Information in Local Curvature: Three Papers on Adaptive Methods in Computational Statistics

by

Berent Ånund Strømnes Lunde

Thesis submitted in fulfilment of
the requirements for the degree of
PHILOSOPHIAE DOCTOR
(PhD)

Faculty of Science and Technology
Department of Mathematics and Physics
2020

University of Stavanger
NO-4036 Stavanger
NORWAY
www.uis.no

ISBN:
ISSN:
PhD: Thesis UiS No.

Preface

This thesis is submitted in partial fulfilment of the requirements for the degree of Philosophiae Doctor (PhD) at the University of Stavanger, Faculty of Science and Technology, Norway. The research has been carried out at the University of Stavanger from September 2017 to August 2020.

The thesis consists of an introduction to relevant background theory, topics and ideas, and the following three papers:

Paper I

Lunde, Berent Ånund Strømnes, Tore Selland Kleppe, and Hans Julius Skaug (2020). Saddlepoint-adjusted inversion of characteristic functions. *Submitted for publication in Computational Statistics and Data Analysis.*

Paper II

Lunde, Berent Ånund Strømnes, Tore Selland Kleppe, and Hans Julius Skaug (2020). An information criterion for automatic gradient tree boosting. *Submitted for publication in The Journal of the Royal Statistical Society, Series B (Statistical Methodology).*

Paper III

Lunde, Berent Ånund Strømnes, and Tore Selland Kleppe (2020). agtboost: Adaptive and Automatic Gradient Tree Boosting Computations. *To be submitted for publication in Journal of Statistical Software.*

Acknowledgements

I would like to thank my supervisor, Professor Tore Selland Kleppe, for his constant support and invaluable guidance. You have allowed me to pursue ideas, to fail, and so many times steered me in the right direction with detailed feedback and questions that would uncover flaws, but which would eventually lead me closer towards truth. Thank you for your encouraging words, enthusiasm, and genuine thoughtfulness.

Thanks are also due to my co-supervisors, the professors Hans Julius Skaug and Jan Terje Kvaløy. Professor Skaug has co-authored two of the papers in this thesis, introduced me to the statistics community in Bergen, and always shared of his time and wide experience, for this I am grateful. Professor Kvaløy has been presence of constant cheerfulness and inspiration. Thank you for always taking a genuine interest in people and their ideas. I extend my thanks to my fellow PhD-students. Utmost appreciation goes out to Kjartan Kloster Osmundsen and Birthe Aarekol, for the many discussions and several enjoyable trips to academic conferences and meetings around the world.

Finally, I want to give many thanks to my friends, utmost to Kjetil, for having accommodated me in Stavanger, and allowed me with his companionship to escape stress. And also my family, in particular my mother, Katrin, for invaluable advice, and my wife, Saeron Min, which has been a force of continued support. You have let me dive into hours of silent thoughts, calculations and coding when it was needed, but have also pulled me away and forced upon me a more balanced life when I would encounter a runtime error of the mind, but of the kind which I would not be able to see or solve by myself.

Berent Ånund Strømnes Lunde
Stavanger, August 2020

Abstract

Advanced statistical computations have become increasingly important, as with the increased flexibility of models capturing complex relationships in new data and use-cases, comes increased difficulties of fitting procedures for the models. For example, if the model is complex, involving multiple sources of randomness, then the probability density function used in maximum likelihood estimation typically does not have a closed form. On the other hand, in regression type problems the closed form of the assumed conditional distribution of the response is often known. However, the relationship between features and response can be complex, high dimensional and is generally unknown, motivating non-parametric procedures that come with new sets of fitting problems.

This thesis explores techniques utilizing the local curvature of objective functions, and using the information inherent in this local curvature, to create more stable and automatic fitting procedures. In the first paper, a saddlepoint adjusted inverse Fourier transform is proposed. The method performs accurate numerical inversion, even in the tails of the distribution. This allow practitioners to specify their models in terms of the characteristic function which often exists in closed form. The second paper proposes an information criterion for the node-splits, after greedy recursive binary splitting, in gradient boosted trees. This alleviates the need for computationally expensive cross validation and expert opinions in tuning hyperparameters associate gradient tree boosting. The third paper focuses on the implementation of the theory presented in the second paper into the R package agtboost, and also builds on the information criterion to suggest an adjustment of ordinary greedy recursive binary splitting, better suited to gradient tree boosting.

Table of Contents

Preface iii

Acknowledgements iv

Abstract vi

1 Introduction 1

2 Maximum likelihood and supervised learning 3
2.1 Maximum likelihood estimation 3
2.2 Supervised learning 3

3 Quadratic approximations in statistics 5
3.1 The saddlepoint approximation 6
3.2 Gradient tree boosting 9
3.3 Distribution of estimated parameters 15
3.4 Model selection 17

4 Computational remarks 21

5 Summary of the papers 23

References 25

Appendix

Saddlepoint-adjusted inversion of characteristic functions 31

An information criterion for automatic gradient tree boosting 53

agtboost: Adaptive and Automatic Gradient Tree Boosting Computations 96

1 Introduction

Advanced statistical methods and procedures have seen increased and widespread usage in later years. This is backed by access to more data and new use-cases, cheaper computational power, and adoption into mainstream languages such as Python and R. Underlying this trend is also the increased usability of said algorithms, in regards to training on data and putting them into production. The goal of this thesis is to further the usability of computational methods in statistics with regards to stability, speed, and automatic functionality.

The main approach of the present work is to, in some loose and wide sense, approximate some objective function with a local quadratic approximation to either solve stability issues, create dynamic step-lengths, or measure the uncertainty of estimators. The hope is that the iterative methods that the local quadratic approximation is somehow applied to, will see an increased adaptivity to individual data and problems and correspondingly decrease the manual tuning performed.

The first part of this thesis will give a brief and informal introduction to the concepts and techniques that are used in papers I-III. The basis is the objectives of maximum likelihood and supervised learning, which are presented in the Section 2. Section 3 introduces the local quadratic approximation and showcase it in the relevant use-cases for papers I-III, i.e. maximum likelihood numerical optimization, the saddlepoint approximation, gradient tree boosting, some asymptotic theory and model selection. Section 4 gives a summary of the important computational frameworks for this thesis, while Section 5 summarises the papers of the thesis.

2 Maximum likelihood and supervised learning

Maximum likelihood estimation and supervised learning are briefly introduced in an informal manner. This is done to provide motivation and understanding of the fundamental objectives for the numerical algorithms that are presented and improved upon in this thesis.

2.1 Maximum likelihood estimation

Let $\mathbf{y}$ denote an n-dimensional vector of observations from a parametric distribution, with density denoted $p_Y(\mathbf{y}; \theta_0)$, where $\theta_0 \in \Theta$, $\Theta \subseteq \mathrm{R}^p$ is a p-dimensional vector. It is often the case that a reasonable parametric family of functions, $p_Y(\mathbf{y}; \theta)$, $\theta \in \Theta$, can be inferred from the problem and from inspection of the data. However, θ_0 will be unknown, and it is reasonable to estimate it using the observed data $\mathbf{y}$. To this end, maximum likelihood estimation is a popular approach. The maximum likelihood estimate (MLE) is the value of θ in Θ which maximizes the probability of the data, i.e. the likelihood. Equivalently, the value of θ that minimizes the negative log likelihood

$$\hat{\theta} = \arg\min_{\theta} \{-\log p_Y(\mathbf{y}; \theta)\}. \tag{2.1}$$

The maximum likelihood estimate, $\hat{\theta}$, is under suitable regularity conditions asymptotically unbiased, efficient, and asymptotically normal. See van der Vaart (1998) for a treatment of asymptotic properties.

2.2 Supervised learning

The supervised learning objective is perhaps easiest stated as a "regression" type objective, but also bears resemblance to maximum likelihood estimation. Assume now that there is access to a dataset $\{y_i, \mathbf{x}_i\}_{i=1}^n$ consisting of n observations of a response $y_i \in R$ and m features (or

covariates) $\mathbf{x}_i \in R^m$ that are indexed by i. In general, individual response observations, y_i, could also be multidimensional, but throughout this thesis they are assumed one-dimensional. Let $\hat{y}_i$ be a prediction for y_i and let the loss function $l(y_i, \hat{y}_i)$ be a function measuring the difference between a response and its prediction. The supervised learning objective is to find the best possible predictive function, $f(\mathbf{x}) = \hat{y}$, which takes a feature vector as its argument, and outputs a prediction $\hat{y}$. "Best possible" is here in reference to the loss, l, over observations not part of the "training" dataset $\{y_i, \mathbf{x}_i\}_{i=1}^n$ used to fit or learn f. More formally, we seek f so that

$$\hat{f} = \arg\min_f \left\{ E\left[l(y^0, f(\mathbf{x}^0)) \right] \right\}, \tag{2.2}$$

where the superscripts of $(y^0, \mathbf{x}^0)$ indicate an observation of a response and feature vector unseen in the training data, and E denotes the expectation. Notice that, if the search is constrained over a parametric family of functions indexed by $\theta \in \Theta$, and the loss function is taken to be the negative log-likelihood, $l = -\log p$, then the supervised learning objective is closely related to the objective of maximum likelihood estimation (2.1) in a regression setting. In fact, the objective in (2.1) scaled with n^{-1} will then be a sample estimator of the expected value in (2.2), but biased downwards in expectation, as evaluation is done over observations in the training set.

3 Quadratic approximations in statistics

The maximum likelihood objective (2.1) and supervised learning objective (2.2) are, except for the most trivial of cases, not straightforward, and must be solved numerically. This then typically involve some iterative algorithm, which may require substantial manual tuning and trial and error before successful application. However, a local quadratic approximation to some otherwise intractable function can often be of help in making these algorithms more automatic and adaptive to the relevant data.

When referring to a local quadratic approximation, as is frequently done in this thesis, it is meant to refer to a 2'nd order Taylor approximation of a function $f(x)$, about some point x_0. For example, the quadratic approximation of the loss function l about some value of θ, say θ_k, gives

$$\begin{aligned} l(y_i, f(\mathbf{x}_i;\theta)) \approx\ & l(y_i, f(\mathbf{x}_i;\theta_k)) + \nabla_\theta l(y_i, f(\mathbf{x}_i;\theta_k))(\theta - \theta_k) \\ & + \frac{1}{2}(\theta - \theta_k)^\intercal \nabla^2_\theta l(y_i, f(\mathbf{x}_i;\theta_k))(\theta - \theta_k). \end{aligned} \tag{3.1}$$

Example 3.0.1 (Newton-Raphson) *Consider the regression setting with a negative log-likelihood*

$$L(\theta) = \sum_i^n l(y_i, f(\mathbf{x}_i;\theta)),$$

where l is the negative log-probability of $y|\mathbf{x}$ and f is a predictive function parametrized by θ. The MLE $\hat{\theta}$ in (2.1) *typically has to be found numerically, as the score equations, $0 = \nabla_\theta L(\theta)$, are not possible to solve analytically for θ. Assuming that L is differentiable and convex in θ, the Newton-Raphson algorithm will converge to the MLE $\hat{\theta}$. The iterative Newton-Raphson algorithm is constructed by employing the r.h.s. of* (3.1) *iteratively to the current value of θ, say θ_k, the next value in the iterative algorithm is then given by*

$$\theta_{k+1} = \theta_k - \left[\nabla^2_\theta L(\theta_k)\right]^{-1} \nabla_\theta L(\theta_k).$$

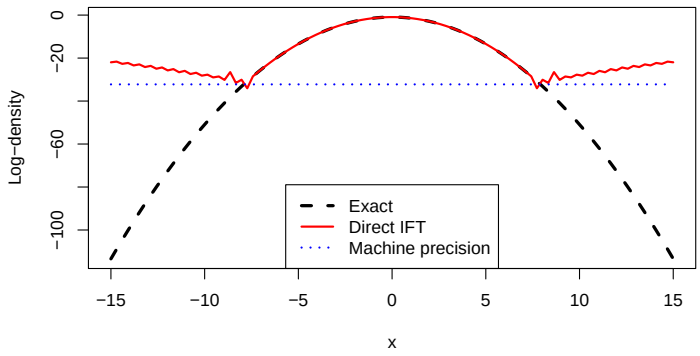

Figure 3.1: Figure included from Paper I. Illustrating the dominance of inaccuracies of the IFT (3.2), calculated with quadrature, at machine precision at $\log(1.0 \times 10^{-14})$ indicated by the dotted horizontal line.

There is an abundance of problems in computational statistics that may be helped by some application of (3.1). Only a few of these are however discussed further, namely the ones that are relevant to papers I-III. The following sections discuss applications of local quadratic approximations as helpful tools in dealing with some of the numerical problems associate optimization of (2.1) and (2.2).

3.1 The saddlepoint approximation

It is often the case that the density $p_X(x;\theta)$ of a random variable X, is not available in closed form when there are multiple sources of randomness present in X. Direct optimization of (2.1) is therefore difficult. However, the characteristic function (the Fourier transform of the density), $\varphi_X(s) = E[\exp(isX)]$ often exist in closed form, even in situations with more than one source of randomness. The density can then be retrieved by numerically evaluating the inverse Fourier

transform

$$p_X(x;\theta) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \varphi_X(s;\theta)e^{-isx}ds = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{K_X(is;\theta)-isx}ds, \quad (3.2)$$

where $K_X(s;\theta) = \log \varphi_X(-is;\theta)$ is the cumulative generating function (CGF) and the last equality holds due to symmetry.

However, consider the case of numerical MLE optimization, for example by using the Newton-Raphson algorithm in Example 3.0.1. Here, at the first iteration, say θ_1, the initial estimate is likely to start at values far from the population MLE, θ_0. Necessarily, observations x will take place in low-density regions of $p(x;\theta_1)$, and this will continue to be the case at subsequent iterations, until θ_k is close to θ_0. This constitutes a problem to direct numerical inversion of (3.2) using quadrature schemes (weighted sum of integrand evaluations), as numerical inaccuracies related to the (binary) representation of floating-point numbers will dominate. More specifically, considering double precision at order 1.0×10^{-16}, if x is in a region with log-density $\log p(x;\theta_k)$ smaller than this value, the inaccuracy of the binary representation is sure to dominate. Even more is that such unwanted behaviour in practice happens a few orders of magnitude higher than the theoretical limit given above. In Figure 3.1, the error dominates already at 1.0×10^{-14}.

An inversion technique that does not suffer from erroneous computations in low-density regions, and in fact is renowned for its tail-accuracy (under regularity conditions, see Barndorff-Nielsen and Kluppelberg (1999)), is the saddlepoint approximation (SPA) (Daniels, 1954). It is developed in paper I through an argument of exponential tilting, which takes place on the "time-domain" side of the Fourier transform. Complimentary, an argument on the "frequency-domain" side is given here, that closely follows the derivation in Butler (2007). First, notice that the value of the integral in (3.2) is unchanged if we integrate through a line parallel to the imaginary axis, say τ,

$$p_X(x;\theta) = \frac{1}{2\pi}\int_{\infty}^{\infty} e^{K_X(\tau+is;\theta)-(\tau+is)x}ds. \quad (3.3)$$

Now, apply the quadratic approximation (3.1) to the log-integrand, $K_X(\tau + is; \theta) - (\tau + is)x$, locally about the value of τ solving the saddlepoint equations

$$\hat{\tau} = \arg\min_{\tau}\{K_X(\tau; \theta) - \tau x\}, \tag{3.4}$$

henceforth called the saddlepoint. This then gives the approximation of the log-integrand

$$K_X(\hat{\tau} + is; \theta) - (\hat{\tau} + is)x \approx K_X(\hat{\tau}) - \hat{\tau}x - \frac{1}{2}\frac{d^2}{d\tau^2}K_X(\hat{\tau})s^2. \tag{3.5}$$

Inserting this into the integral, and performing the transformation $u = \sqrt{\frac{d^2}{d\tau^2}K_X(\hat{\tau})}s$ gives the ordinary SPA as

$$\begin{aligned} p_X(x; \theta) &\approx \frac{\exp(K_X(\hat{\tau}) - \hat{\tau}x)}{2\pi\sqrt{\frac{d^2}{d\tau^2}K_X(\hat{\tau})}} \int_{-\infty}^{\infty} e^{-\frac{1}{2}u^2} du \\ &= \frac{\exp(K_X(\hat{\tau}) - \hat{\tau}x)}{\sqrt{2\pi\frac{d^2}{d\tau^2}K_X(\hat{\tau})}} = spa_X(x; \theta). \end{aligned} \tag{3.6}$$

The SPA (3.6) is often accurate, it is asymptotically exact in n if there is some asymptotic normality underlying X, for example for $X = n^{-1}\sum_i X_i$, and the previously mentioned tail-accuracy is a property that is highly tractable. Furthermore, computation is very fast if the inner problem (3.4) can be solved analytically but may also be efficiently found by numerical optimization. The SPA can be implemented using automatic differentiation (Griewank and Walther, 2008), so that solving the inner problem and evaluating (3.6) is automatic for users, which only needs to implement the characteristic function.

On the downside, the SPA does not integrate to one, except for in a few special cases. Also, the approximation is often unimodal even if the target density is multimodal, which could very well be the case when X consists of multiple sources of randomness. A common technique is to multiply the SPA with a constant value c, where $c^{-1} = \int spa_X(x; \theta)dx$,

that ensure $cspa_X(x;\theta)$ to be a density. This is immediately more computationally costly, requires bespoke implementation, and does not solve the problems of unimodality. These problems are the subject of Paper I.

