Mathematical Statistics
Stockholm University

A note on multi-parametric gradient boosting machines with non-life insurance applications

Łukasz Delong
Mathias Lindholm
Henning Zakrisson

Research Report 2023:2

ISSN 1650-0377

Postal address:

Mathematical Statistics
Dept. of Mathematics
Stockholm University
SE-106 91 Stockholm
Sweden

Internet:

http://www.math.su.se

Mathematical Statistics
Stockholm University
Research Report **2023:2**,
http://www.math.su.se

A note on multi-parametric gradient boosting machines with non-life insurance applications

Łukasz Delong, Mathias Lindholm, and Henning Zakrisson

February 2023

Abstract

In the present note a general multi-parametric gradient boosting machine (GBM) approach is introduced. The starting point is a standard univariate GBM, which is generalised to higher dimensions by using cyclic coordinate descent. This allows for different covariate dependencies in different dimensions. The suggested approach is also easily extended to, e.g., multi-parametric versions of XGBoost.

Given weak assumptions the method can be shown to converge for convex negative log-likelihood loss functions, which is the case, e.g., for d-parameter exponential families. Further, when having d-parametric distribution functions, it is important to design appropriate early stopping schemes. A simple alternative is introduced and more advanced schemes are discussed.

The flexibility of the method is illustrated both on simulated and real insurance data examples using different multi-parametric distributions, with both convex and non-convex losses.

1. Introduction

Standard regression analysis such as generalised linear models (GLMs), using exponential dispersion models (EDMs), focus on the situation

$$Y|\boldsymbol{X} = \boldsymbol{x} \sim F(\boldsymbol{\theta}(\boldsymbol{x})), \quad \boldsymbol{x} \in \mathcal{X},$$

where $\boldsymbol{\theta}(\boldsymbol{x}) = (\mu(\boldsymbol{x}), \phi)^\intercal \in \mathbb{R}^2$, i.e.

$$\mathrm{E}[Y \mid \boldsymbol{X} = \boldsymbol{x}] = \mu(\boldsymbol{x}), \quad \text{and} \quad \mathrm{Var}(Y \mid \boldsymbol{X} = \boldsymbol{x}) = u(\mu(\boldsymbol{x}), \phi), \tag{1}$$

where $u(\cdot)$ is a function depending on the particular EDM, see e.g. Jørgensen (1997). For ease of exposition the possible dependence on deterministic weights has been suppressed. The important point here is that it is only one of the dimensions of $\boldsymbol{\theta}(\boldsymbol{x})$, $\theta_1(\boldsymbol{x}) := \mu(\boldsymbol{x})$, that depends on covariates. By using a constant dispersion parameter ϕ the mean function can be estimated in isolation using a one-parametric loss function. In actuarial applications it is standard to use, e.g., Poisson, Negative binomial and Gamma regression models that all can be parametrised in agreement with (1), which extends to Tweedie models with fixed power variance parameter ξ; see e.g. Ohlsson & Johansson (2010) and Jørgensen (1997). Here one can note that Poisson is truly one-parametric ($\phi \equiv 1$) with convex negative log-likelihood under the canonical link choice, Gamma in its general form belongs to the two-parametric exponential family with convex negative log-likelihood using the canonical link, whereas Negative binomial in its general form does not belong to the two-parametric exponential family and lacks a convex negative log-likelihood; see e.g. Example 9.33 in Barndorff-Nielsen (1978).

Further, when discussing d-parametric models there is no obvious analogy to a deviance function. Due to this we will focus on the negative log-likelihood as the loss function to be minimised when fitting d-parametric models.

When models of type (1) are not sufficient to capture the complexity in data, the natural extension is to allow for covariate dependence in the dispersion dimension, i.e., to introduce $\phi(\boldsymbol{x})$. One way of doing this, still within a GLM framework, is to use so-called double GLMs (DGLMs); see e.g. Smyth (1989). In its simplest form, this relies on noting that one can alternate between the estimation in the μ-dimension and the estimation in the ϕ-dimension, where a certain saddle point approximation is used in order to arrive at an approximate Gamma GLM for the estimation of ϕ; see e.g. Smyth (1989).

There are two obvious drawbacks with the above procedure:

(i) The estimation relies on that one a priori has managed to specify sufficiently flexible parametric forms to describe $\mu(\boldsymbol{x})$ (and $\phi(\boldsymbol{x})$),
(ii) when considering d-parametric models there are not always simple moment parametrisations.

Concerning (i), here one could, of course, use splines and similar models, but what tends to be hard to capture in practice is (transformed) interaction terms.

In the setup of (1), so-called gradient boosting machines (GBMs), see Friedman (2001), have proved to be powerful and easy to use in order to resolve the difficulties of (i) above in a data driven manner. GBMs define a class of additive functional regression models, where so-called "weak learners", such as shallow trees, are fitted and added iteratively in order to successively construct a more complex model. This approach is closely connected to forward stage-wise regression, since the modelling starts from an intercept only model. In the present paper we will extend standard GBMs, as defined in Friedman (2001) using shallow trees as learners, to a multi-parametric setting. This allows one to estimate arbitrary d-parametric parameter functions, where the parameters do not need to correspond to, e.g., moments. Further, given this setup, it is straightforward to construct multi-parametric extensions of related models such as XGBoost; see e.g. Chen & Guestrin (2016), and LightGBM; see e.g. Ke et al. (2017). For more on ensemble based methods, also covering boosting, see Friedman & Popescu (2008), and the references therein.

The approach introduced in the present paper relies on cyclic gradient descent; see, e.g., Luenberger (2016, Chap. 8.6), Bazaraa et al. (2006, Chap. 8.5), and Saha & Tewari (2013), and is simple to implement. Further, as with all over-parametrised models, regularisation will be necessary and a simple multi-parametric early stopping scheme is presented together with possible extensions.

Moreover, the basic algorithm can be shown to converge on training data under weak regularity conditions when assuming a convex loss.

Our contribution is to present a generic method to fit unknown multi-parametric regression functions using gradient boosting machines. There has been a recent interest in bespoke models using similar ideas, and the presented material provides a unified framework in a minimalistic setting: see for example; Lee (2020) introduces multi-parametric Delta-boosting, which performs joint parameter updates on multi-parametric Negative Binomial data; Lee (2021) introduce cyclical multi-parametric Delta-boosting for zero-inflated count models; Meng et al. (2022) introduces cyclically boosted trees for zero-inflated Poisson and Negative binomial models.

Concerning other related work, in Sigrist (2021) a non-cyclic multi-parametric GBM approach is mentioned, but not analysed in detail. Another closely related method is the gamboostLSS; see Mayr et al. (2012) and Thomas et al. (2018). The intended usage of gamboostLSS is to obtain an extension of generalised additive models, i.e., where one uses basis functions applied to (combinations of) components of the covariate vector, using gradient boosting machines when not only modelling the mean, but also location, scale and skewness. In Mayr et al. (2012) and Thomas et al. (2018) they also consider other types of base learners than regression trees. The gamboostLSS method is a cyclic multi-parametric GBM, and if one uses a single learner taking the full covariate vector as argument, together with using arbitrary parameter vectors instead of moments, one obtains the method discussed in the present paper.

