (Akaike, 1974), Takeuchi Information Criterion (TIC) (Takeuchi, 1976) or Network Information Criterion (NIC) (Murata et al., 1994) all result in the optimism estimate (18), for convenience given again here:

$$\hat{C} = \mathtt{tr}\left(E[\nabla^2_\theta l(y_1, f(x_1; \theta_0))] Cov(\hat{\theta}) \right). \tag{32}$$

In the case of TIC and NIC, using the asymptotic normality of $\hat{\theta}$ (see e.g. van der Vaart (1998, Eq. 5.20, p.52)) and the empirical estimator of the Hessian.

AIC follows from assuming that the true data-generating-process is in the family of models being optimized over, and hence asymptotically the $E[\nabla^2_\theta l(y_1, f(x_1); \theta_0)] Cov(\hat{\theta}) = n^{-1} I$. Finally, this result in estimate of the optimism being simply $n^{-1} d$ where d is the number of parameters.

A full derivation of (32) found in Burnham and Anderson (2003, Chapter 7), and we refer to AIC/TIC/NIC for the original articles and derivations. Some insight behind this result is however needed. First, the derivation of (32) relies on the following approximation which according to Slutsky's theorem is valid for large n:

$$n \nabla^2_\theta l(y, f(x; \hat{\theta}))(\hat{\theta} - \theta_0)(\hat{\theta} - \theta_0)^T \approx n \left[\nabla^2_\theta l(y_1, f(x_1; \theta_0)) \right] (\hat{\theta} - \theta_0)(\hat{\theta} - \theta_0)^T, \tag{33}$$

Further, an approximation expressing the difference in test- and training loss is also derived in (Burnham and Anderson, 2003):

$$l(y^0, f(x^0; \hat{\theta})) - l(y_1, f(x_1; \hat{\theta})) \approx (\hat{\theta} - \theta_0)^T \nabla^2_\theta l(y^0, f(x^0; \theta_0))(\hat{\theta} - \theta_0). \tag{34}$$

In the case of a stump CART with fixed split point s, (34) reduces to

$$l(y^0, f(x^0;\hat{\theta})) - l(y_1, f(x_1;\hat{\theta})) \approx 1_{(x^0 \leq s)} \frac{\partial^2}{\partial w_{l,0}^2} l(y^0, w_{l,0})(\hat{w}_l - w_{l,0})^2 + 1_{(x^0 > s)} \frac{\partial^2}{\partial w_{r,0}^2} l(y^0, w_{r,0})(\hat{w}_r - w_{r,0})^2, \tag{35}$$

due to the diagonal Hessian of CART in this case.

In order to characterize the distribution of the right hand side of (35) also under optimization over s, conventional M-estimator asymptotic theory as used in TIC and NIC does not apply directly. This is due to the multiple-comparison problem for different split-points and subsequent selection of $\hat{w} = (\hat{w}_l, \hat{w}_r)$ w.r.t. the training loss which effectively changes the distributions of $\hat{w}_l^2$, $\hat{w}_r^2$ relative to those obtained for fixed s. The next section discuss the distributional change in squares of $\hat{w}$ under profiling.

A.2 A loss function for the deviation from the null-model

Recall that, conditioned on being in a region with prediction w, the relevant Taylor expanded loss (4), modulus unimportant constant terms, is given

$$\hat{l}(y_1, w) = g(y_1, \hat{y}_1)w + \frac{1}{2}h(y_1, \hat{y}_1)w^2.$$

For simplicity we write $g(y_1, \hat{y}_1)$ and $h(y_1, \hat{y}_1)$, with dependence in y_1 and $\hat{y}_1$ as g_1 and h_1 respectively. Let w_t be the constant prediction in the root-node and (w_l, w_r) be the prediction in the left and right descendant nodes. We then write $f_{stump}(x_1;\theta)$ for a stump-model, where the parameter θ holds all relevant information of the tree-stump, namely the split-point, and the left and right weights $\theta = \{s, w_l, w_r\}$.