3.2 Gradient tree boosting

Algorithm 1 Second-order generic gradient boosting Hastie et al. (2001)

Input:
- A training set $\{(y_i, \mathbf{x}_i)\}_{i=1}^n$
- A differentiable loss function $l(\cdot,\cdot)$
- A learning rate $\delta \in (0, 1]$
- Number of boosting iterations K
- Type of statistical model $\mathcal{F}$

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:
   i) Compute derivatives according to (3.9)
   ii) Fit a statistical model $\tilde{f} \in \mathcal{F}$ to derivatives using (3.10)
   iii) Scale the model with the learning rate
   $f_k(\mathbf{x}) = \delta \tilde{f}(\mathbf{x})$
   iv) Update the model:
   $f^{(k)}(\mathbf{x}) = f^{(k-1)}(\mathbf{x}) + f_k(\mathbf{x})$

end for

3. Output the model: **Return** $f^{(K)}(\mathbf{x})$

The idea behind gradient boosting emerged in Friedman (2001) and in Mason et al. (1999) as a way to approximate functional gradient descent for the optimization problem in (2.2). It is conceptually similar to how the Newton-Raphson algorithm from Example 3.0.1 solves the optimization problem in (2.1). Given an initial function $f^{(k-1)}(\mathbf{x})$, one

Algorithm 2 Greedy recursive binary splitting, from Paper II

Input:

- A training set with derivatives and features $\{\mathbf{x}_i, g_{i,k}, h_{i,k}\}_{i=1}^n$

Do:

1. Initialize the tree with a constant value $\hat{w}$ in a root node:
   $\hat{w} = -\frac{\sum_{i=1}^n g_{i,k}}{\sum_{i=1}^n h_{i,k}}$
2. Choose a leaf node t and let I_{tk} be the index set of observations falling into node t
   For each feature j, compute the reduction in training loss
   $$\mathcal{R}_t(j, s_j) = \frac{1}{2n}\left[\frac{\left(\sum_{i\in I_L(j,s:j)} g_{ik}\right)^2}{\sum_{i\in I_L(j,s_j)} h_{ik}} + \frac{\left(\sum_{i\in I_R(j,s_j)} g_{ik}\right)^2}{\sum_{i\in I_R(j,s_j)} h_{ik}} - \frac{\left(\sum_{i\in I_{tk}} g_{ik}\right)^2}{\sum_{i\in I_{tk}} h_{ik}}\right]$$
   for different split-points s_j, and where
   $I_L(j, s_j) = \{i \in I_{tk} : x_{i,j} \leq s_j\}$ and $I_R(j, s_j) = \{i \in I_{tk} : x_{i,j} > s_j\}$
   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.

$x_{i,j}$ is the j'th element of the i'th feature vector.

ideally seeks a function $f_k(\mathbf{x})$ minimizing

$$\hat{f}_k(\mathbf{x}) = \arg\min_{f_k} E\left[l\left(y, f^{(k-1)}(\mathbf{x}) + f_k(\mathbf{x})\right)\right]. \tag{3.7}$$

If this optimization problem is difficult (as it usually is), a reasonable substitute to $\hat{f}_k$ is to find the functional derivative of this objective and add the negative direction to the model, say $f^{(k)} = f^{(k-1)} + f_k$, and then repeat the procedure until termination at iteration K, which would yield the final model $f^{(K)} = f_0 + \cdots + f_K$ when starting with an initial model f_0.

Difficulties arise to this procedure, as the joint distribution of $(y, \mathbf{x})$ is generally unknown. Therefore, the expectation cannot be computed

explicitly, and neither can the functional derivative. The immediate solution to the unknown distribution and expectation, if there is access to a dataset $\mathcal{D}_n = \{y_i, \mathbf{x}_i\}_{i=1}^n$ of independent observations, is to average the loss over these observations, and instead, at iteration k, seek

$$\hat{f}_k(\mathbf{x}) = \arg\min_{f_k} \frac{1}{n} \sum_{i=1}^{n} l\left(y_i, f^{(k-1)}(\mathbf{x}_i) + f_k(\mathbf{x}_i)\right), \tag{3.8}$$

approximately to second order. This is done by using the current model, $f^{(k-1)}$, to compute predictions $\hat{y}_i^{(k-1)} = f_0(\mathbf{x}_i) + \cdots + f_{k-1}(\mathbf{x}_i)$, and then derivatives for observations in the sample as

$$g_{i,k} = \frac{\partial}{\partial \hat{y}_i} l(y_i, \hat{y}_i^{(k-1)}), \;\; h_{i,k} = \frac{\partial^2}{\partial \hat{y}_i{}^2} l(y_i, \hat{y}_i^{(k-1)}). \tag{3.9}$$

These are used in a 2'nd order approximation to the original loss

$$\begin{aligned} \hat{f}_k(\mathbf{x}) &= \arg\min_{f_k} \frac{1}{n} \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(k-1)}) + g_{i,k} f_k(\mathbf{x}_i) + \frac{1}{2} h_{i,k} f_k(\mathbf{x}_i)^2 \\ &= \arg\min_{f_k} \frac{1}{n} \sum_{i=1}^{n} g_{i,k} f_k(\mathbf{x}_i) + \frac{1}{2} h_{i,k} f_k(\mathbf{x}_i)^2 \end{aligned} \tag{3.10}$$

which is a quadratic-type loss amenable to fast optimization. Again, the quadratic approximation (3.1) has been of help. Now, a function that completely minimizes the above sample loss (both the original and/or the approximate), is likely to adapt to the inherent randomness in the sample. Furthermore, a search over all possible functions is obviously infeasible. For these reasons, the search is constrained to a family of functions that admits fast fitting routines, and that are somehow constrained and thus less likely to overfit. Typical families include linear functions, local regression using kernels, or most popularly, trees.

Gradient boosting emerges as the collection of the above-mentioned ideas: Initially, start with a constant prediction $f_0(\mathbf{x}) = \eta$, then iteratively, compute derivative information or pseudo-residuals through (3.9), and fit a statistical model $\tilde{f}_k$ to these observations using (3.10).

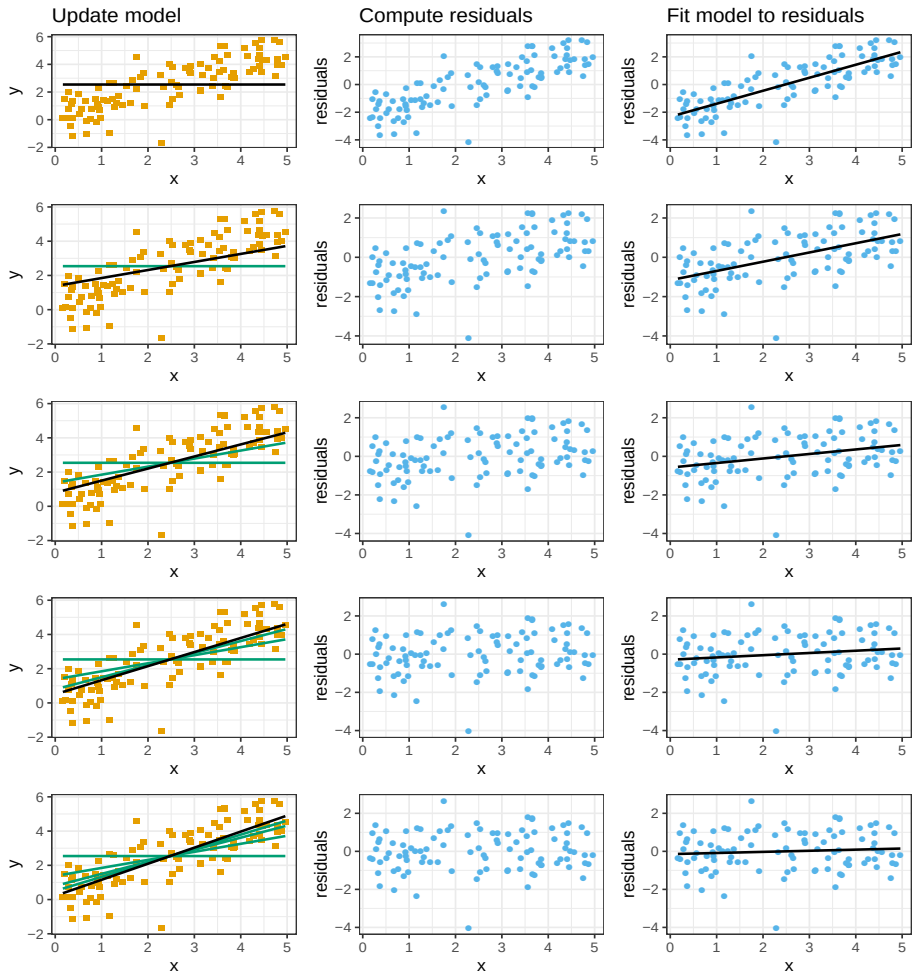

Figure 3.2: Illustration of gradient boosting with linear functions as base learners and the MSE loss. First column shows the observations of target y (orange squares) as a function of a one-dimensional feature, x, and how the ensemble is updated according to $f^{(k)}(x) = f^{(k-1)}(x) + \delta f_k(x)$. The first model is initialized with a constant value. In the second column, residuals (blue circles) are computed from the model f^{k-1}. In the third column, a linear model $f_k(x)$ is fit to the residuals.

The final ingredient of gradient boosting is to shrink the model by some constant $\delta \in (0, 1]$, $\hat{f}_k = \delta \tilde{f}_k$ to make space for new models, and add

it to the ensemble model $f^{(k)} = f^{(k-1)} + \hat{f}_k$. The gradient boosting pseudo-algorithm is given in Algorithm 1. Note that this is the modern type gradient boosting algorithm, slightly different from the original algorithm of Friedman (2001), which is a first-order type algorithm, that would fit the model using mean-squared-error loss, then scale the model with an optimized constant value and finally shrink it. A visual illustration of the steps in gradient boosting with linear functions as the base learners is given in Figure 3.2. The loss is the mean squared error (MSE) loss, corresponding to a Gaussian negative log-likelihood. MSE is especially amenable to visual illustrations, as residuals are equal to the pseudo-residuals $(y_i - \hat{y}_i^{(k-1)}) = -g_{i,k}/h_{i,k}$.

The choice of statistical model to fit to the pseudo-residuals is not arbitrary. The popular choice is to use classification and regression trees (CART) (Breiman et al., 1984), which gives gradient tree boosting (GTB), the boosting type that has dominated in many machine-learning competitions since the introduction of xgboost (Chen and Guestrin, 2016). Using CART as weak learners can be motivated by first considering linear functions, $\hat{y} = \beta^\intercal \mathbf{x}$ used in the illustrations in Figure 3.2. Here only a portion of the full estimate of the $\hat{\beta}_j$ decreasing (3.10) the most is used. This then resembles a type of shrunken forward stagewise procedure, which is closely related to computing LASSO solution paths (Hastie et al., 2001). Boosting thus holds the possibility of efficiently building sparse models in the face of high-dimensional problems, by excluding individual features x_j that does not contribute to significant decreases in (3.10). If a weak learner is used that fits and use all features simultaneously, this pathology of boosting is likely to disappear.

If CART is fit using greedy binary splitting (Algorithm 2), then features are included into the model sequentially by the learning procedure. Furthermore, in contrast to linear functions, CART can learn non-linear functions and interaction effects automatically. In essence, GTB may learn sparse, non-linear models with complex interaction effects efficiently, while the shrinkage δ applied to each tree will smooth out

the piecewise constant functions. Model complexity may range from the constant model, to high-dimensional non-linear functions, and in between these two extremes often lie a model that may decrease the objective in (2.2) more so than other types of models. Figure 3.3 illustrates this. A GTB model is fit to 100 training observations and tested on 100000 test observations, producing the different types of loss against boosting iterations in the top plot. This plot indicates that stopping the boosting procedure at $K = 30$ will give the best of the possible candidate models. At iteration $K = 1$ the model is too simple and is almost not adapted to the training data. At the other end of the spectrum, the model has an almost perfect fit to the training set at the final iteration $K = 1000$, indicated by the training loss convergence towards zero and also by visual inspection of the bottom right plot. Clearly, this complex model has adapted too much to the noise in the data.

Gradient boosting originally has two hyperparameters, namely the number of boosting iterations K, and shrinkage or the learning rate δ, which is usually set to some "small" value. Using CART as weak learners introduces additional tuning to control the complexity of individual trees. Friedman et al. (2000) suggests global hyperparameters fixed equally for all trees, as the greedy recursive binary splitting algorithm is optimized as if the current boosting iteration is the last iteration. Important hyperparameters are the maximum depth of trees, a maximum number of leaves or terminal nodes in a tree, and a threshold for minimum reduction in training loss (3.10) if a split is to be performed. Hyperparamters are typically learned using k-fold cross-validation (CV) (Stone, 1974), which increase computation times significantly.

In summary, GTB is a powerful approach to solving regression-type problems in supervised learning (2.2), practically made possible by iterative quadratic approximations of the loss about the predictions of the current model. There are, however, multiple hyperparameters that must be tuned for each separate dataset. This is computationally costly when done with CV, bothersome, and not always straight-forward. The

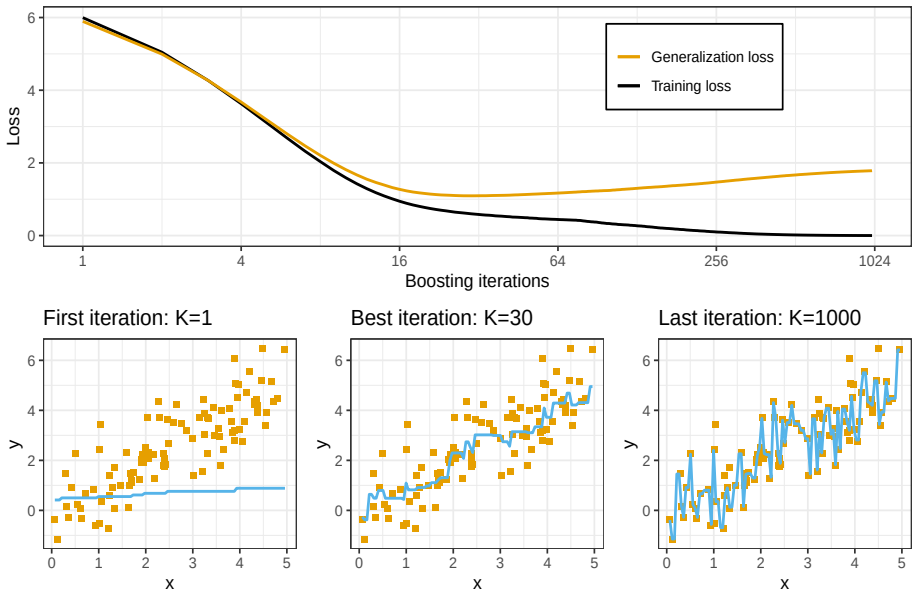

Figure 3.3: Top: Training and test (generalization) loss versus the number of boosting iterations for a GTB model. Notice the logarithmic horizontal axis. The training set consists of 100 observations and the test set of 100000. All observations are i.i.d. sampled $X \sim U(0,5)$ $Y \sim N(X,1)$. Bottom: The fit at iterations $K \in \{1, 30, 1000\}$ of the GTB model to the training data.

removal of GTB hyperparameters is the subject of Paper II.

3.3 Distribution of estimated parameters

A key to solving problems of the type in (2.1), which there are multiple of in the boosting solution to (2.2), is to use the distribution of estimated parameters to evaluate the significance of the model. This can for example be used to control complexity of the model, or to reject alternative hypothesis.

There are multiple ways of approximating the distribution of estimated quantities. The perhaps most straight-forward method if observations

are independent, is the bootstrap (Efron, 1992). The idea is that if the distribution, $P_{Y,\mathbf{X}}(y, \mathbf{x})$, behind the true data-generating process is known, then a large number, say B, of size-n datasets, i.i.d. of the training data $\{y_i, \mathbf{x}_i\}_{i=1}^n$, could be sampled. Then the fitting procedure could be performed for each sampled-dataset, and finally statistical methods could be used to investigate the sampled quantities. However, the true distribution $P_{Y,\mathbf{X}}$ is of course generally unknown. The idea of the bootstrap is to exchange $P_{Y,\mathbf{X}}$ with the empirical distribution, $P^*_{Y,\mathbf{X}}(y, \mathbf{x}) = n^{-1} \sum_{i=1}^n 1(y_i \leq y, \mathbf{x}_{i,1} \leq x_1, \cdots, \mathbf{x}_{i,m} \leq x_m)$, and then perform the above-mentioned procedure.

Sampling procedures are in general quite expensive, and this is no different for the bootstrap. At early experimental stages of this work, the SPA was used together with the idea of the empirical distribution, to retrieve necessary density approximations while avoiding costly sampling. The idea is to use the empirical CGF, $K^*_Y(s) = \log\left(\sum_{i=1}^n e^{sy_i}\right) - \log n$ for n observations of a one-dimensional random variable Y, in the SPA (3.6), from which desired results can be drawn. This is explained in Butler (2007, Chapter 14), and Chapter 12.2 of the same book for the ratio estimators appearing in Algorithm 2. The goal was a computationally efficient version of the Efron Information Criterion (EIC) (Ishiguro et al., 1997), but was abandoned due to concerns regarding stability and speed of computations, in comparison to analytical asymptotic results.

Often the most computationally efficient procedures are analytical results, which may be obtained for estimated quantities through asymptotics. Hence, again, it will seem that the quadratic approximation of some objective function about some local point of optimality is useful: The central limit theorem may be applied to the score equations, $0 = n^{-1} \sum_{i=1}^n \nabla_\theta l(y_i, f(\mathbf{x}_i; \hat{\theta}))$, to obtain asymptotic normality, and from this, asymptotic normality of estimated parameters (under certain regularity conditions, see van der Vaart (1998)) can be obtained through

the delta method as

$$\begin{aligned}\sqrt{n}^{-1}(\hat{\theta}-\theta_0) &\sim N\left(0, J(\theta_0)^{-1}I(\theta_0)\left[J(\theta_0)^{-1}\right]^\intercal\right), \qquad (3.11)\\ J(\theta) &= E[\nabla^2_\theta l(y,\mathbf{x};\theta))],\\ I(\theta) &= E\left[\nabla_\theta l(y, f(\mathbf{x});\theta))\nabla_\theta l(y, f(\mathbf{x});\theta))^\intercal\right].\end{aligned}$$

Estimates of J and I can be obtained through averaging and by using $\hat{\theta}$ in place of θ_0, computation is usually highly efficient, and stability is a non-issue. Again, the quadratic approximation of some objective function about some local point of optimality is seen to be useful. Furthermore, Gaussian results are highly tractable, as a Gaussian empirical process often converge asymptotically to known and well-studied continuous-time stochastic processes. As such, using asymptotic normality emerged as the preferred solution in Paper II and III. Much more can be said about different asymptotic results in statistics, conditions under which normality emerge, and its applications. For an overview see van der Vaart (1998).