The remainder of the paper is organised as follows: In Section 2 background on univariate GBMs is presented, see Section 2.1, together with the extension to cyclic multi-parametric GBMs in Section 2.2. Section 3 covers basic results on convergence and discusses the influence of using different parametrisations, which is followed by Section 4 that introduces a method for dimension dependent early stopping. The paper ends with a number of numerical examples in Section 5 together with a discussion of extensions and concluding remarks in Section 6.

2. Estimating multi-parameter GBMs

Let $(y_i, \boldsymbol{x}_i)_{i=1}^n$ be an i.i.d. sample from $Y|\boldsymbol{X} = \boldsymbol{x} \sim F(\boldsymbol{\theta}(\boldsymbol{x}))$, where $\boldsymbol{x} \in \mathcal{X}$ is a vector of covariates and where $\boldsymbol{\theta}(\boldsymbol{x}) \in \mathbb{R}^d$ is a parameter function that parameterizes the distribution F. The aim of a multi-parametric GBM is to estimate the unknown $\boldsymbol{\theta}(\boldsymbol{x})$ by minimizing the negative log-likelihood $\sum_i \mathcal{L}(y_i; \boldsymbol{\theta}(\boldsymbol{x}_i))$. Before turning our attention to the multi-parametric situation, the case $d = 1$ is treated in some detail.

2.1. One-parameter GBMs.

The basic idea in Friedman (2001) is as follows, assuming that $\mathcal{L}(y; \theta)$ is differentiable w.r.t. θ: Given a starting guess $\widehat{\theta}(\boldsymbol{x})$ of the unknown function $\theta(\boldsymbol{x})$, the direction in which the loss function improves the most for observation $(y_i, \boldsymbol{x}_i)$ with $\widehat{\theta}_i := \widehat{\theta}(\boldsymbol{x}_i)$ is given by the negative of the gradient

$$g(\boldsymbol{x}_i; \widehat{\theta}(\boldsymbol{x}_i)) := \frac{\partial}{\partial \theta} \mathcal{L}(y_i; \theta) \Big|_{\theta = \widehat{\theta}_i}.$$

Hence, by following the negative gradient direction for each $(y_i, \boldsymbol{x}_i)$ the updating procedure in iteration k becomes; see e.g. Luenberger (2016), Nesterov (2018),

$$\widehat{\gamma}_k := \underset{\gamma \in \mathbb{R}_+}{\arg\min} \sum_{i=1}^n \mathcal{L}(y_i, \widehat{\theta}_{i,k-1} - \gamma g_{i,k}),$$

where $g_{i,k} := g(\boldsymbol{x}_i; \widehat{\theta}_{i,k-1})$, and where

$$\widehat{\theta}_k(\boldsymbol{x}_i) := \widehat{\theta}_{i,k-1} - \widehat{\gamma}_k g_{i,k}.$$

Even if $\widehat{\theta}_{i,k} \to \theta_i^*$ as $k \to \infty$ for all i, this does not allow us to learn an unknown functional form. In order to achieve this, Friedman (2001) suggests to include the additional step of fitting an L^2-regression tree to the individual observations' gradients in each iteration: Let

$$h(\boldsymbol{x}; \boldsymbol{\nu}) := \sum_{l=1}^w \delta_l 1_{\{\boldsymbol{x} \in \mathcal{R}_l\}},$$

denote a binary split regression tree with w terminal nodes, with $\boldsymbol{\nu} := (\delta_l, \mathcal{R}_l)_{l=1}^w$. The parameters defining the regression tree, $\boldsymbol{\nu}$, takes on values in $\mathcal{V}$, where $\mathcal{V}$ corresponds to the set of recursive partitionings of $\mathcal{X}$ based on $\log_2(w)$ binary splits of single covariate dimensions that defines the w terminal nodes together with the associated w-dimensional node vectors taking values in $\mathbb{R}^w$; see e.g. Chapter 2.4.1 in Breiman et al. (1984). This general definition is in practice restricted to partitions based on the observed covariate values; see e.g. the CART algorithm in Breiman et al. (1984).

By using the $g_{i,k}$s as working responses we can fit an L^2-regression tree according to

$$\widehat{\boldsymbol{\nu}}_k := \underset{\boldsymbol{\nu}\in\mathcal{V}}{\arg\min} \sum_{i=1}^{n} (g_{i,k} - h(\boldsymbol{x}_i; \boldsymbol{\nu}))^2$$

The univariate GBM procedure that will serve as a basis for all further analysis is summarised in Algorithm 1, see e.g. Friedman (2001).

Algorithm 1 *One-parametric GBM*

Initialise: Let

- $(y_i, \boldsymbol{x}_i)_{i=1}^n$ be an i.i.d. sample from $Y|\boldsymbol{X} = \boldsymbol{x} \sim F(\theta(\boldsymbol{x}))$, $\theta(\boldsymbol{x}) \in \mathbb{R}$,
- $\mathcal{L}(y; \theta)$ be the negative log-likelihood of distribution F,
- $\epsilon \in (0, 1]$ be a shrinkage factor,
- κ be the number of boosting steps to be used,
- $\widehat{\theta}_0(\boldsymbol{x}) = \underset{\theta\in\mathbb{R}}{\arg\min} \sum_{i=1}^{n} \mathcal{L}(y_i; \theta)$.

For $k = 1, \ldots, \kappa$ **do**:

(i) **Compute:**

$$g_{i,k} = \frac{\partial}{\partial\theta} \mathcal{L}(y_i; \theta)\Big|_{\theta=\widehat{\theta}_{k-1}(\boldsymbol{x}_i)}, \ i = 1, \ldots, n.$$

(ii) **Approximate:** Fit a tree

$$\widehat{\boldsymbol{\nu}}_k = \underset{\boldsymbol{\nu}\in\mathcal{V}}{\arg\min} \sum_{i=1}^{n} (g_{i,k} - h(\boldsymbol{x}_i; \boldsymbol{\nu}))^2,$$

and adjust terminal node values according to

$$\widehat{\gamma}_{k,l} = \underset{\gamma\in\mathbb{R}}{\arg\min} \sum_{i:\boldsymbol{x}_i\in\widehat{\mathcal{R}}_{k,l}} \mathcal{L}(y_i; \widehat{\theta}_{k-1}(\boldsymbol{x}_i) + \gamma).$$

(iii) **Update:**

$$\widehat{\theta}_k(\boldsymbol{x}) = \widehat{\theta}_{k-1}(\boldsymbol{x}) + \epsilon \sum_{l=1}^{w} \widehat{\gamma}_{k,l} 1_{\{\boldsymbol{x}\in\widehat{\mathcal{R}}_{k,l}\}}.$$

End.