We start off with rewriting $\hat{l}(y_1, f_{stump}(x_1, \theta))$, such that

$$\omega_i := \hat{l}(y_i, f_{stump}(x_i, \theta)) - \hat{l}(y_i, w_t), \;\; i = 1, \ldots, n, \tag{36}$$

where $\hat{l}(y_1, w_t)$ is the loss of the root model with constant prediction w_t, and hence ω is a measure of deviation from the root model. Loosely speaking, the idea is to calculate how much deviation from the root model we are to expect from pure randomness, and let the split no-split decision calculates w.r.t. this threshold. Further, it is convenient to introduce deviation from root parameters $\tilde{w}_l = w_l - w_t$ and $\tilde{w}_r = w_r - w_t$, and modified first order derivatives $\tilde{g}_i = g_i + h_i w_t$. Then ω might be written

$$\frac{1}{n}\sum_{i=1}^{n} \omega_i = \frac{1}{n}\sum_{i \in I_l} \left(\tilde{g}_i \tilde{w}_l + \frac{1}{2} h_i \tilde{w}_l^2 \right) + \frac{1}{n}\sum_{i \in I_r} \left(\tilde{g}_i \tilde{w}_r + \frac{1}{2} h_i \tilde{w}_r^2 \right). \tag{37}$$

Notice importantly, that $\sum_{i\in I_t} \tilde{g}_i = 0$, which for those familiar with Wiener processes and the functional convergence of estimators might give immediate associations to the Brownian bridge, which indeed follows shortly. Viewing ω as a loss function, the estimates of $\tilde{w}_l$ and $\tilde{w}_r$ are found directly from the score function / estimating equation

$$0 = \nabla_{\tilde{w}} \frac{1}{n} \sum_{i=1}^{n} \omega_i = \frac{1}{n} \sum_{i=1}^{n} \tilde{\psi}_i(\tilde{w}), \;\; \tilde{\psi}_i(\tilde{w}) := \nabla_{\tilde{w}} \omega_i, \tag{38}$$

which we for convenience split into the score function for the left and right estimators $\tilde{\psi}_{i,l}(w)$ and $\tilde{\psi}_{i,r}(w)$. Direct calculation gives

$$\hat{\tilde{w}}_l = \frac{\sum_{i\in I_l} \tilde{g}_i}{\sum_{i\in I_l} h_i} = \hat{w}_l - \hat{w}_t, \;\; \hat{\tilde{w}}_r = \frac{\sum_{i\in I_r} \tilde{g}_i}{\sum_{i\in I_r} h_i} = \hat{w}_r - \hat{w}_t, \tag{39}$$

which verifies that $\tilde{w}_l = w_l - w_t$, and correspondingly for $\tilde{w}_r$.

To directly restate the importance of this specification of the loss: The training loss reduction, R, might now be written only as a function of ω_i's:

$$R = \frac{1}{n} \sum_{i=1}^{n} \hat{l}(y_i, w_t) - \hat{l}(y_i, f_{stump}(x_i, \theta)) = -\frac{1}{n} \sum_{i=1}^{n} \omega_i, \tag{40}$$

and therefore, by using the adjustment factor of R given in (35), to obtain an estimate of R^0, gives

$$\hat{R}^0 = R - \hat{\tilde{w}}_l^2 \frac{\partial^2 \omega^0}{\partial \tilde{w}_l^2} - \hat{\tilde{w}}_r^2 \frac{\partial^2 \omega^0}{\partial \tilde{w}_r^2}, \tag{41}$$

where it is understood that ω^0 obtains as (36) but with (y^0, x^0) in the place of (y_i, x_i), and with parameters at the population minimizer. Note that under the true root model, the population minimizers of $\tilde{w}_l$ and $\tilde{w}_r$ are zero.

Now, an estimate of reduction in generalization loss may be obtained by estimating the expected value. To this end, we need to characterize the joint distribution of the estimator $\hat{\tilde{w}}$ for any given split-point. This distribution is obtained in the preceding sections.

A.3 Asymptotic normality of modified score/estimating equation

The asymptotic normality of $\hat{\tilde{w}}_l$ and $\hat{\tilde{w}}_r$ follows from the convergence of M-estimators to an empirical process. We will make use of the following asymptotic result (van der Vaart, 1998, Theorem 5.21) (Huber, Van der