3.4 Model selection

Denote the true distribution of some random variable X as $G_X(x)$ with corresponding density $g_X(x)$. Further, let $p_X(x;\hat{\theta})$ denote the density with fitted parameters $\hat{\theta}$ used to model X, which can be seen as an approximation to g. Then, the Kullback-Leibler divergence (KLD) (Kullback and Leibler, 1951), denoted D is given by

$$D(g,p) := \int g_X(x)\log g_X(x)dx - \int g_X(x)\log p_X(x;\hat{\theta})dx. \qquad (3.12)$$

Since the first integral in (3.12) is constant w.r.t. different choices of models $p_X(x;\theta)$, only the negative remaining integral is relevant, and is commonly referred to as relative KLD. The negative log-likelihood objective of the optimization problem in (2.1) is a sample version of the relative KLD and appears as the natural objective for optimization over different values of θ, when the overarching goal is the minimization of KLD.

Selection between models is more difficult when candidate models are of different functional form and complexity. This becomes clear if we rewrite the fitted sample negative log-likelihood as the expectation with respect to the empirical empirical distribution G_X^*,

$$-n^{-1} \log p_X(\mathbf{x}; \hat{\theta}) = -\int \log p_X(x; \hat{\theta}) dG_X^*(x). \tag{3.13}$$

The empirical distribution G_X^* corresponds more closely towards fitted models $p_X(x; \hat{\theta})$ with higher complexity, than does true G_X (Konishi and Kitagawa, 1996). Therefore, naively using the negative log-likelihood as the basis for model selection will result in unfairly consistent choices of models with high complexity over parsimonious models. This is termed the "optimism" of the training loss (Hastie et al., 2001). This is taken into consideration in the supervised-learning optimization objective (2.2). If the loss function l appearing in (2.2) is a negative log-likelihood, then the objective of (2.2) is exactly equal to the relative KLD in (3.12), as evaluation is over data $(y^0, \mathbf{x}^0)$ unseen in the fitting of $\hat{\theta}$.

In the coming discussion, consider the loss-based regression setting $l(y, f(\mathbf{x}; \theta))$. The idea of generalization-based information criteria is to adjust for the bias induced by integrating the model w.r.t. the empirical distribution instead of the true distribution G_X in (3.13). Denote this bias or the optimism by $C(\hat{\theta})$, making the dependence upon fitted parameters $\hat{\theta}$ explicit. Then $C(\hat{\theta})$ is given by

$$C(\hat{\theta}) = E\left[l(y^0, f(\mathbf{x}^0; \hat{\theta}))\right] - E\left[l(y, f(\mathbf{x}; \hat{\theta}))\right], \tag{3.14}$$

where in the first expectation, $(y^0, \mathbf{x}^0)$ is independent of data using in fitting of $\hat{\theta}$, while in the second expectation $(y, \mathbf{x})$ is part of the training set. Information criteria like the celebrated Akaike Information Criterion (commonly known as AIC) (Akaike, 1974), Takeuchi Information Criterion (TIC) (Takeuchi, 1976) and Network Information Criterion (NIC) (Murata et al., 1994) targets this bias $C(\hat{\theta})$. A detailed development of AIC and TIC is found in Burnham and Anderson (2003), and the case for NIC is almost completely analogous.

Common for all three information criteria mentioned above, is that they rely on the asymptotic approximation

$$C(\hat{\theta}) \approx \mathtt{tr}\left(E\left[\nabla^2_\theta l(y, f(\mathbf{x};\theta_0))\right] Cov(\hat{\theta})\right), \tag{3.15}$$

which is developed from two quadratic approximations (3.1) of the loss l about θ_0 and $\hat{\theta}$. The approximation is applicable when the loss is appropriately differentiable in θ, and $\hat{\theta}$ is a consistent estimator. In the case of AIC, the true model g is assumed as an interior point in the space of θ. Under this assumption, the covariance in (3.15) is the inverse of the expected hessian. Thus, the right hand side of (3.15) reduces to the number of dimensions of θ, say p. If g is not assumed to be an interior point, the Sandwich-estimator due to Huber (Huber et al., 1967) can be used for the covariance. Then the r.h.s. of (3.15) becomes $C(\hat{\theta}) \approx \mathtt{tr}(J(\theta_0)^{-1} I(\theta_0))$, and estimation of J and I as discussed in Section 3.3 results in TIC and NIC.

Notice that (3.15) is not directly applicable to the optimism seen in Figure 3.3 for GTB models. This is because l is generally not differentiable in the different split points being profiled over to maximize reduction in loss. A fast (but valid) approximation of the optimism (3.14) associate gradient boosted trees is however tractable. From the Algorithms 1 and 2 it is seen that the functional complexity is constantly increasing for every split and every tree that is added to the model. The state-of-the-art implementations of GTB employs CV-based tuning of several hyperparameters to control this complexity. Development of a valid information criterion applicable to the splits of gradient boosted trees is the subject of Paper III.

Finally, note the existence of many other information criteria, that may seek to improve on the above-mentioned criteria, or that targets other objectives than relative KLD and expected generalization loss. Both the Corrected Akaike Information Criterion (also known as AICc) (Sugiura, 1978), and the Bayesian information criterion (known as BIC) (Schwarz et al., 1978) are well known. Interestingly, the BIC can be developed using a quadratic approximation together with a close cousin of the SPA

called the Laplace approximation. A different information criterion that does not target relative KLD for the overall fit of the model, but rather models for individual parameters, say the mean μ, is the Focused Information Criterion (FIC) (Claeskens and Hjort, 2003). Finally, also note the existence of the cross-validation Copula Information Criterion, developed from theoretical results in relation to n-fold CV (Grønneberg and Hjort, 2014). See Claeskens et al. (2008) for an overview of model selection.

4 Computational remarks

The previous chapters have focused on the theoretical basis for the papers I-III. A large part of the concluded research has been of a computational nature, and hence, certain key components of these computations deserve a mention. Also, reproducing the research without knowledge of these (or related) components may seem a Herculean task to the author.

For the most part, the theory of papers I-III has been implemented in the programming languages R (R Core Team, 2018) and C++ (Stroustrup, 2000). Typically, early exploration and treatment of data has been done in R, while the heavy lifting has been done by C++. Interplay between C++ and R is seamless due to the R package Rcpp (Eddelbuettel and François, 2011) and the amazing functionality of Rcpp modules. The R package TMB (Kristensen et al., 2016), similarly as Rcpp, allow for the evaluation of C++ functions in R, but also comes with the powerful tools of automatic differentiation, to the delight of users. On the C++ side, the research has made extensive use of the linear algebra library Eigen (Guennebaud et al., 2010), and the automatic differentiation library Adept (Hogan, 2014). The functionality of the R package ggplot2 (Wickham, 2016) in combination with the R packages tidyverse (Wickham, 2017) and data.table (Dowle and Srinivasan, 2019) has been taken advantage of to create figures for papers I-III.

5 Summary of the papers

The first paper of the thesis, "Saddlepoint adjusted inversion of characteristic functions", written in collaboration with the professors Tore Selland Kleppe and Hans Julius Skaug, develops the SPA through exponential tilting in the time-domain side of the Fourier transform. This development admits a deconstruction of the density of some random variable X as the SPA and the density of a standardized random variable Z evaluated at zero. The variable Z is specified through its CGF $K_Z(s)$, a function of the CGF of X. The representation of p_X is exact, and necessarily takes care of issues regarding renormalization and unimodality of the SPA. Furthermore, as evaluation of the density of Z is only necessary at the, by-design, high-density point zero, inversion using quadrature is accurate and does not suffer from the numerical issues illustrated in Figure 3.1. This is true also at low-density regions of X where the SPA typically will dominate. The methodology is illustrated using the Negative Inverse Gaussian distribution, and applied to financial data through the Merton Jump Diffusion model.

In the second paper, "An information criterion for automatic gradient tree boosting", written in collaboration with the professors Tore Selland Kleppe and Hans Julius Skaug, an information criterion is constructed that takes into consideration the optimism induced into the training loss by the greedy recursive binary splitting in Algorithm 2. The bias (3.14) is found by relating the asymptotic dynamics of the loss under split-profiling, to that of a Cox-Ingersoll-Ross process (CIR). The asymptotic normality (3.11) of estimators is key to establish a familiar continuous-time stochastic process, and eventually the CIR, that makes it possible to build upon the approximate equation (3.15) for generalization-loss based information criteria. Finally, to take into consideration the optimism induced by selecting the maximum reduction in loss over features j in Algorithm 2, extreme-value theory is employed. The criterion is built into an algorithm for gradient tree boosting which is automatic, removing hyperparameters such as the number of boosting iterations

K and constraints regularizing trees. The underlying assumptions are tested on simulated data, and the algorithm is validated on a collection of real data by measuring both speed and accuracy versus comparative methodologies.

The third paper, "agtboost: Adaptive and Automatic Gradient Tree Boosting Computations", written in collaboration with professor Tore Selland Kleppe, focuses on the implementation of the theory in Paper II in an R-package named agtboost. In addition, the paper proposes a modification of the splitting procedure in Algorithm 2, to be more adept to gradient boosting by taking into consideration the possible root-split produced in the coming boosting iteration. Usage of agtboost is illustrated, and its behaviour on the large Higgs dataset is measured. The agtboost package lowers the theoretical and practical threshold for users to employ gradient tree boosting, as detailed knowledge on hyperparameter tuning and setting up $k-$fold CV in code no longer is a necessity.

References

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control 19*(6), 716–723.

Barndorff-Nielsen, O. E. and C. Kluppelberg (1999). Tail exactness of multivariate saddlepoint approximations. *Scandinavian journal of statistics 26*(2), 253–264.

Breiman, L., J. Friedman, C. J. Stone, and R. A. Olshen (1984). *Classification and Regression Trees*. CRC Press.

Burnham, K. P. and D. R. Anderson (2003). *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Springer Science & Business Media.

Butler, R. W. (2007). *Saddlepoint approximations with applications*. Cambridge University Press.

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.

Claeskens, G. and N. L. Hjort (2003). The focused information criterion. *Journal of the American Statistical Association 98*(464), 900–916.

Claeskens, G., N. L. Hjort, et al. (2008). Model selection and model averaging. *Cambridge Books*.

Daniels, H. E. (1954). Saddlepoint approximations in statistics. *The Annals of Mathematical Statistics*, 631–650.

Dowle, M. and A. Srinivasan (2019). *data.table: Extension of 'data.frame'*. R package version 1.12.8.

Eddelbuettel, D. and R. François (2011). Rcpp: Seamless R and C++ integration. *Journal of Statistical Software 40*(8), 1–18.

Efron, B. (1992). Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics*, pp. 569–593. Springer.

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.

Griewank, A. and A. Walther (2008). *Evaluating derivatives: principles and techniques of algorithmic differentiation*. Siam.

Grønneberg, S. and N. L. Hjort (2014). The copula information criteria. *Scandinavian Journal of Statistics 41*(2), 436–459.

Guennebaud, G., B. Jacob, et al. (2010). Eigen v3. http://eigen.tuxfamily.org.

Hastie, T., R. Tibshirani, and J. Friedman (2001). *The Elements of Statistical Learning*. Springer Series in Statistics New York, NY, USA:.

Hogan, R. J. (2014). Fast reverse-mode automatic differentiation using expression templates in C++. *ACM Transactions on Mathematical Software (TOMS) 40*(4), 26.

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.

Ishiguro, M., Y. Sakamoto, and G. Kitagawa (1997). Bootstrapping log likelihood and eic, an extension of aic. *Annals of the Institute of Statistical Mathematics 49*(3), 411–434.

Konishi, S. and G. Kitagawa (1996). Generalised information criteria in model selection. *Biometrika 83*(4), 875–890.

Kristensen, K., A. Nielsen, C. W. Berg, H. Skaug, and B. M. Bell (2016). TMB: Automatic differentiation and Laplace approximation. *Journal of Statistical Software 70*(i05).

Kullback, S. and R. A. Leibler (1951). On information and sufficiency. *The annals of mathematical statistics 22*(1), 79–86.

Mason, L., J. Baxter, P. Bartlett, and M. Frean (1999). Boosting algorithms as gradient descent in function space (technical report). *RSISE, Australian National University*.

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.

R Core Team (2018). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.

Schwarz, G. et al. (1978). Estimating the dimension of a model. *The annals of statistics 6*(2), 461–464.

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

Stroustrup, B. (2000). *The C++ programming language*. Pearson Education India.

Sugiura, N. (1978). Further analysts of the data by akaike's information criterion and the finite corrections: Further analysts of the data by akaike's. *Communications in Statistics-Theory and Methods 7*(1), 13–26.

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.

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.

Wickham, H. (2017). *tidyverse: Easily Install and Load the 'Tidyverse'*. R package version 1.2.1.

Appendix

Saddlepoint-adjusted inversion of characteristic functions.....31

An information criterion for automatic gradient tree boosting 53

agtboost: Adaptive and Automatic Gradient Tree Boosting Computations....96

Paper I

Saddlepoint-adjusted inversion of characteristic functions

Saddlepoint-adjusted inversion of characteristic functions

Berent Å. S. Lunde
Department of Mathematics and Physics, University of Stavanger
and
Tore S. Kleppe
Department of Mathematics and Physics, University of Stavanger
and
Hans J. Skaug
Department of Mathematics, University of Bergen

August 19, 2020

Abstract

For certain types of statistical models, the characteristic function (Fourier transform) is available in closed form, whereas the probability density function has an intractable form, typically as an infinite sum of probability weighted densities. Important such examples include solutions of stochastic differential equations with jumps, the Tweedie model, and Poisson mixture models. We propose a novel and general numerical method for retrieving the probability density function from the characteristic function, conditioned on the existence of the moment generating function. Unlike methods based on direct application of quadrature to the inverse Fourier transform, the proposed method allows accurate evaluation of the log-probability density function arbitrarily far out in the tail. Moreover, unlike ordinary saddlepoint approximations, the proposed methodology is in principle exact modulus discretization and truncation error of quadrature applied to inversion in a high-density region. Owing to these properties, the proposed method is computationally stable and very accurate under log-likelihood optimisation. The method is illustrated for a normal variance-mean mixture, and in an application of maximum likelihood estimation to a jump diffusion model for financial data.

Keywords: Likelihood estimation, Inverse Fourier transform, Exact saddlepoint approximation, Numerical integration, Jump diffusions, Mixture models

1 Introduction

This paper is motivated by the problem of making inference about parameters of statistical models, in which the probability density function is not readily available, but the characteristic function (CF) is. Such models appear frequently when there are more than one source of randomness, which is usually the case in the real world;

(1) The compounded Poisson process and Tweedie model (Tweedie, 1984; Jorgensen, 1987), has a closed form characteristic function, which is usually applied when doing inference.

(2) The transition distributions of the class of affine jump diffusion models (Duffie et al., 2000) admit characteristic functions given in terms of the solution of certain ordinary differential equations, whereas the transition densities are typically not tractable.

(3) The solution of non-linear stochastic differential equations (SDE) can be approximated well using Itô-Taylor expansions, for which the characteristic function can be derived (Preston and Wood, 2012). A further extension to the SDE is to add independent stochastic jumps that occur with an intensity at the given time. The new sources of randomness, the counting process and the jump size, can then be easily added to the characteristic function of the approximate solution to the SDE (Zhang and Schmidt, 2016).

(4) Other important examples include the non-central χ^2, other Poisson mixtures, Normal mixtures and so on.

Estimation methods based on the CF (or more generally, when the density/mass function is unavailable) can roughly be classified in two groups, based on whether or not the inverse Fourier transform (IFT) is used to compute likelihoods. Popular methods in the latter group include (Generalized) Method of Moments (MM), which seeks to match a finite number of empirical moments with those from the model (see e.g. Hansen, 1982, and subsequent literature). A somewhat similar approach is that of using the empirical characteristic function (ECF), where the idea is to minimize a measure of the difference between the model-implied and empirical characteristic functions (see e.g. Yu, 2004, and references therein).

The MM is computationally very efficient, but as only a finite number of moment conditions are considered, these methods may discard important information from the data. Such information

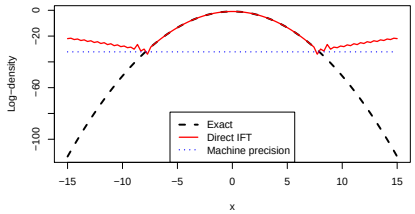

Figure 1: Logarithm of the $N(0,1)$ density calculated using quadrature-based direct inversion (Direct IFT), along with the exact log-density. Also indicated is $\log(1.0 \times 10^{-14})$.

loss does not in theory occur for ECF, but in practice ECF estimators often turn out to have asymptotic inefficiencies (see e.g. Knight et al. (2002, Section 4)).

The former group of methods, e.g. fully likelihood-based methods such as maximum likelihood estimation and Bayesian methods, approximates densities by evaluating IFTs numerically. Direct integration techniques, such as quadrature, will, as illustrated in Figure 1, suffer from numerical inaccuracies that are non-trivial on log-scale in low-density regions. Such problems typically occur when the value of the true density is smaller than around 1.0×10^{-14}, when using double numerics. This is related to the fact that the integrand of the IFT integral takes values $O(1)$. Consequently the weighted sum of integrand evaluations constituting the quadrature approximation will be represented by a floating point number with integer exponent close to 0 (see e.g. Press et al., 1992, Section 1.3, for a discussion of how floating point numbers are represented). The spacing between representable numbers in such a floating point representation (typically on the order 1.0×10^{-16} for exponent close to 0) produces a theoretical lower bound on the magnitude of density values that can be accurately represented. However, from experience, accurate calculation of log-densities via quadrature approximations to the IFT typically fails when the density takes values a few orders of magnitude higher than this bound.