Output: $\widehat{\theta}_\kappa(\boldsymbol{x})$

2.2. **Multi-parametric GBMs.** The natural way to extend a GBM to a multi-parametric setting is to use cyclic gradient (or coordinate) descent; see e.g. Luenberger (2016, Chap. 8.6), Bazaraa et al. (2006, Chap. 8.5), and Saha & Tewari (2013). If we let $\boldsymbol{\theta} \in \mathbb{R}^d$ denote the d-dimensional parameter vector, this means that we need to introduce the dimension dependent partial derivatives

$$g_{i,j} := \frac{\partial}{\partial\theta_j} \mathcal{L}(y_i; \boldsymbol{\theta})\Big|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}}.$$

In this way each parameter dimension

(i) can have unique covariate dependencies,
(ii) is allowed to use an individual number of boosting steps,
(iii) is allowed to have an individual learning rate,
(iv) can have parameters θ_j that live on different scales.

In order to be able to describe our proposed multi-parametric GBM in Algorithm 2 the following notation is needed: Let $\widehat{\boldsymbol{\theta}}_{j,k}(\boldsymbol{x})$ denote the estimated parameter function $\widehat{\boldsymbol{\theta}}(\boldsymbol{x})$ after the kth update of the jth dimension. That is,

$$\widehat{\boldsymbol{\theta}}_{j,k}(\boldsymbol{x}) = \left(\widehat{\boldsymbol{\theta}}_{1,k}(\boldsymbol{x}), \ldots, \widehat{\boldsymbol{\theta}}_{j-1,k}(\boldsymbol{x}), \widehat{\boldsymbol{\theta}}_{j,k}(\boldsymbol{x}), \widehat{\boldsymbol{\theta}}_{j+1,k-1}(\boldsymbol{x}), \ldots, \widehat{\boldsymbol{\theta}}_{d,k-1}(\boldsymbol{x})\right)$$

and $\widehat{\boldsymbol{\theta}}_{j,0}(\boldsymbol{x}) = \widehat{\boldsymbol{\theta}}_0(\boldsymbol{x})$ for all $j = 1, \ldots, d$.

Algorithm 2 *Multi-parametric GBM*

Initialise: Let

- $(y_i, \boldsymbol{x}_i)_{i=1}^n$ be an i.i.d. sample from $Y|\boldsymbol{x} \sim F(\boldsymbol{\theta}(\boldsymbol{x}))$, $\theta(\boldsymbol{x}) \in \mathbb{R}^d$,
- $\mathcal{L}(y; \boldsymbol{\theta})$ be the negative log-likelihood for distribution F,
- $\epsilon_j \in (0, 1]$, $j = 1, \ldots, d$, be shrinkage parameters,
- κ_j, $j = 1, \ldots, d$, be the number of boosting steps to be used, and set

$$\kappa^+ := \max_{1 \le j \le d} \kappa_j,$$

- $\boldsymbol{e}_j$ be the jth unit vector in $\mathbb{R}^d$,
- $\widehat{\boldsymbol{\theta}}_0(\boldsymbol{x}) = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \sum_{i=1}^n \mathcal{L}(y_i; \boldsymbol{\theta})$.

For $k = 1, \ldots, \kappa^+$ **do:**

For $j = 1, \ldots, d$ **do:**

If $k \le \kappa_j$ **do:**

(i) **Set:**

$$\widehat{\boldsymbol{\theta}}_{k-1}(\boldsymbol{x}) := \begin{cases} \widehat{\boldsymbol{\theta}}_{d,k-1}(\boldsymbol{x}), & j = 1 \\ \widehat{\boldsymbol{\theta}}_{j-1,k}(\boldsymbol{x}), & j = 2, \ldots, d \end{cases}$$

(ii) **Compute:**

$$g_{i,j,k} = \frac{\partial}{\partial \theta_j} \mathcal{L}(y_i; \boldsymbol{\theta})\Big|_{\boldsymbol{\theta} = \widehat{\boldsymbol{\theta}}_{k-1}(\boldsymbol{x}_i)}, \; i = 1, \ldots, n.$$

(iii) **Approximate:** Fit a tree

$$\widehat{\boldsymbol{\nu}}_{j,k} = \arg\min_{\boldsymbol{\nu} \in \mathcal{V}} \sum_{i=1}^n (g_{i,j,k} - h(\boldsymbol{x}_i; \boldsymbol{\nu}))^2,$$

and adjust terminal node values according to

$$\widehat{\gamma}_{j,k,l} = \arg\min_{\gamma \in \mathbb{R}} \sum_{i: \boldsymbol{x}_i \in \widehat{\mathcal{R}}_{j,k,l}} \mathcal{L}(y_i; \widehat{\boldsymbol{\theta}}_{k-1}(\boldsymbol{x}_i) + \boldsymbol{e}_j \gamma).$$

(iv) **Update:**

$$\widehat{\boldsymbol{\theta}}_{j,k}(\boldsymbol{x}) = \widehat{\boldsymbol{\theta}}_{k-1}(\boldsymbol{x}) + \boldsymbol{e}_j \epsilon_j \sum_{l=1}^w \widehat{\gamma}_{j,k,l} 1_{\{\boldsymbol{x} \in \widehat{\mathcal{R}}_{j,k,l}\}}.$$

Else set:

$$\widehat{\boldsymbol{\theta}}_{j,k}(\boldsymbol{x}) = \begin{cases} \widehat{\boldsymbol{\theta}}_{d,k-1}(\boldsymbol{x}), & j = 1 \\ \widehat{\boldsymbol{\theta}}_{j-1,k}(\boldsymbol{x}), & j = 2, \ldots, d \end{cases}$$

End.

End.

Output: $\widehat{\boldsymbol{\theta}}_{d,\kappa^+}(\boldsymbol{x})$

Remark 1.

(a) *As for Algorithm 1, Algorithm 2 uses a gradient approximation in each iteration in order to partition data. Given this partition, optimal node values are obtained based on the original loss function. This is different from using, e.g. a DGLM which iterates cyclically between a mean model and dispersion model based on a saddle point approximation of the loss function, see e.g. Smyth (1989). In Section 5 a cyclic GLM approach will be used as a benchmark, for more on this, see CGLM in Section 5.*

(b) *Concerning the updating order in a GBM setting, see Thomas et al. (2018) where cyclic updating is compared with updating in the steepest partial derivative dimension. In Thomas et al. (2018) it is stated that the speed of convergence may be improved using the steepest descent direction, but convergence per se is not affected. Due to this, we continue with the simpler cyclic updating procedure.*

(c) *In a standard GAM with covariates $\boldsymbol{x}$ one models the link transformed mean function, typically, starting with adding functions $f_i(x_i)$, moving on to $f_{ij}(x_i, x_j)$ and so on. An extension of this is to use so-called GAMLSS, see e.g. Rigby & Stasinopoulos (2005), where location, scale, and shape parameters are modelled simultaneously in a GAM manner. A gradient based cyclic boosted version of GAMLSS, gamboostLSS, is introduced in Mayr et al. (2012). Using gamboostLSS on an arbitrary parameter vector $\boldsymbol{\theta}$ with a single function $f(\boldsymbol{x})$ in all parameter dimensions will recover Algorithm 2 in the present paper. This, however, is a non-standard usage of gamboostLSS, see e.g. the usage in Sections 3 and 4 in Mayr et al. (2012) and Section 4 in Thomas et al. (2018).*