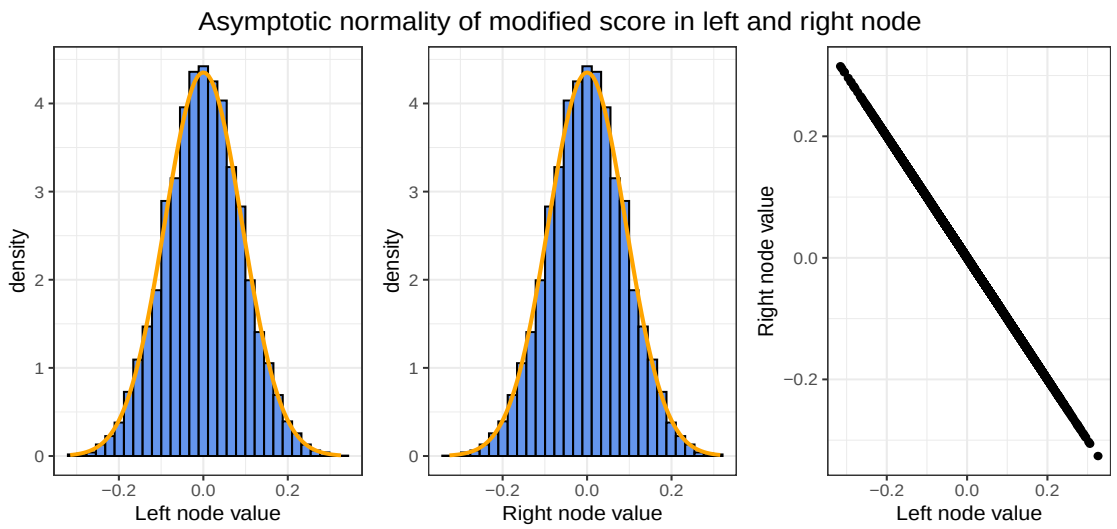

Figure 8: Simulation of the left side of (48), and comparison to the normal on the right. Simulate $n = 100$ observations of $y_i \sim N(3, 1)$ and $x_i \sim U(0, 1)$, to compute derivatives g_i and h_i using the prediction $\hat{y} = 2$ with MSE loss. From this, the modified gradients $\tilde{g}_i$ are computed, and the two (for finite n) quantities on the left in (48) are calculated, using index sets $I_l = \{i : x_i \leq u\}$ and $I_r = \{i : x_i > u\}$ for $u = 0.3$. This experiment is repeated 10000 times to create the observations behind the histogram. Note that the values are completely dependent, as $\sum_{i \in I_l} \tilde{\psi}_{i,l}(\tilde{w}_{l,0}) + \sum_{i \in I_r} \tilde{\psi}_{i,r}(\tilde{w}_{r,0}) = 0$.

Vaart): Let θ be a differentiable parameter satisfying the regularity conditions in Section A, then

$$\sqrt{n}(\hat{\theta} - \theta_0) \underset{p}{\to} E[\nabla^2_\theta l(y, f(x; \theta_0))]^{-1} E[\psi_i(\tilde{w}_0)\psi_i(\tilde{w}_0)^T]. \tag{42}$$

The remaining part of this subsection finds the (joint) empirical process the score converges to. Specifically, the score of $\hat{\tilde{w}}_l$ can be expanded and written as

$$\tilde{\psi}_{i,l}(\tilde{w}_{l,0}) = \psi_{i,l}(w_{l,0}) - \frac{h_i}{\sum h_i}\psi_{i,t}(w_{t,0}), \tag{43}$$

where

$$\psi_{i,l}(w_l) = (g_i + h_i w_l)1_{(x_i \leq s)}, \;\; \psi_{i,t}(w_t) = g_i + h_i w_t, \tag{44}$$

and completely analogous for $\psi_{i,r}$. Let $u \in [0, 1]$ and define the rescaled partial sum

$$S_u = \frac{1}{n} \sum_{i=1}^{\lfloor nu \rfloor} \psi_{i,t}(w_{t,0}). \tag{45}$$

The CLT gives asymptotic convergence of $\sqrt{n}S_u$ to $N\left(0, uE[\psi_{i,t}(w_{t,0})^2]\right)$ for any $u \in [0, 1]$ However, in our

application we need the distribution of S_u for an infinite collection of us. For this purpose, as $\psi_{i,t}(w_{t,0})$s are i.i.d. with finite mean and variance, we may apply Donsker's invariance principle that extends the convergence uniformly and simultaneous over all $u \in [0,1]$. This allow us to write