For likelihood-based estimation purposes, it is typically very important that log-densities can be evaluated in a stable manner, even far out in the tails. Therefore, density retrieval for estimation purposes will often involve rescaling properties of the density, such that the IFT is done in a high-density region and then scaled to the point of evaluation. For example, for the mentioned

Tweedie model, Dunn and Smyth (2008) elaborate on density retrieval by the IFT using numerical integration. To bypass the problems in the tail, an analytically tractable rescaling property is stated and applied. Moreover, it is commented that satisfactory accuracy is only attained by utilizing this property. Unfortunately, such rescaling properties are not always evident for a general model.

An alternative to direct inversion is the saddlepoint approximation (SPA) (Daniels, 1954; Barndorff-Nielsen and Cox, 1979; Butler, 2007). The SPA often admits fast computation, is highly automatic and allows for well-scaled numerics when evaluating log-density approximations. In addition, the SPA has been shown to be accurate in the tails of the distribution (see Butler, 2007). However, the SPA suffers some drawbacks in the context of likelihood estimation. Most notably is that it in general does not integrate to 1 for non-Gaussian models, with values often < 1 and thus, as a given parameter space may hold points that give Gaussian models where the SPA *is* exact, may bias SPA-based estimation towards regions of the parameter space where the integral over the SPA is highest. Further, the SPA is often unimodal, and may thus produce an inaccurate approximation of e.g. mixtures, which is a typical case in which the characteristic function is readily available but the density is not.

To bypass some of these problems, the SPA can be improved upon by employing a non-Gaussian base (Wood et al., 1993). Ait-Sahalia and Yu (2006) considers this approach for Markov processes, including jump diffusions. However, non-Gaussian based SPA requires the user to select a base-distribution with a density that in some sense is similar to the target density, and therefore may require bespoke implementations in each instance. Kleppe and Skaug (2008) introduce a method for choosing this base-distribution automatically for the purpose of high-dimensional inversion under a latent-variable framework.

In this paper, we propose to precondition the integrand in univariate inversion problems so that quadrature-based inversion is as numerically stable as possible. This is done by inverting a standardised and exponentially tilted (Barndorff-Nielsen and Cox, 1979) version of the target density, so that the inversion is done in a high-density region. The method is in large parts automatic, and in particular does not rely on known rescaling properties of the target density or the elicitation of a non-Gaussian base distribution. Tilting is equivalent to deforming the contour of the inversion integral, and is a known technique in the applied mathematics literature for this type of problem (Cohen, 2007), so is rescaling. Our approach is similar to that taken

of Strawderman (2004), which develops an automatic algorithm for inversion to the cumulative density function, albeit the saddlepoint is different than the saddlepoint employed as the tilt in this paper. Strawderman (2004) gives much attention to the numerical evaluation of the resulting inversion integral, and these considerations apply equally to the method we propose.

The rest of the paper is laid out as follows: the proposed method and its implementation are outlined in section 2. In section 3, the proposed methodology is illustrated and contrasted to alternatives using the normal inverse Gaussian distribution and the Merton jump diffusion models as example models. Conclusion and comments follow in section 4.

2 Methodology

This section provides some background and subsequently derives the saddlepoint adjusted IFT method proposed here.

2.1 Background

Suppose that the strictly continuous random variable $X \in \mathbb{R}$ has probability density function $p_X(x)$ and CF $\varphi_X(s) = E_X(\exp(isX))$, where $i = \sqrt{-1}$. As the CF may be considered as a Fourier transform of $p_X(x)$, the density can be recovered from $\varphi_X(s)$ via an IFT:

$$p_X(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \varphi_X(s) e^{-isx} ds. \tag{1}$$

For numerical evaluation, (1) can be simplified by utilizing the Hermitian property of $\varphi_X(s)$:

$$p_X(x) = \frac{1}{\pi} \int_{0}^{\infty} \mathrm{Re}\left[\varphi_X(s) e^{-isx}\right] ds. \tag{2}$$

As explained in the introduction and illustrated in Figure 1, computing the log-density $\log p_X(x)$ by taking the logarithm of the output of a quadrature method applied to either (1) or (2) is problematic when the value of $p_X(x)$ is small. Indeed, from Figure 1 it is seen that evaluation of the $N(0,1)$ log-density based on direct inversion fails around $p_X = \log(1.0 \times 10^{-14})$ (indicated by the horizontal line).

2.2 Saddlepoint adjusted IFT

In order to avoid such numerical problems when evaluating log-densities, this section introduces a general method of preconditioning the integration problem. The overarching idea is to ensure that every numerical IFT approximates the density of a unit variance random variable at its mean. Modulus strongly pathological cases, this approach should ensure that the value of the numerical Fourier transform is $O(1)$.

Again, suppose X is the random variable of interest. Provided it exits, the cumulant generating function (CGF) $K_X(t)$ is defined as $K_X(t) = \log\{E(\exp(tX))\}$, for values of $t \in \Omega$ where $\Omega = \{t : E(\exp(tX)) < \infty\}$. In particular, the CGF may be recovered from the CF via

$$K_X(t) = \log\{\varphi_X(-it)\}.$$

Suppose one wishes to evaluate p_X at some point, say x_0. A general and analytically tractable rescaling of the original density p_X is obtained by first introducing an exponentially tilted (see e.g. Butler, 2007, Section 2.4.5) version of X, say $X(\tau)$, where $\tau \in \Omega$ is the tilt parameter. The tilted random variable $X(\tau)$ has density

$$p_{X(\tau)}(x) = \exp(\tau x - K_X(\tau))p_X(x). \tag{3}$$

Notice in particular that the original random variable X is obtained for $\tau = 0$. Straightforward calculations yield that the CGF associated with $X(\tau)$ is given as

$$K_{X(\tau)}(t) = K_X(t + \tau) - K_X(\tau). \tag{4}$$

Now, in order for the evaluation of $p_{X(\tau)}$ to happen in a high-density region, the tilt parameter $\tau = \hat{\tau}$ is chosen so that $E(X(\hat{\tau})) = x_0$. As $E(X(\tau)) = K'_X(\tau)$, one obtains the saddlepoint equation

$$K'_X(\hat{\tau}) = x_0, \tag{5}$$

for $\hat{\tau}$, where K'_X denotes the derivative of K_X.

A further standardization is then introduced in order to ensure that the target for the numerical

IFT also has unit variance (and thus under most circumstances has density values $O(1)$ around the mean). This is achieved by introducing a standardized version of $X(\hat{\tau})$,

$$\bar{X}(\hat{\tau}) = \frac{X(\hat{\tau}) - x_0}{\sqrt{Var(X(\hat{\tau}))}}, \tag{6}$$

where $Var(X(\hat{\tau})) = K''_X(\hat{\tau})$, so that

$$p_{X(\hat{\tau})}(x_0) = \frac{p_{\bar{X}(\hat{\tau})}(0)}{\sqrt{K''_X(\hat{\tau})}}. \tag{7}$$

Combining (3) and (7) results in the preconditioning formula used throughout this text, namely

$$p_X(x_0) = \exp(K_X(\hat{\tau}) - \hat{\tau}x_0)\frac{p_{\bar{X}(\hat{\tau})}(0)}{\sqrt{K''_X(\hat{\tau})}}. \tag{8}$$

Since (6) represents an affine transformation of $X(\hat{\tau})$, the CF and CGF of $\bar{X}(\hat{\tau})$ are also easily found to be

$$\begin{aligned} \varphi_{\bar{X}(\hat{\tau})}(s) &= \exp(K_{\bar{X}(\hat{\tau})}(is)), \\ K_{\bar{X}(\hat{\tau})}(t) &= -K_X(\hat{\tau}) - \frac{tx_0}{\sqrt{K''_X(\hat{\tau})}} \\ &\quad + K_X\left(\frac{t}{\sqrt{K''_X(\hat{\tau})}} + \hat{\tau}\right), \end{aligned}$$

and thus density of $\bar{X}(\hat{\tau})$ evaluated at zero can be calculated as

$$p_{\bar{X}(\hat{\tau})}(0) = \frac{1}{\pi}\int_0^\infty \mathrm{Re}[\varphi_{\bar{X}(\hat{\tau})}(s)]ds. \tag{9}$$

Before proceeding, notice that the conventional SPA is obtained by substituting $p_{\bar{X}(\hat{\tau})}(0)$ in (8) with the $N(0,1)$ density evaluated at 0, namely $(2\pi)^{-1/2}$. Thus, it follows from results on tail-exactness of the SPA for densities that are log-concave (Barndorff-Nielsen and Kluppelberg, 1999) that also $p_{\bar{X}(\hat{\tau})}(0)$ must converge to $(2\pi)^{-1/2}$ in the tails of p_X. In the high-density regions of p_X on the other hand, $p_{\bar{X}(\hat{\tau})}(0)$ typically takes values somewhat higher than $(2\pi)^{-1/2}$. These properties are illustrated for a normal inverse Gaussian distribution (to be discussed in more detail shortly)

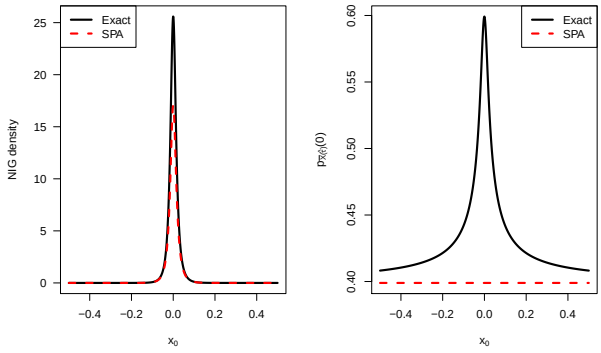

Figure 2: The normal inverse Gaussian (NIG) density along with its SPA (left panel). The right panel shows $p_{\bar{X}(\hat{\tau})}(0)$ (Exact) used in the proposed methodology, together with the asymptote $(2\pi)^{-1/2}$ (SPA). The parameter values $\chi = 0.0003$, $\psi = 1000$, $\mu = -0.0003$, and $\gamma = 2$ in (10) were applied in both plots.

in Figure 2. Note that $p_{\bar{X}(\hat{\tau})}(0)$ approaches $(2\pi)^{-1/2}$ in the tails of the distribution, suggesting that the SPA has the tail exactness property for the normal inverse Gaussian distribution. Moreover, notice that $p_{\bar{X}(\hat{\tau})}(0)$ remains well scaled $O(1)$ across the support of the density, and is therefore easy to approximate using quadrature.

2.3 Implementation

Provided the CGF/CF of some distribution X, equations (8) and (9) form the basis for implementing the proposed saddlepoint adjusted IFT technique for evaluating (log-)densities at say x_0. Each evaluation involves the following steps:

1. Obtain the saddlepoint $\hat{\tau}$ by solving (5). Notice that $\hat{\tau} = \arg\min_{t\in\Omega} K_X(t) - x_0 t$, where the objective function is convex on Ω. The convexity ensures both a unique such solution to (5), and also that Newton's method of optimization (Press et al., 1992, Chapter 9.4) may be used to obtain a rapidly- and stably converging solution.

2. Approximate the latter integral of (9) by a quadrature approximation, say $\hat{p}_{\bar{X}(\hat{\tau})}(0)$.

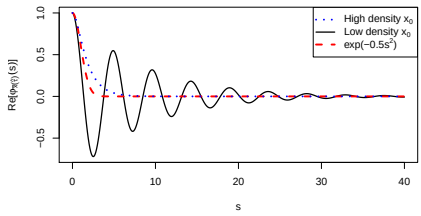

Figure 3: Integrands of the latter representation of (9) for the normal inverse Gaussian distribution. The parameters are the same as in Figure 2. High density x_0 correspond to $x_0 = E(X)$ and low density to $x_0 = E(X) - 8\sqrt{Var(X)}$. As a reference, also the the CF associated with the $\bar{X}(\hat{\tau})$ obtained whenever X is Gaussian, i.e. $\exp(-s^2/2)$ is indicated. It is seen that the integrands typically take non-negligible values for larger values of s for relative to the Gaussian case.

3. Compute log-density approximation as

$$\log(p_X(x_0)) \approx K_X(\hat{\tau}) - \hat{\tau}x_0 - \frac{1}{2}\log\left(K_X''(\hat{\tau})\right) + \log(\hat{p}_{\bar{X}(\hat{\tau})}(0)).$$

The saddlepoint adjusted IFT comes with some extra cost compared to the saddlepiont approximation and, depending on number of quadrature evaluation, potentially direct IFT. However, for the normal-inverse Gaussian model considered shortly, the location of a single saddlepoint $\hat{\tau}$ per evaluation is a minor part of the required CPU time. This is typically the case even if the saddlepoint equation must be solved numerically, as is often the case for non-trivial models. Thus, of highest importance for good and robust performance is the selection of a quadrature rule for implementing point 2 above.

Figure 3 displays integrands $\mathrm{Re}[\varphi_{\overline{X}(\hat{\tau})}(s)]$ for the normal inverse Gaussian distribution also considered in Figure 2. For values of x_0 in the high-density region of X, the integrand falls to zero rapidly, and the resulting integral may be accurately approximated using Gauss-Hermite quadrature (Press et al., 1992, p. 153). On the other hand, is seen that the integrand may take non-trivial values far from the origin when x_0 is in the tails of X (even if the resulting integral attains values close to $(2\pi)^{-1/2}$ (Barndorff-Nielsen and Kluppelberg, 1999)).

In the present work, composite Simpson's quadrature with a fixed integration range is used for the integration problem in point 2 above. This choice is made mainly for robustness and in order to obtain (log-)density approximations that are smooth functions in the parameters. This works fine for the selected illustrations, but more heavy-tailed distributions that have a slower convergence of $\lim_{s\to\infty} \mathrm{Re}[\varphi_{\overline{X}(\hat{\tau})}(s)] \to 0$, might severely suffer from truncation error. Abate and Whitt (1992) argues for the even simpler Trapezoid method for inversion problems, which' relation to the Poison summation formula may be used to control discretization error, in combination with Euler-summation for convergence acceleration which targets truncation error. Strawderman (2004) extensively discuss the issues of discretization and truncation error, and suggests Wynn's epsilon-method for convergence acceleration over Euler-summation. A more adaptive selection of integration range that is also smooth in parameters, type of convergence acceleration, and also choosing integration rules that account for the potentially oscillating nature of the integrand holds scope for future research. Notice, however, that such adaptive integration would not alone solve the problems with log-density evaluation for direct IFT as illustrated in Figure 1.

Notice, for comparison, that renormalised (via numerical integration) SPAs require the solution of many saddlepoint equations, while at the same time, introduce non-vanishing approximation errors and loses tail exactness. Thus, due both to a higher computational cost and non-vanishing errors, renormalised SPAs are not considered further here.

Throughout this paper, all methods are implemented in C++ and run in R (R Core Team, 2018) using the RCPP package (Eddelbuettel and François, 2011). Exact gradients of log-likelihood functions were obtained using the automatic differentiation (AD) library Adept (Hogan, 2014). All derivatives of the CGF are hand-coded in the present work, but this process may also be automated using a tool that allows for nested computation of derivatives such as TMB (Kristensen et al., 2016).

3 Illustrations

The focus of this section is to highlight several properties of the saddlepoint adjusted IFT method, and to contrast these to the SPA and direct IFT. The included target distributions hold Gaussian solutions as special cases in their respective parameter space, and thus in some part selected for their illustrative purposes with regards to the Gaussian bias in SPA based estimation.

3.1 The Normal inverse Gaussian distribution

Throughout this section, the normal-inverse Gaussian (NIG) (see e.g. McNeil et al., 2005) is used as a test case since this distribution admit exact evaluation of the density. The NIG distribution is defined as a normal-variance mixture,

$$X = \mu + \gamma W + \sqrt{W}Z,\ Z \sim N(0,1), \tag{10}$$

where W has an inverse Gaussian distribution with a parametrization corresponding to

$$\begin{aligned} p_W(w) &\propto w^{-3/2}\exp(-(\chi/w + \psi w)/2), \\ E(W) = \sqrt{\chi/\psi} \text{ and } Var(W) &= (\sqrt{\chi/\psi})/\psi. \end{aligned}$$

The marginal density of X is expressible only in terms of the modified Bessel function of the third kind K_λ:

$$\begin{aligned} p_X(x) = &\frac{\sqrt{\chi(\psi+\gamma^2)}K_{-1}\left(\sqrt{(\chi+(x-\mu)^2)(\psi+\gamma^2)}\right)}{\pi\sqrt{\chi+(x-\mu)^2}} \\ &\times e^{\sqrt{\chi\psi}+(x-\mu)\gamma}. \end{aligned}$$

On the other hand, the CGF is highly tractable:

$$K_X(t) = s\mu + \sqrt{\chi}\left(\sqrt{\psi} - \sqrt{\psi - s^2 - 2s\gamma}\right).$$

In particular, based on the CGF, one obtains $E(X) = \mu + \gamma\sqrt{\chi/\psi}$ and $Var(X) = \sqrt{\chi}\psi^{-3/2}(\gamma^2 - \psi)$.

In order to implement saddlepoint adjusted IFT, a quadrature scheme must be chosen to approximate $p_{\bar{X}(\tilde{\tau})}(0)$ as given in the latter representation of (9). Here, Simpson's quadrature based on 512 equidistant evaluations in the interval $s \in [0, 100]$ were used throughout. The interval and number of evaluations were chosen to be highly robust for a wide range of parameters and evaluation points x_0. Notice, that direct IFT on the other hand requires more manual tuning depending on parametrisation.