3. Convergence, parametrisations, and hyper-parameter tuning

3.1. **Convergence and parametrisations.** Recall from Section 1 that for GLMs it is possible to estimate the mean function separately from the dispersion parameter dimension. This, however, relies on that the dispersion dimension of $\boldsymbol{\theta}(\boldsymbol{x})$ is constant, i.e. $\theta_1(\boldsymbol{x}) := \mu(\boldsymbol{x})$ and $\theta_2(\boldsymbol{x}) = \phi(\boldsymbol{x}) \equiv \phi$ constant. If this is not the case it is still, of course, possible to estimate the mean and dispersion functions separately if the likelihood factorises into one part only depending on μ and one part only depending on ϕ. This, however, is a very strong assumption that is not even fulfilled for a Gaussian distribution with mean $\mu(\boldsymbol{x})$ and variance $\sigma^2(\boldsymbol{x})$. Consequently, in general, estimation needs to be done either by updating the parameter dimensions jointly, or cyclically. For the purposes of the present paper, as argued in Section 2.2, cyclic updating is more natural.

Further, when turning to d-parametric distributions it is not always possible to obtain analogs to mean dispersion parametrisations. For the special case when having distributions that belong to the proper d-parametric exponential family, it is always possible to obtain *block diagonal* Fisher parametrisations, see e.g. Ch. 9.8(vi) in Barndorff-Nielsen (1978) and Lemma 2.3 in Barndorff-Nielsen (1988). This is particularly helpful when $d = 2$, but remains problematic for $d > 2$. Still, as stated above, Fisher orthogonality is still too weak in order to justify estimating the regression functions in different dimensions separately. With this said, it may still be computationally beneficial to use a Fisher (block) orthogonal parametrisation if available. We will henceforth focus on estimating $\boldsymbol{\theta}(\boldsymbol{x})$ directly.

When it comes to convergence we start with basic results for the univariate GBM:

Proposition 1. *Let $(y_i, \boldsymbol{x}_i)_{i=1}^n$ be an i.i.d. sample from $Y|\boldsymbol{x} \sim F(\theta(\boldsymbol{x}))$, $\theta(\boldsymbol{x}) \in \mathbb{R}$. Assume that $\mathcal{L}(y, \theta)$ is convex in θ and that the loss for a given y evaluated in the unique minimum is finite. Then Algorithm 1 converges.*

Proof. Given that $\mathcal{L}(y, \theta)$ is convex in θ there are unique minimisers for each observation $(y_i, \boldsymbol{x}_i)$, denoted by θ_i^*. That is, $\mathcal{L}(y_i, \widetilde{\theta}_i) \geq \mathcal{L}(y_i, \theta_i^*)$ for all $\widetilde{\theta}_i := \widetilde{\theta}(\boldsymbol{x}_i) \neq \theta_i^*$.

Next, in iteration k in Algorithm 1, starting from our current estimate $\widehat{\theta}_{k-1}(\boldsymbol{x})$, Step (ii) produces new node regions that defines the new tree to be added to the previous update of the function approximation. Given the node regions, the node values are updated according to

$$\widehat{\gamma}_{k,l} = \underset{\gamma \in \mathbb{R}}{\arg\min} \sum_{i:\boldsymbol{x}_i \in \widehat{\mathcal{R}}_{k,l}} \mathcal{L}(y_i; \widehat{\theta}_{k-1}(\boldsymbol{x}_i) + \gamma),$$

and by the assumed convexity of $\mathcal{L}(y,\theta)$ it follows that

$$\sum_{i:\boldsymbol{x}_i\in\widehat{\mathcal{R}}_{k,l}}\mathcal{L}(y_i;\widehat{\theta}_{k-1}(\boldsymbol{x}_i))\geq\min_{\gamma\in\mathbb{R}}\sum_{i:\boldsymbol{x}_i\in\widehat{\mathcal{R}}_{k,l}}\mathcal{L}(y_i;\widehat{\theta}_{k-1}(\boldsymbol{x}_i)+\gamma)\geq\sum_{i:\boldsymbol{x}_i\in\widehat{\mathcal{R}}_{k,l}}\mathcal{L}(y_i;\widehat{\theta}_i^*)>-\infty,$$

where the last strict inequality follows by assumption. Since this relation holds for all k, this means that the overall loss will be non-increasing and bounded from below, and the desired conclusion follows from monotone convergence. □

Recall that the regular d-parametric exponential family has a convex negative log-likelihood, see e.g. Ch. 9.3 in Barndorff-Nielsen (1978), which means that Algorithm 2 will converge when optimising in any single one of the d-dimensions, when keeping the other dimensions constant. Note, however, that Proposition 1 only states that Algorithm 1 converges if sufficiently large κs are used, and this is a pure in-sample result.

If we continue with the univariate GBM and Algorithm 1: Introduce the stacked vector $\widehat{\boldsymbol{\theta}}_k^+$ defined as

$$\widehat{\boldsymbol{\theta}}_k^+:=(\widehat{\theta}_k(\boldsymbol{x}_1),\ldots,\widehat{\theta}_k(\boldsymbol{x}_n))^\mathsf{T},$$

together with the shorthand notation

$$\mathcal{L}(\boldsymbol{\theta}^+):=-\sum_{i=1}^n\mathcal{L}(y_i;(\boldsymbol{\theta}^+)_i),$$

where $(\boldsymbol{\theta}^+)_i:=\theta_i^+:=\theta(\boldsymbol{x}_i)$ and let

$$(\nabla\mathcal{L}(\boldsymbol{\theta}^+))_i:=\frac{\partial}{\partial\theta_i^+}\mathcal{L}(\boldsymbol{\theta}^+),\quad i=1,\ldots,n.$$

The optimal stacked $\boldsymbol{\theta}^+$ vector can be found using the gradient updating procedure for $j>0$

$$\widehat{\boldsymbol{\theta}}_k^+:=\widehat{\boldsymbol{\theta}}_{k-1}^+-\gamma\nabla\mathcal{L}(\widehat{\boldsymbol{\theta}}_{k-1}^+),$$

where $\widehat{\boldsymbol{\theta}}_k^+$ denotes the kth parameter vector update, and where we use the shorthand notation

$$(\nabla\mathcal{L}(\widehat{\boldsymbol{\theta}}_k^+))_j:=\frac{\partial}{\partial\theta_j^+}\mathcal{L}(\boldsymbol{\theta}^+)\bigg|_{\boldsymbol{\theta}^+=(\widehat{\boldsymbol{\theta}}_k^+)_j}.$$

This allows us to state the following basic result:

Lemma 1. *Let*

$$\widehat{\boldsymbol{\theta}}_k^+:=\widehat{\boldsymbol{\theta}}_{k-1}^+-\gamma\widehat{\boldsymbol{h}}_k,\quad\gamma>0,\tag{2}$$

where $(\widehat{\boldsymbol{h}}_k)_i:=h(\widehat{\boldsymbol{\nu}}_k;\boldsymbol{x}_i)$ *is a tree approximation of* $\nabla\mathcal{L}(\widehat{\boldsymbol{\theta}}_{k-1}^+)$. *Assume that there exists a vector* $\widetilde{\boldsymbol{\theta}}^+$, *such that* $\nabla\mathcal{L}(\widetilde{\boldsymbol{\theta}}^+)=\mathbf{0}$. *It then holds that* (2) *generates a sequence of* $\widehat{\boldsymbol{\theta}}_k^+$*s such that*

$$\|\widehat{\boldsymbol{\theta}}_{k+1}^+-\widetilde{\boldsymbol{\theta}}^+\|\leq\|\widehat{\boldsymbol{\theta}}_k^+-\widetilde{\boldsymbol{\theta}}^+\|+\gamma\|\widehat{\boldsymbol{h}}_{k+1}\|,$$

where $\|\cdot\|$ *denotes the Euclidean norm, and*

$$\sum_{k\geq0}\|\widehat{\boldsymbol{\theta}}_k^+-\widetilde{\boldsymbol{\theta}}^+\|<+\infty\quad\textit{if}\quad\sum_{k\geq0}\gamma\|\widehat{\boldsymbol{h}}_{k+1}\|<+\infty.$$

That is, given that the norm of the tree-approximations converge to 0, then

$$(\widehat{\boldsymbol{\theta}}_k^+)_i:=\widehat{\theta}_k(\boldsymbol{x}_i)\to\widetilde{\theta}(\boldsymbol{x}_i),\quad\textit{as } k\to\infty,\ \textit{where } i=1,\ldots,n.$$

The proof of Lemma 1 follows directly from the triangle inequality and an application of Combettes (2001, Lemma 3.1).

Remark 2.

(a) From Proposition 1 it follows that the gradient approximations converge, and that the conclusion of Lemma 1 holds for Algorithm 1.

(b) *Note that the recursive formulation of* (2) *will coincide with standard gradient descent if one uses a tree with n terminal nodes in each iteration. This means that all observations could be assigned a unique value, which together with the assumed convexity satisfies the condition in Lemma 1 for Algorithm 1. This is, however, not a particularly interesting situation.*
(c) *Using the same setup as in Lemma 1, using stacked* $\boldsymbol{\theta}^+$ *vectors, it is straight forward to obtain bounds on the loss improvement by modifying the steps for standard gradient descent, see e.g. Nesterov (2018), if one in addition assumes Lipschitz bounds on the gradients.*
(d) *Note that Lemma 1 does not rely explicitly on that the loss function is convex.*
(e) *Proposition 1 and Lemma 1 only tells us that if the GBM converges in-sample, then it has found the optimal parameter values based on the unique values of the observed covariates.*

Concerning the corresponding results for the situation with a d-parametric $\boldsymbol{\theta}$ and Algorithm 2 the situation becomes more complicated. Even if $\mathcal{L}(y;\boldsymbol{\theta}), \boldsymbol{\theta} \in \mathbb{R}^d$, is convex in $\boldsymbol{\theta}$, implicating a unique minimum $\boldsymbol{\theta}^*$ based on a single observation y, it may be problematic to assume that $\mathcal{L}(y;\boldsymbol{\theta}^*) > -\infty$. As an example, this condition is not satisfied for, e.g., a Gaussian distribution with unknown mean and variance parameters. A situation where the above argumentation works in the d-parametric setting is if one assumes a discrete finite covariate space and a sufficiently large n: Let

$$\mathcal{X}^{(n)} := \{\widetilde{\boldsymbol{x}} \in \mathcal{X} : \widetilde{\boldsymbol{x}} = \boldsymbol{x}_i, \text{ for some } i, i = 1, \ldots, n\} = \{\widetilde{\boldsymbol{x}}_1, \ldots, \widetilde{\boldsymbol{x}}_{m^{(n)}}\},$$

where $m^{(n)} := |\mathcal{X}^{(n)}| \leq n$, and let

$$\boldsymbol{\theta}_j^{*,(n)} := \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{\arg\min} \sum_{i:\boldsymbol{x}_i=\widetilde{\boldsymbol{x}}_j} \mathcal{L}(y_i;\boldsymbol{\theta}).$$

Note that $\mathcal{X}^{(n)}$ corresponds to the finest resolution of the observed covariates that could ultimately be explored using the GBM. Note that given that n is sufficiently large it is clear that

$$\sum_{i:\boldsymbol{x}_i=\widetilde{\boldsymbol{x}}_j} \mathcal{L}(y_i;\boldsymbol{\theta}_j^{*,(n)}) > -\infty \tag{3}$$

holds for $j = 1, \ldots, m^{(n)}$, and it is possible to re-use the arguments underlying Proposition 1 for Algorithm 2 as well. That is, the above corresponds to assuming a sufficient number of unique y observations per $\widetilde{\boldsymbol{x}}_i$-value. Further, by using the resolution of the covariates given by $\mathcal{X}^{(n)}$ and only consider the overall loss per unique $\widetilde{\boldsymbol{x}}_j$, Lemma 1 can be applied cyclically.[1]

Remark 3.

(a) *The assumption of assuming a discrete finite covariate space is a common assumption in insurance pricing applications, and is, hence, not very restrictive.*
(b) *Note that the arguments leading up to* (3) *is in order to assure that there are a sufficient number of unique observations per* $\widetilde{\boldsymbol{x}}_i$ *in order for the loss to be bounded from below when evaluated in the corresponding optimal value.*

Continuing, Proposition 1 and Lemma 1 provides information about in-sample convergence, and illustrates that the method will converge to the finest partitioned model, given the observed data. The results do not, however, tell us anything about how the estimated function $\widehat{\theta}(\boldsymbol{x})$, or its multi-parameter analog $\widehat{\boldsymbol{\theta}}(\boldsymbol{x})$, performs on test data, when letting $n \to \infty$, or when using early stopping. One can, however, note that similar arguments as those that lead to Lemma 1 can be applied to the situation with unseen test data. That is, if we again focus on the univariate GBM, let $\boldsymbol{X}^{\text{test}}$ be a sample from the distribution of $\boldsymbol{X}$ not used in training, let $\widehat{\theta}_{k,n}^{\text{test}} := \widehat{\theta}_{k,n}(\boldsymbol{X}^{\text{test}})$, $\widehat{h}_{k,n}^{\text{test}} := h(\boldsymbol{X}^{\text{test}}; \widehat{\boldsymbol{\nu}}_{k,n})$, and let

$$\mathcal{L}^{\text{test}}(\eta) := \mathbb{E}[\mathcal{L}(Y;\eta(\boldsymbol{X}))].$$

[1] A more non-standard approach is to enforce this manually: take all training data $(y_i, \boldsymbol{x}_i)_{i=1}^n$ and duplicate adding small perturbations to the y_is according to $(y_i, y_i + \delta_i)_{i=1}^n$. This can be repeated until the loss evaluated in the $\boldsymbol{\theta}_i^*$s becomes finite.