$$\sqrt{n} S_u \to_d \sqrt{E[\psi_{i,t}(w_{t,0})^2]} W(u), \tag{46}$$

where $W(u)$ is a standard Brownian motion on $u \in [0,1]$. Now, for the index i sorted by x, and defining u from $u = p(x \leq s)$, then $n^{-1} \sum_{i \in I_l} \psi_{i,l} = S_u$ and $n^{-1} \sum_{i \in I_t} \psi_{i,t} = S_1$. Furthermore, from the time reversibility property of the Brownian motion, the same result applies to the right node and $\psi_{i,r}$ but with $(1-u)$ in place of u and perfect negative dependence with that of the left node. Lastly, notice that from the law of large numbers, $\sum_{i \in I_l} \frac{h_i}{\sum_{i \in I_1} h_i} \to_p u$. Thus, when inspecting the asymptotic normality of the score of ω, we might use (43) together with (46) to obtain

$$\sqrt{n} \sum_{i \in I_l} \tilde{\psi}_{i,l}(\tilde{w}_{l,0}) \to_d \sqrt{E[\psi_{i,t}(w_{t,0})^2]} (W(u) - uW(1)) = \sqrt{E[\psi_{i,t}(w_{t,0})^2]} B(u) \tag{47}$$

where $B(u)$ is a standard Brownian bridge on $[0,1]$, i.e. $B(u) \sim N(0, u(1-u))$ and $Cov(B(u), B(v)) = \min\{u,v\} - uv$. Necessarily, the standardized sum of scores of ω in the left and right nodes has the same marginal asymptotic distribution

$$\lim_{n \to \infty} \sqrt{n} \sum_{i \in I_l} \tilde{\psi}_{i,l}(\tilde{w}_{l,0}) \sim \lim_{n \to \infty} \sqrt{n} \sum_{i \in I_r} \tilde{\psi}_{i,r}(\tilde{w}_{r,0}) \sim N\left(0, u(1-u) E\left[\psi_{i,t}(w_{t,0})^2\right]\right), \tag{48}$$

and have perfect negative dependence

$$\sqrt{n} \begin{pmatrix} \sum_{i \in I_l} \tilde{\psi}_{i,l}(\tilde{w}_{l,0}) \\ \sum_{i \in I_r} \tilde{\psi}_{i,r}(\tilde{w}_{r,0}) \end{pmatrix} \underset{p}{\to} \sqrt{E[\psi_{i,t}(w_{t,0})^2]} \begin{pmatrix} B(t) \\ -B(t) \end{pmatrix}, \tag{49}$$

as $\sum_{i \in I_l} \tilde{\psi}_{i,l}(\tilde{w}_{l,0}) + \sum_{i \in I_r} \tilde{\psi}_{i,r}(\tilde{w}_{r,0}) = 0$.

A.4 Asymptotic normality of modified estimator

The remaining part to characterize in (42) is the expected Hessian. This is rather straightforward, as the population equivalent of (37) might be written using indicator functions. Necessarily, the Hessian is diagonal, expectations over indicator functions are probabilities, and its inverse a diagonal matrix with the reciprocal of the diagonal elements of the expected Hessian.