To illustrate how the saddlepoint adjusted IFT performs under log-likelihood-based estimation,

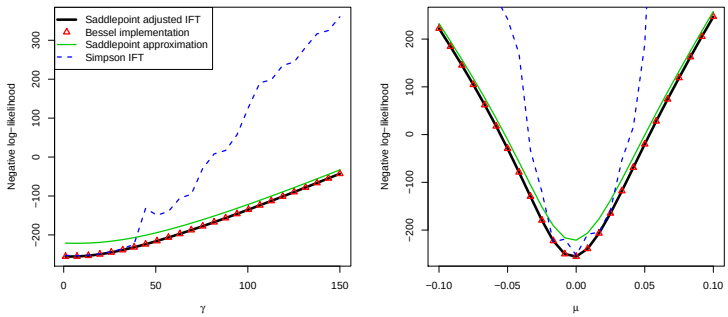

Figure 4: Approximate (negative) log-likelihood profiles based on 100 observations simulated from a NIG distribution with parameter $\chi = 0.0003$, $\psi = 1000$, $\mu = -0.0003$, and $\gamma = 2$. The left panel shows log-likelihoods plotted as a function of γ while keeping the remaining parameters fixed at their "true" values. The right panel shows a similar plot with varying μ.

100 observations from a NIG distribution with parameters $\chi = 0.0003$, $\psi = 1000$, $\mu = -0.0003$, and $\gamma = 2$ were simulated. These parameter values correspond to $E(X) \approx 0.0008$ and $Var(X) \approx 0.023^2$, which are typical of financial returns. The log-likelihood, approximated by saddlepoint adjusted IFT, the SPA, the direct IFT, and using the native R Bessel implementation (regarded as being exact), are plotted in Figure 4 as functions of γ and μ, respectively, while keeping the remaining parameters fixed at the values used for simulation. In both cases it is seen that the direct IFT based on Simpson's method produces unreliable results for parameter settings far from the "true" parameters. Moreover, the SPA produces a log-likelihood approximation which deviates from the true log-likelihood (Bessel implementation) with a parameter-dependent amount. Hence, the SPA based log-likelihood approximation is likely to result in biases relative to the exact maximum likelihood estimator. Finally, it is seen that the saddlepoint adjusted IFT based approximation is indistinguishable from the true log-likelihood.

To avoid numerical problems resulting from non-positive density approximations, the direct IFT log-likelihood was implemented as $\log(\max(1.0e-14, \hat{I}))$ where $\hat{I}$ is the quadrature-based approximation of (2). Such non-positive density approximations are obtained as a consequence of a highly oscillating integrand that takes values far larger than the value of the integral itself. Simpson's quadrature, with 512 function evaluations over the integration range $s \in [0, 150]$, was

used. Nevertheless, Figure 4 shows that direct IFT produces erratic behaviour for parameters far from those used in the simulation. When increasing γ (left panel) the distribution of X changes from being rather symmetric around $\gamma = 2$ to being highly skewed with a heavy right tail and a very thin left tail around $\gamma = 150$. Thus, in this latter case, some of the simulated observations will have very small density values, which is problematic for direct IFT, as demonstrated in Figure 1. Similar reasoning holds for the right panel, where the location parameter μ is is varied by around 4 standard deviations in either direction from the "true" μ, and in this case the majority of the simulated data are in the far tails at either extreme of the plot.

As is also seen in Figure 4, the SPA produces approximate log-likelihoods that deviate by a parameter-dependent offset from the exact log-likelihood. This behaviour is related to the fact that the SPA typically does not integrate to 1 (typically < 1), and that the appropriate normalisation factor of the SPA is parameter dependent. However, the SPA is exact in the Gaussian case, and it is therefore often seen (see e.g. Kleppe and Skaug, 2008) that SPA-based parameter estimates are biased towards the "Gaussian part" of the parameter space in models that nest Gaussian distributions. For the NIG distribution, a $N(\mu + \gamma\sigma^2, \sigma^2)$ distribution obtains when $E(W) \to \sigma^2$ and $Var(W) \to 0$ (e.g. when $\chi = \sigma^4\psi$ and $\psi \to \infty$).

To illustrate this effect, we fixed parameters $\mu = \gamma = 0$ and chose $\chi = \psi$ so that $\mathbb{E}[W] = 1$. The remaining free parameter $\theta = 1/\psi = Var(W)$ controls the variance of W, and thus the deviation from normality, with $X \to N(0, 1)$ as $\theta \to 0$. We then computed estimators of θ by numerically minimizing the relative Kullback-Leibler divergence to obtain asymptotic maximum likelihood estimators,

$$\hat{\theta}(\theta_0) = \arg\min_{\theta} \left\{ - \int \log\{\tilde{p}_X(x;\theta)\} p_X(x;\theta_0) dx \right\}, \tag{11}$$

for different settings of the inverse Gaussian variance θ_0. Here, $\tilde{p}$ is either the SPA or the saddlepoint adjusted IFT. The integral in the objective function of (11) was resolved using quadrature with 200 evaluations on an interval centered at the mean and spanning 12 standard deviations. The results showed that for the SPA we have $\hat{\theta}(\theta_0) = 0$, i.e. a Gaussian distribution, for all values of θ_0. For the saddlepoint adjusted IFT, on the other hand, $\hat{\theta}(\theta_0)$ is indistinguishable from θ_0, i.e. the estimator is asymptotically unbiased.

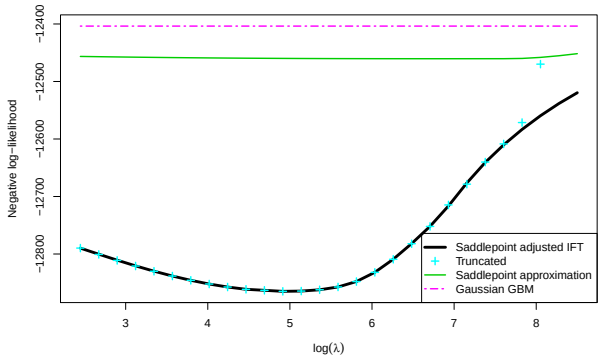

Figure 5: Profile negative log-likelihoods of the Merton Jump Diffusion model versus fixed logarithmic jump-intensities, λ, fitted to DJIA daily stock-market data from 01.01.2000 until 01.01.2018. The evaluations referred to as "Truncated" are obtained by summing conditional probabilities in (15), until either the jump probability is less than 1.0×10^{-14} or reaching 20 jumps. Saddlepoint adjusted IFT uses Simpson's quadrature with 128 equidistant steps over $s \in [0, 16]$. Direct inversion was deemed infeasible, as it severely suffers from the numerical problems discussed in this article.

3.2 Application to real data: Merton jump diffusion

This section considers the application of inversion techniques to likelihood optimisation based on real data. Specifically, the Merton jump diffusion (MJD) (Merton, 1976) for stock prices is considered. Under the MJD model, the dynamics of a stock price S_t are described by the jump diffusion model

$$\frac{dS_t}{S_{t^-}} = (r - \lambda k)dt + \sigma dW_t + (Y_t - 1)dN_t, \tag{12}$$

where $r > 0$ is the instantaneous expected return on the asset, $\sigma > 0$ the instantaneous volatility (if a jump does not occur), and k the expected relative jump size. N_t denotes a Poisson process with intensity $\lambda > 0$, independent of the Brownian motion W_t. Y_t is the price jump size, meaning that, if a jump occurs at time t, the price jumps from S_{t^-} to $Y_t S_{t^-}$. The jump sizes Y_t are assumed to be log-normally distributed with $\log Y_t \sim N(\mu, \nu^2)$, and thus $k = e^{\mu + 0.5\nu^2} - 1$.

	Methods			
	SPI	Truncated	SPA	GBM
r	0.0445 (0.045)	0.0445 (0.045)	0.0432 (0.043)	0.0461 (0.047)
$\log \sigma$	-2.41 (0.077)	-2.41 (0.076)	-2.23 (1.11)	-1.67 (0.011)
$\log \lambda$	4.96 (0.18)	4.96 (0.18)	6.81 (2.36)	
μ	-0.00114 (0.00039)	-0.00114 (0.00039)	-0.000703 (0.00065)	
$\log \nu$	-4.32 (0.074)	-4.32 (0.074)	-5.4 (0.51)	

Table 1: Likelihood estimates and standard deviations in parentheses for the MJD fitted to the DJIA daily stock-market data from 2000 until 2018, applying different methods for likelihood approximation. SPI refers to saddlepoint adjusted IFT. The settings for the methods are the same as those described in Figure 5. The inversion methods were implemented using first order AD data-types with Adept (Hogan, 2014), and Hessian matrices could therefore be retrieved as finite-difference Jacobians to the exact gradients; these could in turn be used to calculate standard deviations (see for instance Kristensen et al. (2016) for details on such computations). The Hessian of the truncation method was retrieved via the `optim` function in R (R Core Team, 2018).

The SDE (12) can be solved to yield the following representation for logarithmic prices:

$$X_t = X_0 + \left(r - \lambda k - \frac{\sigma^2}{2}\right) t + \sigma W_t + \sum_{i=1}^{N_t} \log Y_i, \tag{13}$$

and thus the conditional CGF of $X_t|X_0$ is given as a sum of a normal CGF and a normal compounded Poisson CGF:

$$\begin{aligned} K_{X_t|X_0}(s) =& sX_0 + st\left(r - \lambda k - \frac{\sigma^2}{2}\right) + \frac{s^2\sigma^2 t}{2} \\ &+ \lambda t\left(e^{s\mu + \frac{\nu^2 s^2}{2}} - 1\right). \end{aligned} \tag{14}$$

The conditional CGF (14) does not appear to admit a closed form saddlepoint, $\hat{\tau}$, and thus numerical solution of (5) is required.

Notice that the Geometric Brownian Motion (GBM) processes for S_t obtains as special cases either for $\lambda = 0$, or when $\mu = 0$, $\lambda \to \infty$ and ν decays as $O(\lambda^{-1/2})$ or faster. Furthermore, $X_t|X_0$ is Gaussian under the GBM model.

Note also that for the MJD, $X_t|(X_0, N_t)$ is Gaussian, and thus the exact transition probability

density is available as a Poisson mixture with Gaussian components:

$$P(X_t|X_0) = \sum_{i=0}^{\infty} P(N_t = i)P(X_t|X_0, N_t = i). \tag{15}$$

Specifically, $X_t|(X_0, N_t)$ is Gaussian with mean $X_0 + t\left(r - \lambda k - \frac{\sigma^2}{2}\right) + N_t\mu$ and variance $\sigma^2 t + N_t\nu^2$. As a reference for the saddlepoint adjusted IFT, we also consider an approximate log-likelihood function (referred to as "truncated"), based on truncating the Poisson mixture infinite sum representation either when the Poisson weights become smaller than 1.0e-14 or at 20 jumps per transition.

The MJD was applied to stock-market data from the Dow Jones Industrial Average index from 01.01.2000 until 01.01.2018. A yearly time scale, and thus observations separated in time with $t = 1/252$, was used. Maximum likelihood estimation based on saddlepoint adjusted IFT (using Simpson's quadrature with 128 equidistant evaluations over $s \in [0, 16]$), SPA and the truncated method were considered. Direct IFT was deemed infeasible due to similar problems as discussed above.

Table 1 provides maximum likelihood estimates. First of all, it is seen that the parameter estimates obtained using saddlepoint adjusted IFT and the truncated method are close to indistinguishable. This observation suggests that the proposed methodology performs very well in also this situation, even when using a static integration rule. The parameter estimates suggest rather frequent jumps at a rate of around 0.6 per day. The jumps have close to zero mean and a standard deviation that is roughly double that of the diffusive part (i.e. $\sigma\sqrt{t}$).

Also included in Table 1 are parameter estimates obtained using a SPA-based log-likelihood approximation. This approximation favours a model with smaller and more frequent jumps, which as discussed above, suggests a model with closer to Gaussian transition distributions. Still, the model obtained using SPA-based log-likelihood is well separated from the exactly Gaussian GBM.

Figure 5 presents negative profile log-likelihoods over $\log(\lambda)$ based on the different considered methods. It is seen that moderate jump intensities, (say below $\log(\lambda) < 7$, $P(N_t = 20|\log(\lambda) = 7) = 3.1e{-}8$), the truncated and saddlepoint adjusted IFT profile log-likelihoods coincide very well, whereas for higher jump intensities, they diverge as the truncation of jump counts starts taking effect. Though not particularly relevant for the data and model at hand here, these observations

illustrate the utility of performing the mixing in transform-space, as the computational complexity remains the same for any jump intensity, whereas summing many jumps in the Gaussian mixture representation may be become very computationally expensive.

Figure 5 also includes profile likelihoods for SPA-based approximate log-likelihood and the GBM model (invariant of λ). It is seen that the approximate log-likelihood associated with the SPA is substantially closer to the GBM. Notice also that for high jump intensities, the saddlepoint adjusted IFT and SPA approach the GBM as close to Gaussian models are obtained for very high jump intensities are imposed. The truncated method, on the other hand, fails in representing this effect for high jump intensities.

4 Discussion

This paper proposes a new method for numerical inversion of characteristic functions, conditioned on the existence of the CGF. The proposed method is very reliable for obtaining log-density values, even far out in the tails of the distribution. In particular, the method resolves numerical problems that may occur using direct inverse Fourier transformation. Moreover, the method may be seen as a way of substantially improving the accuracy of the classical saddlepoint approximation when applied in likelihood-based inference.

Further work will involve more automatic rules for choosing integration ranges, under the constraint of producing smooth (in parameters) log-likelihood functions. A further extension would be to consider the low- but multi-dimensional analogue of the proposed methodology. However, more work regarding how to implement the resulting multidimensional integral in point 2 in section 2.3 must be carried out also in this case.

References

Abate, J. and W. Whitt (1992). The fourier-series method for inverting transforms of probability distributions. *Queueing systems 10*(1-2), 5–87.

Ait-Sahalia, Y. and J. Yu (2006). Saddlepoint approximations for continuous-time Markov processes. *Journal of Econometrics 134*(2), 507–551.

Barndorff-Nielsen, O. and D. R. Cox (1979). Edgeworth and saddle-point approximations with statistical applications. *Journal of the Royal Statistical Society. Series B (Methodological)*, 279–312.

Barndorff-Nielsen, O. E. and C. Kluppelberg (1999). Tail exactness of multivariate saddlepoint approximations. *Scandinavian journal of statistics 26*(2), 253–264.

Butler, R. W. (2007). *Saddlepoint approximations with applications*. Cambridge University Press.

Cohen, A. M. (2007). *Numerical methods for Laplace transform inversion*, Volume 5. Springer Science & Business Media.

Daniels, H. E. (1954). Saddlepoint approximations in statistics. *The Annals of Mathematical Statistics*, 631–650.

Duffie, D., J. Pan, and K. Singleton (2000). Transform analysis and asset pricing for affine jump-diffusions. *Econometrica 68*(6), 1343–1376.

Dunn, P. K. and G. K. Smyth (2008). Evaluation of Tweedie exponential dispersion model densities by Fourier inversion. *Statistics and Computing 18*(1), 73–86.

Eddelbuettel, D. and R. François (2011). Rcpp: Seamless R and C++ integration. *Journal of Statistical Software 40*(8), 1–18.

Hansen, L. P. (1982). Large sample properties of generalized method of moments estimators. *Econometrica 50*(4), 1029–1054.

Hogan, R. J. (2014). Fast reverse-mode automatic differentiation using expression templates in C++. *ACM Transactions on Mathematical Software (TOMS) 40*(4), 26.

Jorgensen, B. (1987). Exponential dispersion models. *Journal of the Royal Statistical Society. Series B (Methodological)*, 127–162.

Kleppe, T. S. and H. J. Skaug (2008). Building and fitting non-Gaussian latent variable models via the moment-generating function. *Scandinavian Journal of Statistics 35*(4), 664–676.

Knight, J. L., S. E. Satchell, and J. Yu (2002). Theory & methods: Estimation of the stochastic volatility model by the empirical characteristic function method. *Australian & New Zealand Journal of Statistics 44*(3), 319–335.

Kristensen, K., A. Nielsen, C. W. Berg, H. Skaug, and B. M. Bell (2016). TMB: Automatic differentiation and Laplace approximation. *Journal of Statistical Software 70*(i05).

McNeil, A., R. Frey, and P. Embrechts (2005). *Quantitative Risk Management: Concepts, Techniques, and Tools*. Princeton University Press: Princeton, NJ.

Merton, R. C. (1976). Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics 3*(1), 125–144.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery (1992). Numerical recipes in C. *Cambridge: Cambridge University*.

Preston, S. and A. T. Wood (2012). Approximation of transition densities of stochastic differential equations by saddlepoint methods applied to small-time Ito–Taylor sample-path expansions. *Statistics and Computing 22*(1), 205–217.

R Core Team (2018). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.

Strawderman, R. L. (2004). Computing tail probabilities by numerical fourier inversion: The absolutely continuous case. *Statistica Sinica*, 175–201.

Tweedie, M. (1984). An index which distinguishes between some important exponential families. In *Statistics: Applications and new directions: Proc. Indian statistical institute golden Jubilee International conference*, Volume 579, pp. 6o4.

Wood, A. T., J. G. Booth, and R. W. Butler (1993). Saddlepoint approximations to the CDF of some statistics with nonnormal limit distributions. *Journal of the American Statistical Association 88*(422), 680–686.

Yu, J. (2004). Empirical characteristic function estimation and its applications. *Econometric reviews 23*(2), 93–123.

Zhang, L. and W. M. Schmidt (2016). An approximation of small-time probability density functions in a general jump diffusion model. *Applied Mathematics and Computation 273*, 741–758.

Paper II

An information criterion for automatic gradient tree boosting

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.