Similar to the above, this leads to that adding an additional iteration to the approximation of the function θ based on $\boldsymbol{X}^{\text{test}}$, analogously to the iteration scheme (2), results in the inequality

$$|\widehat{\theta}^{\text{test}}_{k+1,n} - \widetilde{\theta}^{\text{test}}| \leq |\widehat{\theta}^{\text{test}}_{k,n} - \widetilde{\theta}^{\text{test}}| + \gamma|\widehat{h}^{\text{test}}_{k+1,n}|,$$

where $\widetilde{\theta}^{\text{test}} \in \arg\inf_{\theta\in\Theta} \mathcal{L}^{\text{test}}(\theta)$. Here one can note that in order for the convergence to the (local) minimum to hold, there is need of uniform convergence, since $\widehat{\theta}_{k,n}$ is trained on the in-sample loss $\mathcal{L}(\cdot) := \mathcal{L}_n(\cdot)$. Moreover, this also implies that $k \leq n$, which implies early stopping, see Zhang & Yu (2005) and Section 4 below. Concerning, the cyclic GBM, as before, the univariate arguments can be applied cyclically.

The above intuitive discussion based on Proposition 1 and Lemma 1 only provide general conditions for when convergence can be achieved, but state no practically implementable conditions relating specifically to Algorithms 1 and 2, and, ultimately, conditions on the tree-approximations and loss functions. For the univariate case this is analysed in detail assuming certain convex loss functions that are Lipschitz in specific senses using a general "greedy boosting algorithm" in Zhang & Yu (2005). Their analysis treats consistency and early stopping and is based on empirical process theory, but focuses mainly on classification problems, and their univariate arguments can be applied cyclically.

4. Hyperparameter tuning

GBMs are over-parametrised and hyperparameter tuning is necessary. The relevant hyper parameters are, apart from the ones connected to the regression trees, the shrinkage factors ϵ_j and the number of boosting steps κ_j. Finding appropriate values for these in the one-parametric setting can be done using so called *early stopping*; see e.g. Zhang & Yu (2005). This is done by fitting a GBM with a small ϵ to a training set until the boosting step where the loss on a validation set no longer improves by adding a new tree. Extending this to the setting with $d > 1$ comes with apparent issues, relating to that the complexity of the parameter function $\boldsymbol{\theta}(\boldsymbol{x})$ may vary between different dimensions. Hence, using the same shrinkage ϵ and stopping time κ for all dimension will likely result in a compromise between under- and overfitting in different dimensions. Due to this we suggest a simple approach where individual stopping times are obtained by looking at the loss improvement per dimension and boosting step, see Algorithm 3.

Remark 4.

(a) Note that the training procedure in Algorithm 3 first train all *dimensions for κ_0 iterations. This means that if the resulting κ_j^*s differ considerably between dimensions, some of the larger κ_j^*s will be influenced by dimensions that have been over-trained. Consequently, in this situation, one can consider adjusting the shrinkage parameters ϵ_j in order to push the κ_j^*s closer to each other.*

(b) There are other tuning procedures in the literature, see e.g. Thomas et al. (2018) and the references therein. A simple procedure, similar in spirit to Thomas et al. (2018), is the following:

- *In iteration (j, k), for each fold try to add a new tree and store the validation loss when doing so.*
- *Let $t_{j,k} \in \{0, 1\}$ indicate whether a tree should be added or not: If the fraction of folds where adding a tree improves the validation loss is greater than a pre-specified threshold value π, add a tree in all folds and calculate the cross-validation error when doing so and set $t_{j,k} = 1$; else, do not add a tree in any of the folds and calculate the cross-validation error when doing so and set $t_{j,k} = 0$.*
- *Iterate until all dimensions have been trained for κ_0 boosting steps.*
- *Let $\kappa^* := \min\{k, k = 1, \ldots, \kappa_0 : t_{j,k} = 0$ for all $j\}$*

This procedure will generate a sequence of $t_{j,k}$s that indicate whether or not to add a tree in each of the κ^* iterations, which will, at least partly, circumvent the problems of Remark 4(a). In the numerical examples the simpler Algorithm 3 will be used.

Algorithm 3 *Tuning procedure for the multi-parametric GBM*

Initialise: Let

- $(y_i, \boldsymbol{x}_i)_{i=1}^n$ be an i.i.d. sample from $Y|\boldsymbol{x} \sim F(\boldsymbol{\theta}(\boldsymbol{x}))$, $\boldsymbol{\theta}(\boldsymbol{x}) \in \mathbb{R}^d$,
- $\mathcal{L}(y_i; \boldsymbol{\theta})$ be the negative log-likelihood for distribution F,
- $\epsilon_j \in (0, 1], j = 1, \ldots, d.$ be shrinkage parameters,
- κ_0 be the number of boosting steps to be used for tuning,
- $(\mathcal{I}_{\text{v}}^{(l)})_{l=1}^K$ be a random partition of $\mathcal{I} = \{1, \ldots, n\}$, where $|\mathcal{I}_{\text{v}}^{(l)}| = [n/K]$, and let

$$\mathcal{I}_{\text{t}}^{(l)} = \mathcal{I} \setminus \mathcal{I}_{\text{v}}^{(l)}.$$

(i) **For** $l = 1, \ldots, K$ **do:**

Run Algorithm 2 using $(\boldsymbol{x}_i, y_i)_{i \in \mathcal{I}_{\text{t}}^{(l)}}$, $\epsilon_j, j = 1, \ldots, d$, and κ_0 to get $\widehat{\boldsymbol{\theta}}_{j,k}^{(l)}(\boldsymbol{x})$, $j = 1, \ldots, d$, $k = 1, \ldots, \kappa_0$.

End.

(ii) **For** $j = 1, \ldots, d$ **do:**

Get number of boosting steps

$$\kappa_j^* = \min \left\{ k, k = 1, \ldots, \kappa_0 : \sum_{l=1}^K \sum_{i \in \mathcal{I}_{\text{v}}^{(l)}} \mathcal{L}\left(y_i; \boldsymbol{\theta}_k^{(l,j)}(\boldsymbol{x})\right) \geq \sum_{l=1}^K \sum_{i \in \mathcal{I}_{\text{v}}^{(l)}} \mathcal{L}\left(y_i; \boldsymbol{\theta}_{k-1}^{(l,j)}\right) \right\},$$

where

$$\widehat{\boldsymbol{\theta}}_{k-1}^{(l,j)}(\boldsymbol{x}) := \begin{cases} \widehat{\boldsymbol{\theta}}_{d,k-1}^{(l)}(\boldsymbol{x}), & j = 1 \\ \widehat{\boldsymbol{\theta}}_{j-1,k}^{(l)}(\boldsymbol{x}), & j = 2, \ldots, d \end{cases}$$

End.