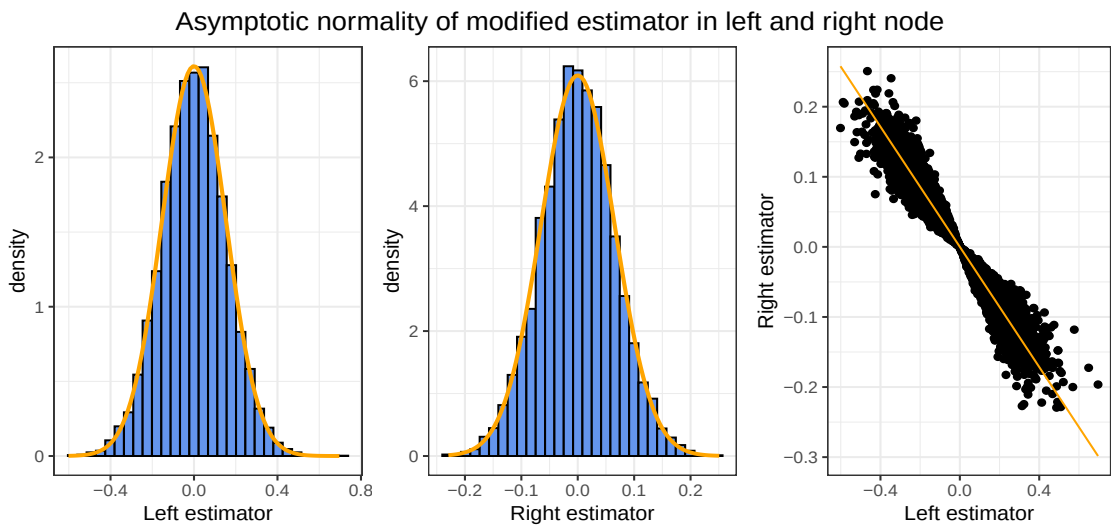

Figure 9: Simulation of (50), where $\hat{\tilde{w}}_l$ and $\hat{\tilde{w}}_r$ are extracted from the simulation experiment explained in the caption of Figure 8, and comparisons with the marginal normal distributions in (51) and (52). Right: Scatter plot of the simulated estimators. Notice that the exact dependence described in (50) is illustrated with an orange line, and that this is of an asymptotic nature. As $n = 100 < \infty$ for this experiment, the scatter plot emits some randomness about the dependence line, but for higher n this deviation from the dependence line tends to zero.

The expected Hessian of the loss in the left node is

$$E\left[\frac{\partial}{\partial \tilde{w}_l}\tilde{\psi}_{i,l}\right] = uE[h]$$

and the right node

$$E\left[\frac{\partial}{\partial \tilde{w}_r}\tilde{\psi}_{i,r}\right] = (1-u)E[h].$$

Further, the off-diagonal elements of the Hessian are zero. The asymptotic distribution therefore may be characterized by

$$\sqrt{n}\begin{pmatrix}\hat{\tilde{w}}_l\\ \hat{\tilde{w}}_r\end{pmatrix} \to_p \begin{pmatrix}\frac{\sqrt{E[\psi_{i,t}(w_{t,0})^2]}}{uE[h]}B(u)\\ -\frac{\sqrt{E[\psi_{i,t}(w_{t,0})^2]}}{(1-u)E[h]}B(u)\end{pmatrix}, \ u \in [0,1]. \tag{50}$$

Notice in particular that (50) implies the marginal limiting distributions

$$\sqrt{n}\hat{\tilde{w}}_l \to_d N\left(0, \frac{1-u}{uE[h]^2}E\left[\psi_{i,t}(w_{t,0})^2\right]\right) \tag{51}$$

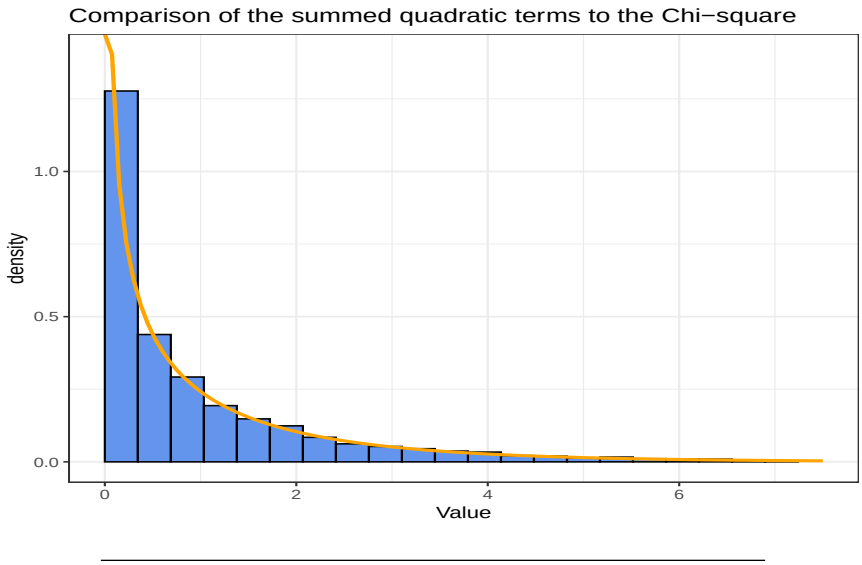

Figure 10: Simulation of the steps in (53) using the squares of $\hat{\tilde{w}}$ in the simulation experiment explained in the caption of Figure 8, and comparison with a Chi-square distribution with one degree of freedom.

and

$$\sqrt{n}\hat{\tilde{w}}_r \to_d N\left(0, \frac{u}{(1-u)E[h]^2} E\left[\psi_{i,t}(w_{t,0})^2\right]\right), \tag{52}$$

but (50) also provides the degenerate dependence structure of $(\hat{\tilde{w}}_l, \hat{\tilde{w}}_r)$.