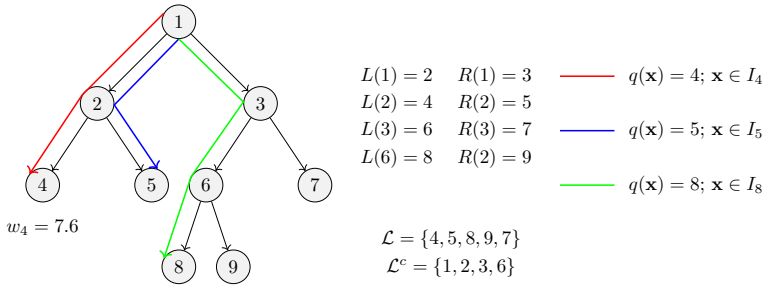

Figure 1: Example of a CART with $T = 5$ leaf nodes ($\mathcal{L}$) and 4 internal nodes ($\mathcal{L}^c$). $\mathbf{w} = (w_4, w_5, w_7, w_8, w_9)$ is the vector of possible predictions. The operator $q(\mathbf{x})$ maps different instance sets ($I_t,\ t \in \mathcal{L}$) to leaf nodes. The mappings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

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

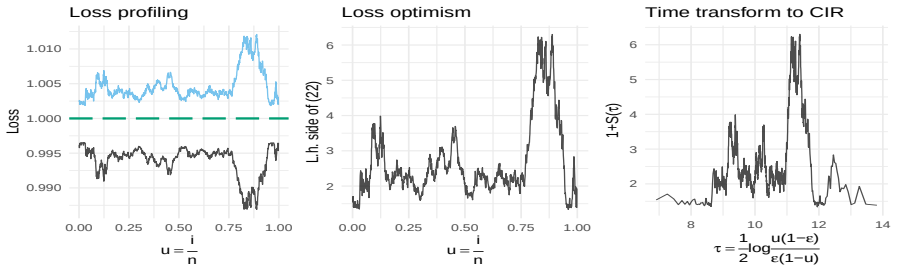

Figure 2: Left: Training loss $\hat{l}(y, f(x;\hat{\theta}))$ (black) and generalization loss $E_{y^0,x^0}\left[\hat{l}(y^0, f(x^0;\hat{\theta}))\right]$ (blue) as a function of $u = \frac{i}{n}$, defined from the sorted order of x_j. Green long-dashed line is the expected loss-value at $\theta_0 = \lim_{n\to\infty}\hat{\theta}_n$ 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

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

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).

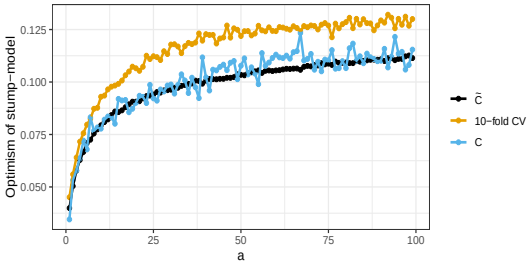

Figure 3: Illustration of Equation (24) by repeatedly simulating $n = 100$ observations in training data, for each value of a possible split-points, under the assumption $y \perp x$, $y \sim N(0, 1)$, for which $\tilde{C}_{stump}$ is asymptotically exact. The feature x is constructed to have the same 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.

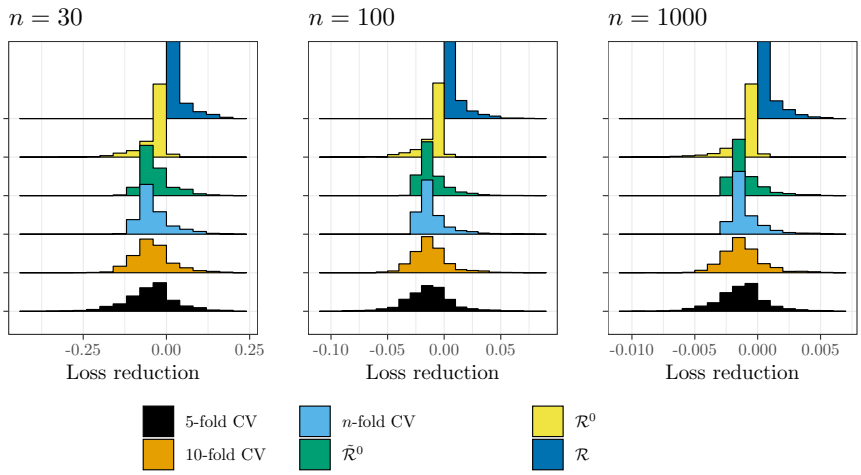

Figure 4: Histograms of single feature root versus stump loss reduction, with $n = \{30, 100, 1000\}$, $a = 1$ and the DGP being $y \sim N(0, 1)$. The results are based on 1000 simulation replica in each case. The two values taken by the feature were simulated uniformly on $(0, 1)$.

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.

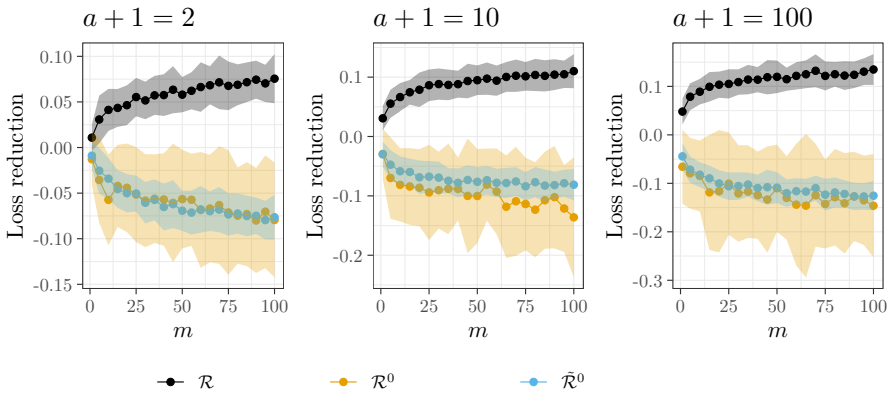

Figure 5: Root versus stump loss reduction as a function of the number of features m, when all features are uninformative. The DGP is $y_i \sim N(0, 1)$, and the m features are simulated conditional on the given number a of potential split points. Dots give the average loss reductions over 100 simulated 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.

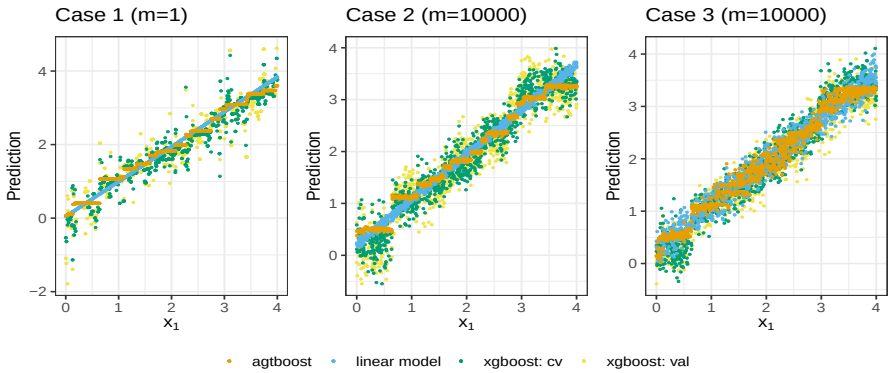

Figure 6: Predictions for the linear model (31) with the different cases corresponding to different design matrices described in Section 4.2. The predictions are evaluated on the test data set features and plotted as function of x_1.

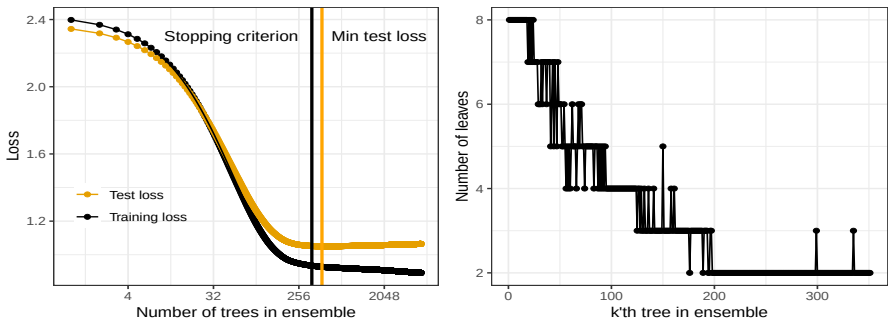

Figure 7: (left) Training loss (black) and test loss (orange) plotted versus the number of boosting iterations or the number of trees in the ensemble (note the $\log_{10}$ axis). Included is a vertical line (orange) representing the iteration number that the stopping criterion (29) terminates the procedure 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

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

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

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

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

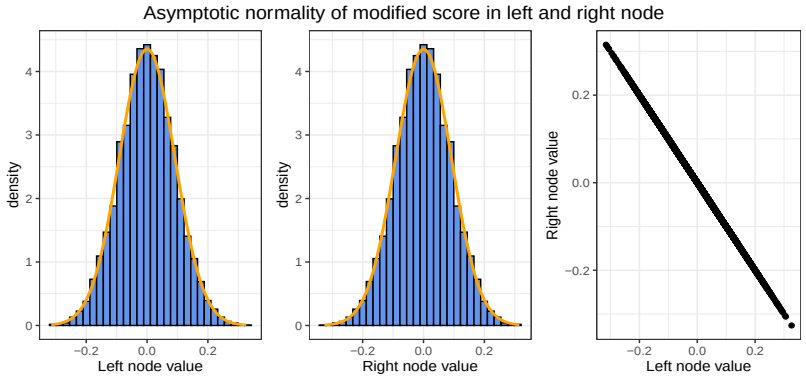

Figure 8: Simulation of the left side of (48), and comparison to the normal on the right. Simulate $n = 100$ observations of $y_i \sim N(3, 1)$ and $x_i \sim U(0, 1)$, to compute derivatives g_i and h_i using the prediction $\hat{y} = 2$ with MSE loss. From this, the modified gradients $\tilde{g}_i$ 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

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

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.

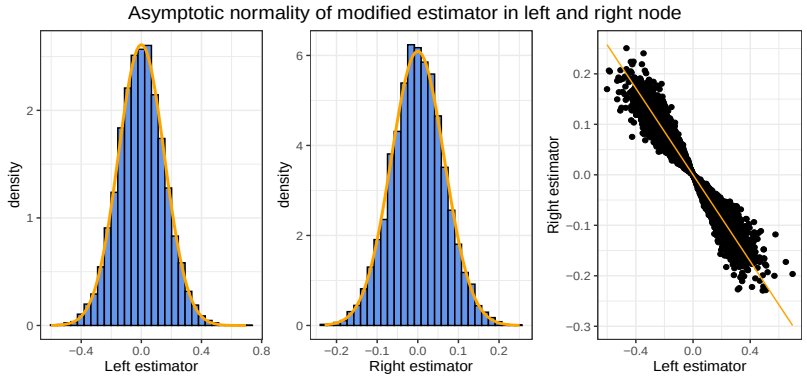

Figure 9: Simulation of (50), where $\hat{\tilde{w}}_l$ and $\hat{\tilde{w}}_r$ are extracted from the simulation experiment explained in the caption of Figure 8, and comparisons with the marginal normal distributions in (51) and (52). Right: Scatter plot of the simulated estimators. Notice that the 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

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

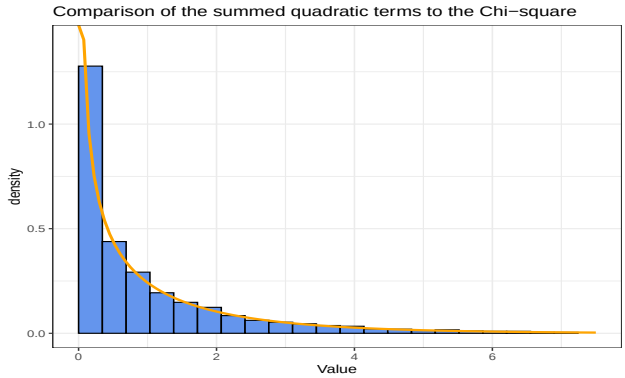

Figure 10: Simulation of the steps in (53) using the squares of $\hat{\tilde{w}}$ in the simulation experiment explained in the caption of Figure 8, and comparison with a Chi-square distribution with one degree of freedom.

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

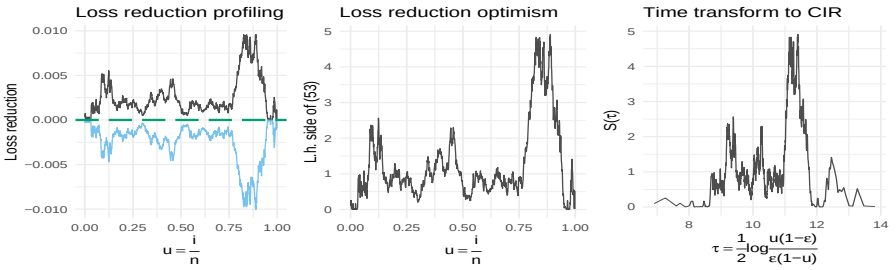

Figure 11: The equivalent and same process as in Figure 2, for loss reduction R and R^0. Left: Reduction in training loss R (black) and reduction in generalization loss $E_{y^0,x^0}\left[R^0\right]$ (blue) as a function of $u = \frac{i}{n}$, defined from the sorted order of x_j. Green long-dashed line 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.

Paper III

agtboost: Adaptive and Automatic Gradient Tree Boosting Computations

agtboost: Adaptive and Automatic Gradient Tree Boosting Computations

Berent Ånund Strømnes Lunde
University of Stavanger

Tore Selland Kleppe
University of Stavanger

Abstract

agtboost is an R package implementing fast gradient tree boosting computations in a manner similar to other established frameworks such as **xgboost** and **LightGBM**, but with significant decreases in computation time and required mathematical and technical knowledge. The package automatically takes care of split/no-split decisions and selects the number of trees in the gradient tree boosting ensemble, i.e., **agtboost** adapts the complexity of the ensemble automatically to the information in the data. All of this is done during a single training run, which is made possible by utilizing developments in information theory for tree algorithms (Lunde, Kleppe, and Skaug 2020). **agtboost** also comes with a feature importance function that eliminates the common practice of inserting noise features. Further, a useful model validation function performs the Kolmogorov-Smirnov test on the learned distribution.

Keywords: gradient tree boosting, information criterion, automatic function estimation, R.

1. Introduction: Tuning of gradient tree boosting

Gradient tree boosting (GTB) (Friedman 2001; Mason, Baxter, Bartlett, and Frean 1999) has risen to prominence for regression problems after the introduction of **xgboost** (Chen and Guestrin 2016). The GTB model is an ensemble-type model, that consist of classification and regression trees (CART) (Breiman, Friedman, Stone, and Olshen 1984) that are learned in an iterative manner. GTB models are very flexible in that they automatically learn non-linear relationships and interaction effects. However, with the increased flexibility of GTB models comes substantial worries of overfitting. The top performing gradient tree boosting libraries, such as **xgboost**, **LightGBM** (Ke, Meng, Finley, Wang, Chen, Ma, Ye, and Liu 2017) and **catboost** (Dorogush, Ershov, and Gulin 2018), all come with a large number of hyperparameters available for manual tuning to constrain the complexity of the GTB models. Training of gradient tree boosting models, in general, thus require some familiarity with both the chosen package, and the data for efficient tuning and application to the problem at hand.

The main focus of the hyperparameters and tuning are to solve the following problems:

- **The complexity of trees:** What are the topology of all the different trees? Too complex trees overfits, while simple stump-models cannot capture interaction effects. This is typically solved using a hyperparameter that penalizes (equally) the number of leaves in the tree. **xgboost** hyperparameters for this are `gamma`, `max_depth`, `min_child_weight`, and `max_leaves`.

- **The number of trees:** How many iterations should the tree-boosting algorithm do before terminating? Too early stopping will leave information unlearned, while too late stopping will see the last trees adapting mostly to noise. An early-stopping hyperparameter is usually tuned to obtain ensembles of adequate size. Tuned in **xgboost** with `nrounds`.

- **Making space for feature trees to learn:** If each tree is optimized alone, early trees will have a tendency to learn additive relationships and information that subsequent trees could learn more efficiently (Friedman, Hastie, Tibshirani *et al.* 2000). An additional downside of large early trees is difficult model-interpretability. The hyperparameter solution typically involves tuning the maximum depth (`max_depth` in **xgboost**) globally for all trees.

The five parameters of **xgboost** mentioned above are typically selected as the top-performing parameters found from k-fold cross validation (CV) (Stone 1974). CV, however, increases computation times extensively, and requires more work through coding and knowledge on the part of the user.

agtboost is an implementation of the theory in Lunde *et al.* (2020), which unlocks computationally fast and automatic solutions to the problems listed above, and as a consequence removes selection of hyperparameters through CV from the problem. The key is an information criterion that can be applied after the greedy binary-splitting profiling procedure used in learning trees. The theory is built upon maximal selection of chi-squared statistics (White 1982; Gombay and Horvath 1990) and the convergence of an empirical process to a continuous time stochastic process. Lunde *et al.* (2020) subsequently discuss how both tree-size and the number of trees then can be chosen automatically. This paper supplements Lunde *et al.* (2020) by describing a package built on this theory, i.e., **agtboost**. Some new innovations are also introduced, which all have their basis in the information criterion. In general, they address the problems with feature importance and the optimization of trees alone.

Note that there exist other hyperparameters that may increase accuracy (but that are not vital) for GTB models. Most notably are parameters for a regularized objective (see e.g., Chen and Guestrin (2016)) and stochastic sampling of observations during boosting iterations (for an overview, see Hastie, Tibshirani, and Friedman (2001) and for recent innovations for GTB see Ke *et al.* (2017)). These features are not yet implemented, but subject to further research, as more work is required to adhere to the philosophy of **agtboost** – that all hyperparameters should be automatically tuned.

This paper starts by introducing gradient tree boosting in Section 2 and the information criterion in Section 3, and proceeds with the innovations and software implementation in Section 4. Section 5 describes **agtboost** from a user's perspective. Section 6 studies and compares the different variants of **agtboost** models for the large sized Higgs dataset. Finally, Section 7 discusses and concludes.