(iii) **Output:** $\boldsymbol{\kappa}^* := (\kappa_j^*)_{j=1}^d$

5. Numerical illustrations

The performance of the cyclical GBM method, henceforth referred to as CGBM, will be illustrated in a number of numerical examples. Regarding implementation, the individual trees are, in all examples, fitted using the `DecisionTreeRegressor` function of the `sklearn` package in `Python` where individual node values are optimised using `scipy`'s function `minimize`, and the resulting set of trees is used as the CGBM. The hyperparameters maximal tree depth and number of data points per node have been set to reasonable, but fixed values. Algorithm 3 is used to find dimension dependent number of trees, κ_j^*, using 10-fold cross validation. Further, 20% of the original data is saved for proper out-of-sample testing.

The CGBM's performance will be compared with the following benchmark models for $d = 2$:

Intercept an intercept only model.

CGLM a GLM fitted to both parameters using cyclical gradient descent. This corresponds to assuming linear regression functions, using suitable link functions, in each parameter dimension, and updating the resulting gradients cyclically. This differs from a DGLM, since no saddle point approximation of the loss function is used.

GBM a univariate GBM model for the mean function assuming an auxiliary constant dispersion parameter. Both parameter dimensions are initiated in the constant MLE, but only one of the dimensions is fitted using a GBM.

CGBM the cyclical GBM presented in this paper.

Note, however, below we also have a $d = 3$ example, a multivariate Gaussian distribution, but this is only used to illustrate that the method works for situations with $d > 2$, and no in-depth assessment is carried out.

In the real data examples where we have a low signal to noise ratio, we do not expect to see any dramatic improvement in the global loss on test data. In order to visualise *local* model performance, two different types of plots will be used: (i) binned response plots, (ii) concentration curves. In order to be able to describe the different plots, it will throughout the numerical illustrations be

assumed that

$$\mathrm{E}[Y \mid \boldsymbol{X} = \boldsymbol{x}, W = w] := w\mu(\boldsymbol{x}), \text{ and } \mathrm{Var}(Y \mid \boldsymbol{X} = \boldsymbol{x}, W = w) := w\sigma^2(\boldsymbol{x}),$$

where W corresponds to either policy duration (claim counts) or number of claims (total claim payment). For simulated data we set $W \equiv 1$.

The binned response plots are constructed as follows: Let $\widehat{\mu}_i := \widehat{\mu}(\boldsymbol{x}_i), i = 1, \ldots, n$, and let π_i be such that $\widehat{\mu}_{\pi_i} = \widehat{\mu}_{(i)}$, where $\widehat{\mu}_{(i)}$ corresponds to the ith smallest $\widehat{\mu}_i$-value. This allows us to construct $(y_{\pi_i}, w_{\pi_i}, \widehat{\mu}_{\pi_i})_i$ ordered according to the $\widehat{\mu}$-predictions. Next, split the data into κ bins and introduce

$$\mathcal{I}_j := \left\{ i = 1, \ldots, n : \left\lfloor \frac{(j-1)n}{\kappa} \right\rfloor + 1 \leq i \leq \left\lfloor \frac{jn}{\kappa} \right\rfloor \right\}, \ j = 1, \ldots, \kappa,$$

and define

$$\bar{y}_j := \frac{1}{\sum_{i \in \mathcal{I}_j} w_{\pi_i}} \sum_{i \in \mathcal{I}_j} y_{\pi_i}, \tag{4}$$

together with

$$\widehat{\bar{\mu}}_j := \frac{1}{\sum_{i \in \mathcal{I}_j} w_{\pi_i}} \sum_{i \in \mathcal{I}_j} w_{\pi_i} \widehat{\mu}_{\pi_i}, \text{ and } \quad \widehat{\bar{\sigma}}_j^2 := \frac{1}{(\sum_{i \in \mathcal{I}_j} w_{\pi_i})^2} \sum_{i \in \mathcal{I}_j} w_{\pi_i} \widehat{\sigma}_{\pi_i}^2. \tag{5}$$

The binned response plots are then given by plotting $(\bar{y}_j, \widehat{\bar{\mu}}_j)_j$ together with adding $\pm$ the predicted within-bin standard deviations $\widehat{\bar{\sigma}}_j$. This is similar to what is done in Lindholm et al. (2022).

Regarding the concentration curves, these are used in insurance applications in e.g. Denuit et al. (2019) and Wüthrich (2023). In the present paper we will calculate concentration curves only in the real data examples, and analogously as for the binned response plots, risk orderings will be based on the standardised $\widehat{\mu}$-predictions. That is, following Eq. (3.1) in Denuit et al. (2019) we define the concentration curve as follows:

$$\mathrm{CC}(Y, \widehat{\mu}(\boldsymbol{X}); \alpha) := \frac{\mathrm{E}[Y 1_{\{\widehat{\mu}(\boldsymbol{X}) \leq F_{\widehat{\mu}}^{-1}(\alpha)\}}]}{\mathrm{E}[Y]}, \ \alpha \in [0, 1], \tag{6}$$

where $F_{\widehat{\mu}}^{-1}(\alpha)$ corresponds to the $100\alpha\%$-percentile of $\widehat{\mu}(\boldsymbol{X})$. In the real data illustrations all predictions have been adjusted to be globally unbiased, and the expected values in Eq. (6) will be replaced with their empirical counterparts.

Further, the standard usage of concentration curves focuses on the performance of mean predictions. Since we are assessing multi-parametric models it is also of interest to assess, e.g. variance predictions. This can be done by changing Y to the corresponding squared residuals (treating W as known). When calculating concentration curves for variance predictions, the expected values needed for centering will be replaced with the corresponding model predictions, and the model which is being compared with the CGBM will be used for centering (i.e. its $\widehat{\mu}$ will be used). This ought to be disadvantageous for the performance evaluation of the CGBM. We will come back to this in the numerical real data illustrations below.

5.1. **Tuning procedure example.** In order to illustrate the tuning procedure described in Algorithm 3, consider the following stylised situation: Sample 100,000 data points from a Gaussian distribution with mean function $\mu(\boldsymbol{x}) := 0$ (dimension 1) and variance function $\sigma^2(\boldsymbol{x})$ (dimension 2), where $\boldsymbol{x} \in \mathbb{R}^5$. That is, the optimal number of trees in dimension 1 is given by $\kappa_1^* = 0$.