A.5 Limiting distribution of loss reduction

Returning to taking the expectation w.r.t. (y^0, x^0) of Equation (41); equipped with the joint distribution of $(\hat{\tilde{w}}_l, \hat{\tilde{w}}_r)$ (50), the two terms of (41) can be combined and specified in terms of the single Brownian bridge. To see this, take expectations w.r.t. test data (y^0, x^0), and multiply with $\frac{u(1-u)}{u(1-u)}$ to obtain a common denominator.

$$\begin{aligned} R - E_{y^0,x^0}[R^0] &\approx E_{(y^0,x^0)}\left[\frac{\partial^2}{\partial \tilde{w}_l^2}\omega^0\hat{\tilde{w}}^2 + \frac{\partial^2}{\partial \tilde{w}_r^2}\omega^0\hat{\tilde{w}}^2\right] \\ &= \frac{E\left[\psi_{i,t}(w_{t,0})^2\right]}{nE[h]} \frac{\left((1-u)B(u)^2 + uB(u)^2\right)}{u(1-u)} \\ &= \frac{E\left[\psi_{i,t}(w_{t,0})^2\right]}{nE[h]} \frac{B(u)^2}{u(1-u)}. \end{aligned} \tag{53}$$

The right hand side of (53) gives a convenient asymptotic representation of $R - E_{y^0,x^0}[R^0]$. The subsequent section shows that $B(u)^2/(u(1-u))$, subject to a suitable time-transformation $\tau(u)$, $u \in (0,1)$, is a Cox-Ingersoll-Ross process (Cox et al., 1985) which constitutes the right-hand side of (22). To get from (53) to

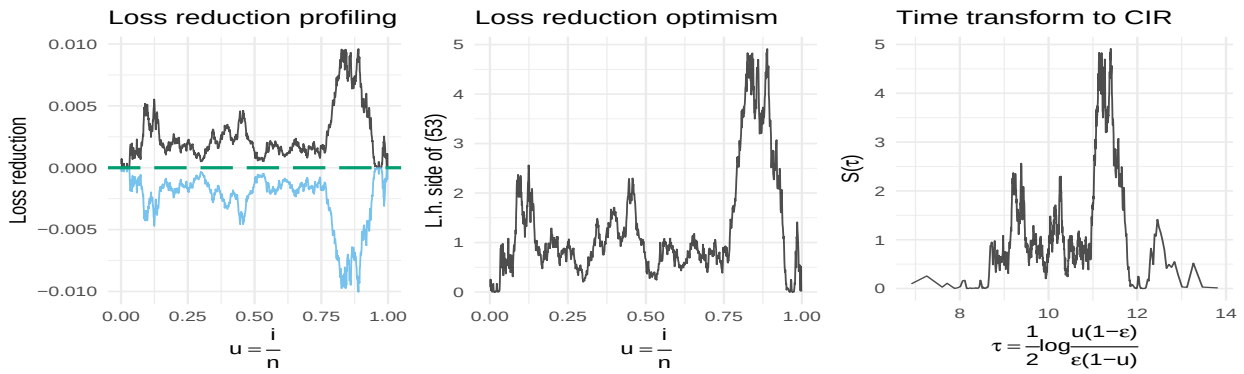

Figure 11: The equivalent and same process as in Figure 2, for loss reduction R and R^0. Left: Reduction in training loss R (black) and reduction in generalization loss $E_{y^0,x^0}\left[R^0\right]$ (blue) as a function of $u = \frac{i}{n}$, defined from the sorted order of x_j. Green long-dashed line is the expected loss-value at $\theta_0 = \lim_{n\to\infty}\hat{\theta}_n$, constant and zero as there is no information in x_j for this instance. Right plot: The transformation of distance between generalization loss and training loss into a CIR process. In this case with no information in feature x_j, choosing the value of s giving the smallest value of training loss in the left plot induces an optimism at the value of the expected maximum of the CIR-process in the right plot.