2. Gradient tree boosting

This and the following section closely follow the setup of Lunde *et al.* (2020). The (typical)

objective of gradient tree boosting procedures is the supervised learning problem

$$f(x) = \arg\min_f E[l(y, f(x))], \tag{1}$$

for a response $y \in R$, feature vector $x \in R^m$, and loss function l measuring the difference between the response and prediction $\hat{y} = f(x)$. For gradient boosting to work, we require l to be both differentiable and convex. Then, using training data, say $\mathcal{D}_n = \{(y_i, x_i)\}_{i=1}^n$, we seek to approximate (1) by learning f in an iterative manner: Given a function $f^{(k-1)}$, we seek f_k to minimize

$$f_k(x) = \arg\min_{f_k} E[l(y, f^{(k-1)}(x) + f_k(x))], \tag{2}$$

approximately to second order. This is done by computing the derivatives

$$g_{i,k} = \frac{\partial}{\partial \hat{y}^{(k-1)}} l(y, \hat{y}^{(k-1)})\Big|_{\hat{y}^{(k-1)}=f^{(k-1)}(x)}, \; h_{i,k} = \frac{\partial^2}{\partial (\hat{y}^{(k-1)})^2} l(y, \hat{y}^{(k-1)})\Big|_{\hat{y}^{(k-1)}=f^{(k-1)}(x)}, \tag{3}$$

for each observation in the training data, and then approximate the expected loss by averaging,

$$f_k(x) = \arg\min_f \frac{1}{n} \sum_{i=1}^n l(y, \hat{y}_i^{(k-1)}) + g_{i,k} f_k(x) + \frac{1}{2} h_{i,k} f_k(x)^2. \tag{4}$$

Terminating this procedure at iteration K, the final model is an additive model of the form

$$\hat{y}_i = \hat{y}^{(K)} = \sum_{k=1}^K f_k(x_i). \tag{5}$$

Still, (4) is a hard problem, as the search among all possible functions is obviously infeasible. Therefore, it is necessary to constrain the search over a a family of functions, or "base learners". While multiple choices exist, **agtboost** follows the convention of using CART (Breiman *et al.* 1984).

For a full discussion of decision trees, see (Hastie *et al.* 2001) for a general treatment, and Lunde *et al.* (2020) for details on its use in gradient tree boosting and **agtboost**. We constrain ourselves to a brief mention of important aspects. Firstly, decision trees learn constant predictions (called leaf-weights) in regions of feature space. We let $I_{t,k} = \{i : q_k(x) = t\}$ denote the index-set of training indices that falls into region (or leaf) t, denoted by $q_k(x) = t$, where q_k is the topology of the k'th tree, a function that takes the feature vector and returns the the node-index or corresponding index of region in feature space, t. The prediction from the tree is given by

$$f_k(x_i) = w_{q_k(x_i)}. \tag{6}$$

Secondly, the estimated leaf-weights have closed form in the 2'nd order boosting procedure described above, namely

$$\hat{w}_{t,k} = -\frac{G_{t,k}}{H_{t,k}} \text{ where } G_{t,k} = \sum_{i \in I_{t,k}} g_{i,k} \text{ and } H_{t,k} = \sum_{i \in I_{t,k}} h_{i,k}. \tag{7}$$

Thirdly, the regions of feature space are learned by iteratively splitting all leaf-regions (starting with the full feature space as the only leaf) creating new leaves and regions. The region is split on the split-point that gives the largest reduction in training loss, say R_t, among all possible binary splits. This search is fast, as the training loss, modulus unimportant constant terms, in region t is given as

$$l_t = -\frac{G_{t,k}^2}{2nH_{t,k}}, \tag{8}$$

and enumeration of possible splits can therefore be done in $mn \log n$ time.

The above procedure creates the decision tree f_k, which is then added to the model $f^{(k-1)}$ by

$$f^{(k)} = f^{(k-1)} + \delta f_k, \ \delta \in (0, 1). \tag{9}$$

The constant δ, typically called the "learning rate" or "shrinkage", leaves space for feature models to learn. Values are often taken as "small", but this comes at the added computational cost of an increase in the number of boosting iterations (infinite when $\delta \to 0$) before convergence. The learning rate is the only hyperparameter of the boosting procedure in **agtboost** that is not tuned automatically. The default value is set at 0.01, which should be sufficiently small for most applications without incurring too much computational cost.

3. Information criteria

This section introduces generalization loss-based information criteria, which includes types such as Akaike Information Criterion (AIC) (Akaike 1974), Takeuchi Information Criterion (TIC) (Takeuchi 1976) and Network Information Criterion (NIC) (Murata, Yoshizawa, and Amari 1994). Generalization loss is perhaps better known as an instance of test loss, say $l(y^0, f(x^0))$, where (y^0, x^0) is an observation unseen in the training phase, i.e., not an element and independent of $\mathcal{D}_n$. This specification is important, as (1) is intended for this quantity, and using the training loss, in our case (4) as an estimator, care must be taken as the training loss is biased downwards in expectation. This bias is known as the *optimism* of the training loss (Hastie *et al.* 2001), and denoted C. A generalization loss-based information criterion, say $\tilde{C}$, is intended to capture the size of the optimism, such that adding $\tilde{C}$ to the training loss gives an (at least approximately) unbiased estimator of the expected generalization loss.

The equation behind the generalization loss-based information criteria mentioned above is

$$\begin{aligned} \tilde{C} &= \texttt{tr}\left(E\left[\nabla_\theta^2 l\left(y, f(x;\theta_0)\right)\right] Cov(\hat{\theta})\right), \\ \tilde{C} &\approx E[l(y^0, f(x^0;\hat{\theta}))] - E[l(y_1, f(x_1;\hat{\theta}))], \end{aligned} \tag{10}$$

where (y_1, x_1) is a (random) instance of the training dataset, and θ_0 is the population minimizer of (1) where optimization is done over a parametric family of functions and $\hat{\theta}$ is not at the boundary of parameter space (Burnham and Anderson 2003, Eqn. 7.32).

In the GTB training procedure described in section 2, the model selection questions of added complexity are always at the "local" root (constant prediction) versus stump (a tree with two leaf-nodes) model. It is always one, and only one, node at a time that is tried split in this gradient tree boosting procedure. Thus isolate this relevant node by assuming and

conditioning on that $q(x) = t$, then node t may be referenced and treated as a root. Further, for convenient notation, assume to be at some boosting iteration k, thus dropping notational dependence on current iteration, and let l and r denote the left and right child nodes of node t. The idea of Lunde *et al.* (2020) is to use the optimism of the root (node t, conditioned on data falling into leaf t, i.e., $q(x) = t$) and stump model (split of node t, with the same conditioning as for the root), say C_{root} and C_{stump}, to adjust the root and stump training loss, to see if there is expected a positive reduction in generalization loss,

$$-\frac{1}{2n}\left[\frac{G_t^2}{H_t} - \left(\frac{G_l^2}{H_l} + \frac{G_r^2}{H_r}\right)\right] + C_{\text{root}} - C_{\text{stump}} > 0, \tag{11}$$

which would be used to decide if to split further in a tree, and to see if a tree-stump model f_k could be added to the ensemble f^{k-1}. The root and stump training loss combined in the parenthesis is the loss-reduction, R, that is profiled over for different binary splits. This profiling, however, complicates evaluation of the above inequality as it induces optimism into C_{stump} which (10) cannot handle directly.

We may combine the root and stump optimisms to create a loss-reduction optimism, say $C_R = C_{\text{root}} - C_{\text{stump}}$. Lunde *et al.* (2020) derives the following estimator for the optimism of loss-reduction

$$\tilde{C}_R = -\tilde{C}_{root}\pi_t E\left[B(\mathbf{x})\right], \tag{12}$$

where π_t is the probability of being in node t, estimated by the fraction of training data passed to node t, and $\tilde{C}_{root}$ is calculated as (10) conditioned on being in node t, and may be estimated using the sandwich estimator (Huber *et al.* 1967) together with the empirical Hessian. The remaining quantity in (12), $B(\mathbf{x})$ where $\mathbf{x}$ is the full design matrix of the training data, is a maximally chosen random variable

$$B(\mathbf{x}) = \max_{j \le 1 \le m} B_j(x_{\cdot j}),\ B_j(x_{\cdot j}) = \max_{1 \le k \le a} S_j(\tau_k) \tag{13}$$

defined from a specification of the Cox-Ingersoll-Ross process (CIR) (Cox, Ingersoll, and Ross 1985), $S(\tau)$, with speed of adjustment to the mean 2, long term mean 1 and instantaneous rate of volatility $2\sqrt{2}$, therefore having dynamics given by the SDE

$$dS(\tau) = 2(1 - S(\tau))dt + 2\sqrt{2S(\tau)}dW(\tau), \tag{14}$$

which is "observed" at time-points $\tau_k = \frac{1}{2}\log\frac{u_k(1-\epsilon)}{\epsilon(1-u_k)}$, $\epsilon \to 0$ defined from the split-points $u_k = p(x_{ij} \le s_k)$, $i = 1 : (n-1)$ on feature j. To estimate $E[B(\mathbf{x})]$ is, however, not straight forward. Lunde *et al.* (2020) discuss a solution using exact simulation of the CIR process using $\epsilon = 10^{-7}$, together with the fact that the stationary CIR is in the maximum domain of attraction of the Gumbel distribution (Gombay and Horvath 1990), an independence assumption on features, and numerical integration to evaluate the expectation.

4. Software implementation and innovations

agtboost employs the r.h.s. of (12) to solve the list of problems presented in the introduction, which for other implementations are solved by tuning the hyperparameters using techniques

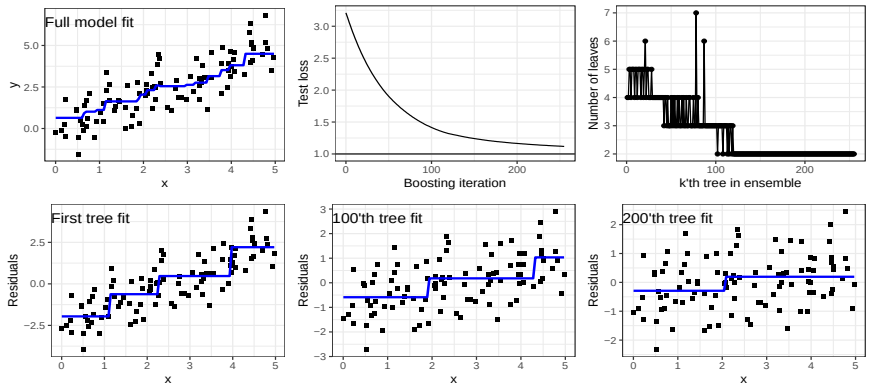

Figure 1: Top: [left] Full model fit on training-data, [middle] test loss versus the number of boosting iterations, with the minimum expected loss possible (true model) included with a horizontal line, and [right] the number of leaves in the k'th tree. Bottom: The First, 100'th and 200'th tree fits to the training MSE-residuals, $r_i^{(k)} = -g_{i,k}/h_{i,k} = y_i - \hat{y}^{(k-1)}$ at respective iterations. The $n = 100$ training and test observations were i.i.d. generated with a linear structural relationship and Gaussian noise, $x \sim U(0, 5)$ and $y \sim N(x, 1)$.

such as cross-validation (CV) (Stone 1974). The use of (12) alleviates the need for hyper-parameters tuned with CV, as it allows the base-learner trees f_k and the ensemble $f^{(k)}$ to stop at a given complexity that is adapted to the training data at hand. Thus significantly increasing the speed of training a gradient tree boosting ensemble. Furthermore, the technical knowledge imposed on the user, with respect to both gradient tree boosting and the dataset at hand, is reduced. **agtboost** is coded in C++ for fast computations, and relies on **Eigen** (Guennebaud, Jacob *et al.* 2010) for linear algebra, the R header files (R Core Team 2018) for some distributions, and **Rcpp** (Eddelbuettel and François 2011) for bindings to R. The remainder of this section goes through the innovations in **agtboost** that directly attacks the previously mentioned tuning-problems.

4.1. Adaptive tree size

Equipped with an information criterion for loss reduction after greedy-split-profiling, the necessary adjustments are rather straight-forward, and are also discussed in Lunde *et al.* (2020). For completeness, the usage of (12) towards selecting the complexity of trees f_k is restated here: After the split that maximizes training loss reduction R is found, the following inequality is tested

$$R + \tilde{C}_R > 0, \tag{15}$$

and if it evaluates to `TRUE`, then two new leaves (and regions) are created and successive splitting on them is performed. This continues until (15) evaluates to `FALSE`. This criterion is employed for all but the first (root) split, which is forced. This forced split is done to assure

some increase in model complexity, as $\sum_i g_i = 0$ is always true and a root model will therefore be equivalent to adding zero to the model.

Figure 1 illustrates this adaptivity: Visually, it is seen that trees assign higher leaf-node predictions to observations with high values of x than to small values of x, thus capturing the structural relationship $y = x$ in the simulated data. Furthermore, of the trees plotted in the lower lane together with residuals $y_i - \hat{y}_i^{(k-1)} = -g_{i,k}/h_{i,k}$ at iterations $k \in \{1, 100, 200\}$, none of them can be seen to be complex enough to adapt to the Gaussian noise in the training data. Indeed, every subsequent iteration reduces the value of test loss, seen from the top-middle panel. Further verification of this is given by the decreasing complexity of trees (measured in terms of number of leaves), corresponding to residuals at early iterations necessarily containing more information than later ones. Thus, early trees therefore tend to be more complex than at later stages of the training.

4.2. Automatic early stopping

The natural stopping criterion for the iterative boosting procedure, in the context of supervised learning, is to stop when the increase in model complexity no longer gives a reduction in generalization loss. From Figure 1, the top-right plot shows that the later iterations tend to be tree-stumps. Indeed, a tree constructed using the method described in Section 4.1 will be a tree stump at the iteration where the natural stopping criterion terminates the algorithm. This is because a more complex tree must have passed the "barrier" (i.e., inequality) (15), and necessarily will have a decrease in generalization loss, as long as $\delta \in (0, 1]$. Care must however be taken, as we are scaling the k'th tree with the learning rate, δ, and 15 might therefore not be used directly. Lunde *et al.* (2020) discuss this: The solution lies in the general equation for Equation (10) (Hastie *et al.* 2001)

$$\tilde{C} = \frac{2}{n} \sum_{i=1}^{n} Cov(y_i, \hat{y}_i), \tag{16}$$

and the optimism therefore scales linearly. The training loss, on the other hand, does not and can be seen to scale with the factor $\delta(2 - \delta)$ from direct computations. Scaling the training loss and optimism appropriately, **agtboost** evaluates a similar inequality as (15), namely

$$\delta(2 - \delta)R + \delta\tilde{C}_R > 0, \tag{17}$$

which if evaluates to `FALSE`, indicates that the increased complexity does not decrease generalization loss, and subsequently the boosting procedure is terminated.

The top-left panel of Figure 1 illustrates the fit of a model that converged after $K = 255$ iterations, and also shows a "convergence plot" (top-middle panel with test loss at different boosting iterations) that flattens out. Indeed, repetitions of the same experiment with an increased number of training instances, n (see top row of Figure 2), shows that the test loss converges on average towards 1, indicated by the horizontal line. This is the expected minimum value possible due to the standard Gaussian noise. Furthermore, if considering the lower-right plot with the fit of the 200'th tree to the residuals at that iteration, we can see that the algorithm still finds information that is hard to see for the naked eye. The algorithm continues for another 55 iterations, that still manages to decrease test loss.

4.3. The global-subset algorithm

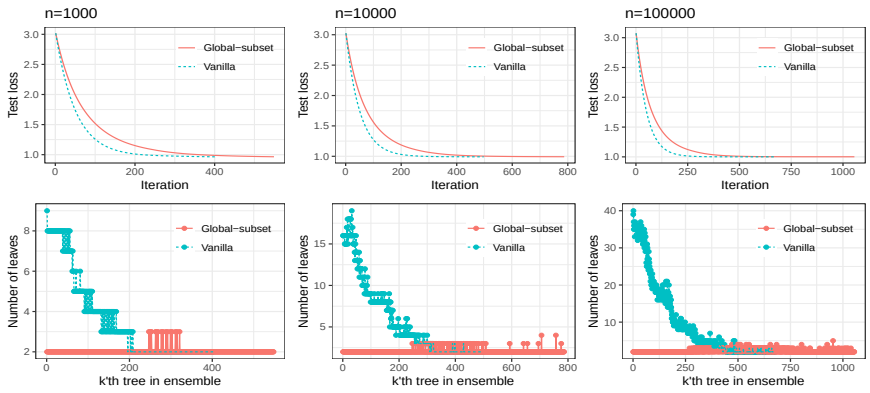

Figure 2: Top: Test loss at different boosting iterations for the global-subset and vanilla algorithms training and testing with an increasing number of observations. Lower: The number of leaves in the k'th tree for the global-subset and vanilla algorithms, also training and testing with the same number of observations as in the top row. The data generating process is the same as described in the caption of Figure 1, but with the number of training and test observations set to $n = \{1000, 10000, 100000\}$ for the three columns.

Equipped with the information criterion (12), it is possible to construct a solution to the problem that each tree is optimized alone, mentioned as the third point in the introduction. For this subsection, denote the reduction in training loss from splitting some node t at the k'th boosting iteration by ${}_kR_t$. For example, the reduction from the root-node split at the same k'th iteration is denoted ${}_kR_1$ and the reduction from splitting the root-node at the $(k+1)$'th boosting iteration is denoted ${}_{k+1}R_1$.

The idea is rather simple, namely to compare the average generalization loss reduction from a chosen split in the k'th boosting iteration, with that of the average generalization loss reduction we would obtain from the root-split in the $(k+1)$'th boosting iteration if the aforementioned split was not performed and the recursive splitting at the k'th iteration terminated. This then allows the tree-boosting algorithm to consider (in a greedy manner) all possible allowed changes in function complexity of the ensemble, not just a deeper tree. The naive approach to do this – at each possible split in the k'th iteration, temporarily terminate, and start on the $(k+1)$'th iteration for inspection of the root-split reduction in generalization loss – is computationally infeasible. Instead, notice that, as $\delta \to 0$, the 2'nd order gradient boosting approximation to the loss is increasingly accurate, and that, at the limit $\delta \to 0$, we have $f_k = f_{k+1}$, as f_k is scaled to zero by the learning rate. Necessarily, we have ${}_kR_1 \approx {}_{k+1}R_1$ and $\tilde{C}_{{}_kR_1} \approx \tilde{C}_{{}_{k+1}R_1}$.

The quantities ${}_{k+1}R_1$ and $\tilde{C}_{{}_{k+1}R_1}$ may be used to adjust the right-hand-side of (15) with the expected reduction in generalization loss of the next split, so that the recursive binary splitting terminates when splitting the next root-node is more beneficial. Using ${}_kR_1$ and $\tilde{C}_{{}_kR_1}$

as replacements, the reformulated split-stopping inequality yields

$$\pi_t^{-1}\left({}_kR_t + \tilde{C}_{{}_kR_t}\right) > \max\left\{0, {}_kR_1 + \tilde{C}_{{}_kR_1}\right\}. \tag{18}$$

The probability π_t is introduced to adjust for the difference in the number of training observations, as the root works on the full dataset, while node t necessarily works on some subset of the data. The inequality (18) is then employed as a replacement for- and in the exact same manner as (15)

Figure 2 illustrates the practical difference in pathological learning behaviour: The data exhibits purely additive behaviour. Friedman *et al.* (2000) argues for a model consisting only of tree-stumps in this case. Both method converges to a test loss of approximately 1, the minimum expected test loss possible for a perfect model, for all values of n. The difference lies in the complexity and number of trees. The vanilla algorithm (using (15)) builds each tree as if it was the last, and already at the first iteration, several regions of feature space will be split into sub-regions, seen from the plots in the lower row. The global-subset algorithm, however, "looks ahead" and often evaluates that terminating the recursive binary splitting procedure and starting on a new boosting iteration is more beneficial. Subsequently, trees are rarely complex and thus easier to interpret, but comes to the cost of a higher number of boosting iterations before terminating by the inequality (17). This cost is decreased, however, as boosting iterations are overall faster than for the vanilla algorithm since individual trees are less complex.

5. Using the agtboost package

The goal of the **agtboost** package is to avoid expert opinions and computationally costly brute force methods with regards to tuning the functional complexity of GTB models. Usage should be as simple as possible. As such, the package has only two main functions, `gbt.train` for training an **agtboost** model, and a predict function that overloads the `predict` function in R. The main responsibility of the user is to identify a "natural" loss-function and link-function. To this end, **agtboost** also comes with a model validation function, `gbt.ksval`, which performs a Kolmogorov-Smirnov test on supplied data, and a function for feature importance, `gbt.importance`, that functions similarly to ordinary feature importance functions (see for instance Hastie *et al.* (2001)) but which calculates reduction in loss with respect to (approximate) generalization loss and not the ordinary training loss. Due to implementation using **Rcpp modules**, saving and loading of **agtboost** cannot be done by the ordinary `save` and `load` functions in R, but is made possible through the functions `gbt.save` and `gbt.load`. Table 1 gives an overview of the implemented loss functions in **agtboost**.

Following is a walk-through of the **agtboost** package, applied to the `caravan.train` and `caravan.test` data (Van Der Putten and van Someren 2000) that comes with the package and documented there. The caravan dataset has a binary response, indicating purchase of caravan insurance, and 85 socio-demographic covariates. Due to the nature of the response, classification using the `logloss` loss function is natural. To train a GTB model, it is only needed to specify the `loss_function` argument in `gbt.train`

```
R> mod <- gbt.train(y = caravan.train$y, x = caravan.train$x,
                    loss_function = "logloss", verbose = 100)
```

Type	Distribution	Link	Comment
`mse`	Gaussian	$\mu = f(x)$	Ordinary regression for continuous response
`logloss`	Bernoulli	$\log\left(\frac{\mu}{1-\mu}\right) = f(x)$	Regression for classification problems
`gamma::neginv`	Gamma	$-\frac{1}{\mu} = f(x)$	Gamma regression for positive continuous response
`gamma::log`	Gamma	$\log(\mu) = f(x)$	regression for positive continuous response
`poisson`	Poisson	$\log(\mu) = f(x)$	Poisson regression for count data exhibiting $Var(y\|x) = E[y\|x]$
`negbinom`	Negative binomial	$\log(\mu) = f(x)$	For count data exhibiting overdispersion. `dispersion` must be supplied to `gbt.train`

Table 1: Overview of the loss functions available in **agtboost**.

```
it: 1   |  n-leaves: 3  |  tr loss: 0.2166  |  gen loss: 0.2167
it: 100  |  n-leaves: 2  |  tr loss: 0.1983  |  gen loss: 0.2019
it: 200  |  n-leaves: 3  |  tr loss: 0.1927  |  gen loss: 0.1987
it: 300  |  n-leaves: 2  |  tr loss: 0.1898  |  gen loss: 0.1978
```

Note the `verbose=100` argument, which creates output at the first and every 100'th iteration. The output consists of the iteration number, the number of leaves of the k'th tree, the training loss and approximate generalization loss. By default, the global-subset algorithm using inequality (18) is used instead of (15), if the latter is preferred, specify `algorithm = "vanilla"` as an argument to `gbt.train`.

The overloaded `predict` function may be used to check the fit on the training data, or to predict new data. To predict data, just pass the model object and the design matrix of data to be predicted

```
R> prob_te <- predict(mod, caravan.test$x)
```

It is often meaningful to do some formal goodness-of-fit test in addition to only visually inspecting the fit. A quite natural and general way to do this for loss functions associated negative log-likelihoods, is to use the Kolmogorov-Smirnov test (Kolmogorov 1933). The idea is to, for a continuous response, perform the CDF transform

$$u_i = p\left(y \leq y_i; \hat{\theta}(x_i)\right), \tag{19}$$

preferably for test-data, and test the u_i's against the standard uniform distribution, which holds if the model is correctly specified. If the response is discrete, then employ the transform

$$u_i = p\left(y \leq y_i - 1; \hat{\theta}(x_i)\right) + Vp\left(y_i; \hat{\theta}(x_i)\right), \tag{20}$$

where $V \sim U(0,1)$, instead of the CDF transform. All of this is implemented in the `gbt.ksval` function. To apply it to the caravan data model, simply write

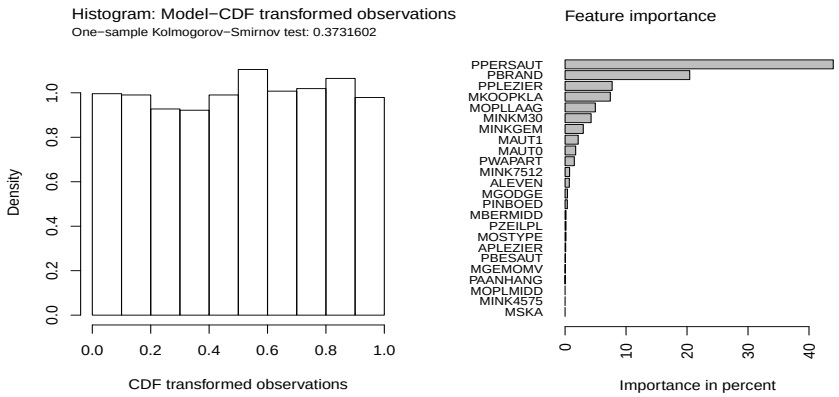

Figure 3: [left] Histogram generated by `gbt.ksval`, where test-response observations are transformed using (20). As the histogram closely resembles the histogram of standard uniformly distributed random variables, the model should not be discarded, something also indicated by the formal Kolmogorov-Smirnov test. [right] Feature importance plot generated by `gbt.importance`.

```
R> gbt.ksval(object = mod, y = caravan.test$y, x = caravan.test$x)

Classification
One-sample Kolmogorov-Smirnov test
data:  u
D = 0.021877, p-value = 0.3732
alternative hypothesis: two-sided
```

which produces output of the test-statistic, the p-value and the histogram in Figure 3. Note that for multi-parameter distributions, such as the Gaussian, gamma and negative binomial, the remaining parameters are assumed constant and maximum-likelihood estimates are produced during evaluation of the `gbt.ksval` function to allow for the transforms (19) and (20). The estimates will then be produced in the output.

In addition to the `gbt.ksval` function, the `gbt.importance` function can be used to inspect the model. This function produces a traditional feature importance plot (see e.g. Hastie *et al.* (2001, Chap. 15.3.2)), but different from other packages, the calculations are with respect to approximate generalization loss. Formally, for non-leaf nodes t in all trees in the ensemble, the value

$$\delta(2-\delta)R_t + \delta\tilde{C}_{R_t}$$

is added to the j-th element of a vector, where j is the j-th feature used for the split. The way in which `gbt.importance` calculates importance, and due to the sparse models produced by `gbt.train`, know-how tricks such as inserting Gaussian noise-features are not necessary

Algorithm	Loss	AUC	Time	#trees	#leaves	#features
			$n = 100$			
vanilla	0.6728	0.5942	0.1417	32	64	1
global-subset	0.6734	0.5942	0.1386	30	60	1
			$n = 1000$			
vanilla	0.6483	0.703	2.335	162	422	7
global-subset	0.6437	0.7071	1.623	184	504	8
			$n = 10000$			
vanilla	0.5692	0.7796	1.191	684	4976	22
global-subset	0.5708	0.7781	1.201	768	4008	18
			$n = 100000$			
vanilla	0.5317	0.8087	32.57	1055	43908	28
global-subset	0.5321	0.8085	34.17	1176	29105	28

Table 2: Results of the two algorithms available in **agtboost** for different number of training samples available. The Loss (Logloss) and AUC is computed on the test set with 1 million observations. The remaining columns are the total number of trees in the models, the total number of leaves summed up over all trees, and the total number of features used by the models. Loss metrics and computation times are almost identical, while complexity and construction differs.

when using **agtboost**. The right plot in Figure 3 produces the feature importance plot for the caravan model. Note that only 24 of 85 possible features are used by the model. Re-training the model with only these relevant features might improve the fit, as the 61 features not used by the model are noise that mask information and enlarges the absolute value of the information criterion (12).

6. Higgs big-data case study

The two variants of `agtboost` is tested across increasing training sizes of a dataset, and their intrinsic behaviour with regards to reduction in loss, number of trees and leaves of trees, numbers of features used, and convergence across boosting iterations is studied. We refer to models using inequality (15) as "vanilla" models, and models using the global-subset algorithm (18) as "global-subset" models. To this end, the Higgs dataset[1] is used. The Higgs data consists of 11 million observations of a binary response and 28 continuous features. The first 10 million observations are used for training, and the last million for testing for which results are reported. The training set is sampled randomly without replacement for $n = \{10^2, 10^3, 10^4, 10^5\}$ observations, and trained on the respective training indices. Tests and reported results are always done on the one-million sized test-set.

In the "Loss" column of Table 2 it is seen that the test-loss is decreasing in the size of the training set, as it should. Figure 4 compliments this result: Each model is seen to converge and values of test loss flatten out as a function of boosting iterations. But, as the training set increases, more information in the data is present that allow lower points of convergence. None of the models are seen to overfit in the number of boosting iterations, as the curves

[1] https://archive.ics.uci.edu/ml/datasets/HIGGS

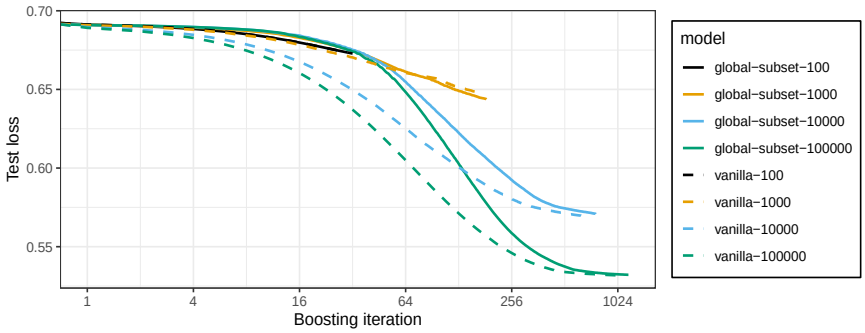

Figure 4: Test loss as a function of boosting iterations, for the vanilla and global-subset models trained on $n \in \{100, 1000, 10000, 100000\}$ training observations. More training observations imply more information, which allow for lower points of convergence and a greater number of boosting iterations. The methods stop at different boosting iterations. Notice the log scale on the horizontal axis.

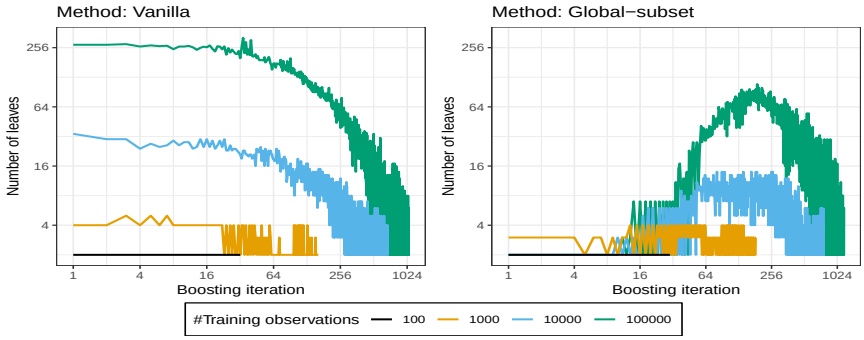

Figure 5: The number of leaves for the k'th tree (boosting iteration k) for different **agtboost** models trained on the Higgs data with a varying number of training observations. Notice the log scale on both the vertical and horizontal axes.

never increase.

While the two variants of **agtboost** converge to similar results in terms of test-loss, and the methods take a similar amount of time (column 5 of Table 2), their behaviour during training and the complexity of the resulting models differ. Figure 5 shows two different ways of learning the structural signal in the data. The early trees of the vanilla algorithm start with deep trees, and as the signal is learned, trees become smaller. Trees from the global-subset algorithm, on the other hand, start out with mere tree stumps, and then increase in size as interaction effects become more beneficial to learn than additive relationships. Trees do however not reach the depth of the deepest early trees of the vanilla algorithm. As interaction effects

are taken into the model, these trees also become smaller before convergence of the boosting algorithm. The total number of trees and leaves of the models are shown in columns 6 and 7 in Table 2. While the global-subset algorithm produces models with a larger number of trees than the vanilla algorithm, the total number of leaves is typically smaller, but without a loss of accuracy. As sparsity is a good defence against the curse of dimensionality, the performance of the global-subset algorithm might become more evident on big-p small-n datasets.

7. Discussion

This paper describes **agtboost**, an R package for gradient tree boosting solving regression-type problems in an automated manner. The package takes advantage of recent innovations in information theory with regards to the splits in gradient boosted trees Lunde *et al.* (2020), implements these in C++ for fast computation and employs **RcppEigen** for bindings to R which provides user-friendly application. The package comes with two different utilizations of the information criterion (12), that vary little in final accuracy and in training time but vary in terms for individual tree-size and complexity. The package can be used for early exploratory data analysis for selecting features and an appropriate loss-function, but also for building a final highly predictive model.

References

Akaike H (1974). "A New Look at the Statistical Model Identification." *IEEE Transactions on Automatic Control*, **19**(6), 716–723.

Breiman L, Friedman J, Stone CJ, Olshen RA (1984). *Classification and Regression Trees*. CRC Press.

Burnham KP, Anderson DR (2003). *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Springer Science & Business Media.

Chen T, Guestrin C (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.

Cox JC, Ingersoll JE, Ross SA (1985). "A Theory of the Term Structure of Interest Rates." *Econometrica*, **53**(2), 385–407.

Dorogush AV, Ershov V, Gulin A (2018). "CatBoost: Gradient Boosting with Categorical Features Support." *arXiv preprint arXiv:1810.11363*.

Eddelbuettel D, François R (2011). "Rcpp: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. doi:10.18637/jss.v040.i08. URL http://www.jstatsoft.org/v40/i08/.

Friedman J, Hastie T, Tibshirani R, *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 JH (2001). "Greedy Function Approximation: a Gradient Boosting Machine." *Annals of Statistics*, pp. 1189–1232.

Gombay E, Horvath L (1990). "Asymptotic Distributions of Maximum Likelihood Tests for Change in the Mean." *Biometrika*, **77**(2), 411–414.

Guennebaud G, Jacob B, *et al.* (2010). "Eigen v3." http://eigen.tuxfamily.org.

Hastie T, Tibshirani R, Friedman J (2001). *The Elements of Statistical Learning*. Springer Series in Statistics New York, NY, USA:.

Huber PJ, *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.

Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, Liu TY (2017). "LightGBM: A Highly Efficient Gradient Boosting Decision Tree." In *Advances in Neural Information Processing Systems*, pp. 3146–3154.

Kolmogorov A (1933). "Sulla determinazione empirica di una lgge di distribuzione." *Inst. Ital. Attuari, Giorn.*, **4**, 83–91.

Lunde BÅS, Kleppe TS, Skaug HJ (2020). "An information criterion for automatic gradient tree boosting." *arXiv preprint arXiv:2008.05926*.

Mason L, Baxter J, Bartlett P, Frean M (1999). "Boosting Algorithms as Gradient Descent in Function Space (Technical Report)." *RSISE, Australian National University*.

Murata N, Yoshizawa S, Amari Si (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.

R Core Team (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Stone M (1974). "Cross-Validatory Choice and Assessment of Statistical Predictions." *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 111–147.

Takeuchi K (1976). "Distribution of Information Statistics and Validity Criteria of Models." *Mathematical Science*, **153**, 12–18.

Van Der Putten P, van Someren M (2000). "CoIL challenge 2000: The insurance company case." *Technical report*, Technical Report 2000–09, Leiden Institute of Advanced Computer Science ....

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

Affiliation:

Berent Ånund Strømnes Lunde
Department of Mathematics and Physics
Faculty of Science and Technology
University of Stavanger
Kristine Bonnevies vei 22
4021 Stavanger, Norway
E-mail: berent.a.lunde@uis.no