(22) (modulus the time-transformation) first observe that

$$\hat{C}_{root} = E[h]Var(\hat{w}_t) = E[h]\left(\frac{E\left[\psi_{i,t}(w_{t,0})^2\right]}{nE[h]^2}\right) = \frac{E\left[\psi_{i,t}(w_{t,0})^2\right]}{nE[h]}, \tag{54}$$

and thus simply adding the root optimism on both sides of (53) gives

$$\begin{aligned} R - E_{y^0,x^0}[R^0] + \hat{C}_{root} &= \left[E_{y^0,x^0}\left[\hat{l}(y^0, f(\mathbf{x}^0; \hat{w}_l(u), \hat{w}_r(u)))\right] - \hat{l}(y, f(\mathbf{x}; \hat{w}_l(u), \hat{w}_r(u)))\right] \\ &\approx \hat{C}_{root}\left(1 + \frac{B(u)^2}{u(1-u)}\right). \end{aligned} \tag{55}$$

The final step of the calculations leading to (22) is to show that $B(u)^2/u(1-u)$ is indeed equivalent to the CIR process (23).

A.6 The process $B(u)^2/(u(1-u))$ is a time-transformed CIR process

It was previously mentioned, and used in notation, that $B(u)^2/(u(1-u))$ is a CIR process over time $\tau(u)$, where $u \in (0,1)$. Note that the interval is $u \in (0,1)$ and not $[0,1]$ as for the functional convergence of ψ. This is due to the denominator in $B^2/(u(1-u))$ which almost certainly blows up the value at the endpoints. For this reasons, Miller and Siegmund (1982) approximates the search over $[0,1]$ by $(\epsilon, 1-\epsilon)$ for $\epsilon n > 1$ and $(1-\epsilon)n > 1$, which is of little practical importance, as it makes sense to at least have a few observations when estimating each leaf-weight. Gombay and Horvath (1990) relaxes this assumption, and shows that

the supremum of $B^2/(u(1-u))$ over $(0,1)$ asymptotically have a Gumbel distribution. This result is in alignment with the use the Gumbel distribution in the simulation approach discussed in Section 3.7.

We show that the sum of the scaled-squared Brownian bridge is a Cox-Ingersoll-Ross process. As this paper eventually takes a simulation approach to obtain the distribution of $\max Y(\tau(u)),\ u \in \{u_1, \cdots, u_a\}$, the exact same results would be obtained by simulating $\max \frac{B^2}{u(1-u)},\ u \in \{u_1, \cdots, u_a\}$. Here $u \in \{u_1, \cdots, u_a\}$ are the time-points and probabilities on $(0,1)$, $p(x \leq s)$, for which we observe the process. The specification of the scaled-squared Brownian bridge, through a time-transform, as a CIR is therefore not strictly necessary. However, for completeness, and for the purpose/benefit of working with a time-homogenous stationary process that is well known and studied, we show that this is indeed the case. Important is also the CIR's stationary Gamma distribution, which implies that the CIR is in the maximum domain of attraction of the Gumbel distribution, and warrants its use as an asymptotic approximation to supremums of the CIR.

Anderson et al. (1952) shows that

$$\frac{|B(u)|}{\sqrt{u(1-u)}} = |U(\tau(u))|,\ \tau(u) = \frac{1}{2}\log\frac{u(1-\epsilon)}{\epsilon(1-u)}. \tag{56}$$

where $U(\tau)$ is an Ornstein-Uhlenbeck process which solves the stochastic differential equation

$$dU(\tau) = -U(\tau)d\tau + \sqrt{2}dW(\tau). \tag{57}$$

Notice that (56) is the square root of $B(u)^2/(u(1-u))$ appearing in right-hand side of (53). Hence, obtaining a stochastic differential equation for $B(u)^2/(u(1-u))$ simply amounts to applying Ito's Lemma (Øksendal, 2003) to obtain the stochastic differential equation for the square of $U(\tau)$. More precisely, define $S(\tau) = U(\tau)^2$, which gives the stochastic differential equation given in Equation (23), namely

$$dS(\tau) = 2\left(1 - S(\tau)\right)d\tau + 2\sqrt{2S(\tau)}dW(\tau). \tag{58}$$

This is recognized as a Cox-Ingersoll-Ross (CIR) process (Cox et al., 1985), with speed of adjustment to the mean $a = 2$, long-term mean $b = 1$, and instantaneous rate of volatility $2\sqrt{2}$.

The profiling over loss reduction R for different split points s, the optimism, and the time-transform to the CIR process is illustrated in Figure 11. This is the same experiment as in Figure 2, but with loss reduction profiling instead of stump-loss profiling.

B Maximal CIR as a bound on optimism

Section A shows that $R - E[R^0]$ behaves asymptotically as a CIR process, $S(\tau)$, when profiling over a continuous feature. It immediately follows that a bound on this optimism is given as the expected maximal element of the CIR process

$$E[R - R^0] \leq \hat{C}_{root} E[\max_u S(\tau(u))]. \tag{59}$$

If we are comparing maximum reductions of multiple features, then we would instead be interested in the distribution, $p(\max S(\tau) \leq s)$, for its use in Equation (26), which reduces to the equation above when $m = 1$.

However, more can be said, namely that this bound is tight when the feature being profiled over is independent of the response. To see this, Taylor expand ω_i about its estimate $\hat{\tilde{w}}$ and again make use of the approximation in (33)

$$0 = \frac{1}{n}\sum_{i=1}^{n}(\tilde{g}_i + h_i\tilde{w}_0) \approx \left[\frac{1}{n}\sum_{i=1}^{n}\omega_i\right] + \frac{1}{2}E[h]\left(u\hat{\tilde{w}}_l^2 + (1-u)\hat{\tilde{w}}_r^2\right) \tag{60}$$

since both $\tilde{w}_0$ and $\sum_{i=1}^{n}\tilde{g}_i$ are zero. Rearranging, we may re-express the training loss-reduction

$$R = -\frac{1}{n}\sum_{i=1}^{n}\omega_i \approx \frac{1}{2}E[h]\left(u\hat{\tilde{w}}_l^2 + (1-u)\hat{\tilde{w}}_r^2\right). \tag{61}$$

By recognizing that the term on the right is exactly half the value of (53), it is evident that maximizing R corresponds to selecting split-point and leaf-weights that are at the time-point where the CIR process, $S(\tau(u))$, attains its maximum. Consequently, we obtain equality in Equation (59), i.e.

$$E[\hat{R}^0] = E[R] - \frac{E\left[\tilde{\psi}_{i,t}(\tilde{w}_{t,0})^2\right]}{nE[h]} E\left[\max_u S(\tau(u))\right]. \tag{62}$$

Finally, notice that $E[R - R^0]$ might also be expressed in terms of the optimism of the root and stumps models, so that $E[R - R^0] = \hat{C}_{stump} - \hat{C}_{root}$. Thus, rearranging, we immediately obtain the pure stump optimism, expressed as an adjustment of the root optimism

$$\hat{C}_{stump} = E[R - R^0] + \hat{C}_{root} = \frac{E\left[\tilde{\psi}_{i,t}(\tilde{w}_{t,0})^2\right]}{nE[h]}\left(1 + E\left[\max_u S(\tau(u))\right]\right). \tag{63}$$

A check: In likelihood theory, we would expect one additional degree of freedom, thus $R - R^0 = 1$. Indeed, if we take the final expectation w.r.t. the training data, we have $E[B^2] = u(1-u)$, multiply with

n to obtain a log-likelihood, and assume the expected Hessian equals the variance of the score, then this indeed reduces to exactly 1.

References

Akaike, H. (1974). A new look at the statistical model identification. *IEEE transactions on automatic control 19*(6), 716–723.

Anderson, T. W., D. A. Darling, et al. (1952). Asymptotic theory of certain" goodness of fit" criteria based on stochastic processes. *The annals of mathematical statistics 23*(2), 193–212.

Burnham, K. P. and D. R. Anderson (2003). *Model selection and multimodel inference: a practical information-theoretic approach*. Springer Science & Business Media.

Cox, J. C., J. E. Ingersoll, and S. A. Ross (1985). A theory of the term structure of interest rates. *Econometrica 53*(2), 385–407.

Gombay, E. and L. Horvath (1990). Asymptotic distributions of maximum likelihood tests for change in the mean. *Biometrika 77*(2), 411–414.

Miller, R. and D. Siegmund (1982). Maximally selected chi square statistics. *Biometrics*, 1011–1016.

Murata, N., S. Yoshizawa, and S.-i. Amari (1994). Network information criterion-determining the number of hidden units for an artificial neural network model. *IEEE Transactions on Neural Networks 5*(6), 865–872.

Øksendal, B. (2003). *Stochastic differential equations*. Springer.

Takeuchi, K. (1976). Distribution of information statistics and validity criteria of models. *Mathematical Science 153*, 12–18.

van der Vaart, A. (1998). *Asymptotic Statistics*. Cambridge University Press, New York.