As an alternative to Algorithm 3, we use the following naive early stopping procedure:

$$\kappa^* = \min \left\{ k, k = 1, \ldots, \kappa_0 : \sum_{l=1}^{K} \sum_{i \in \mathcal{I}_{\mathrm{v}}^{(l)}} \mathcal{L}\left(y_i; \boldsymbol{\theta}_k^{(l,d)}(\boldsymbol{x})\right) \geq \sum_{l=1}^{K} \sum_{i \in \mathcal{I}_{\mathrm{v}}^{(l)}} \mathcal{L}\left(y_i; \boldsymbol{\theta}_{k-1}^{(l,d)}\right) \right\}.$$

That is, training is evaluated after full parameter vector updates, and the training is stopped the first time a full parameter vector update does not improve the cross-validation error. In this example, this means that $\kappa_1^* = \kappa_2^* = \kappa^*$. For our simulated data this results in $\kappa^* = 192$. When instead using Algorithm 3 on the same data this results in $\kappa_1^* = 0$ and $\kappa_2^* = 241$. Since data is simulated we can calculate the root mean squared error (RMSE) for the function estimates resulting

in 2.27 and 0.05 for the mean function using the naive tuning and Algorithm 3, respectively. For the variance function the RMSE becomes 2.19 and 1.64, respectively. This illustrates that the naive method needs to settle with a compromise where, in this situation, the mean function $\mu(\boldsymbol{x})$ has been over-trained, and the variance function $\sigma^2(\boldsymbol{x})$ has been under-trained. Since we use simulated data we know that the difference in κ_j^*s is expected, but also recall Remark 4.

5.2. **Simulated data.** We will illustrate the CGBM performance based on a number of simulated data sets using the distributions in Table 1. In all simulations we will use $n = 100,000$ i.i.d. sampled data points. The same simulated covariates will be used in all examples, where the covariates are simulated from an 8-dimensional standardised Gaussian distribution. The parameter functions that are used can be found in Table 3 in Appendix A. Note that not all features of $\boldsymbol{x}$ are used in all functions. Results in terms of average negative log-likelihood can be found in Table 1. It is apparent that the CGBM outperforms the other methods in terms of minimising the loss function. The univariate GBM, i.e. only modelling one of the parameter dimension using a GBM, the other set to a constant, performs particularly poorly on simulated data from the Beta Prime distribution, which can also be seen in the binned response plot in Figure 1. In Figure 1 all predictions have been ordered and split into $\sqrt{n}$ bins, where the corresponding observed bin averages are represented by dots and the model's average bin prediction is given by solid lines. The grey area represents the models' predicted within bin standard deviation. The poor performance of the univariate GBM in Figure 1 illustrates the dependence between the partial derivatives, something that is no problem for the CGBM. One can also note that that the CGBM performs well on Negative binomial distributed data despite not having a convex loss function.

As a final example with $d = 3$ parameters, we consider a multivariate Gaussian distribution with equal μ and σ for the two response dimensions, and correlation ρ. It is again seen, from Table 1, that the CGBM performs well.

Simulated data

Gamma	Train	3.15	3.21	3.16	3.19	**3.14**
	Test	3.13	3.20	3.14	3.18	**3.13**
Beta Prime	Train	−3.03	−2.81	−2.94	−2.99	**−3.04**
	Test	−3.01	−2.79	−2.93	−2.97	**−3.00**
Inverse Gaussian	Train	−0.16	−0.01	−0.04	−0.09	**−0.16**
	Test	−0.15	−0.00	−0.04	−0.08	**−0.15**
Gaussian	Train	2.37	3.16	2.78	3.14	**2.37**
	Test	2.38	3.18	2.79	3.16	**2.39**
Negative Binomial	Train	0.18	0.25	0.20	0.18	**0.18**
	Test	0.19	0.25	0.20	0.19	**0.19**
Multivariate Gaussian	Train	2.88	3.74	-	-	**2.88**
	Test	2.87	3.74	-	-	**2.88**
		True	Intercept	CGLM	GBM	CGBM

Real data (`freMTPL2`)

Number of claims	Train	-	0.21	0.21	0.20	**0.20**
(Negative binomial)	Test	-	0.21	0.21	0.20	**0.20**
Payment size	Train	-	1.25	1.22	1.25	**1.20**
(Gamma)	Test	-	1.24	1.22	1.24	**1.20**
		-	Intercept	CGLM	GBM	CGBM

TABLE 1. Average negative log-likelihood results for the numerical illustrations. The numbers in bold face indicate the best performing method.

5.3. **Real data.** The real data examples use the `freMTPL2` dataset from `CASdatasets`, consisting of rating factors, claim numbers, and payment information for third-party French motor liability policies. The dataset consists of a total of 678,013 contracts, 34,060 of which has 1 or more claims in the observed time period. We use a CGBM to model the number of claims (Negative binomial

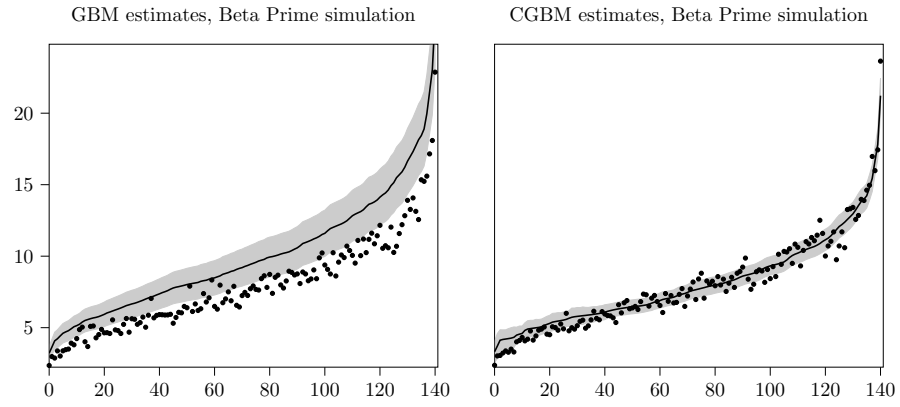

FIGURE 1. Simulated Beta Prime data. Binned response plot where the test dataset of size m has been split into $[\sqrt{m}]$ bins based on the model's sorted mean predictions using standardised exposure. Dots correspond to the observed bin averages according to (4). The thick black line represents the model's mean predictions per bin and the grey area corresponds to $\pm$ one standard deviation predicted by the model within each bin calculated according to (5).

distribution) and payment sizes (Gamma distribution) respectively, using contract duration and number of claims as observed exposures. In both cases, parameterisations such that both the mean and variance are assumed to vary linearly with the exposure are used, see e.g. Ohlsson & Johansson (2010). The tree depth was set to 2 and 3 for the number of claims and total payment data respectively, and the corresponding minimum number of datapoints per node was set to 5 for both datasets. The features used for the regression are presented in Table 4 in Appendix A.

The results for average overall negative log-likelihood are presented in Table 1. One can here note that there is only a modest improvement in overall loss, but this is to be expected, since there are only 5% claims (i.e. non-zero observations).

From the binned response plots in Figure 2 it is clear that both the GBM and CGBM work well for claim counts. For the total claim payment data, however, the signal is very weak and the mean predictor is essentially constant, although one can see more variation in the standard deviations for the CGBM. These observations are supported by the concentration curves based on the mean predictions, see Figure 3, where a slight improvement is seen for the count data CGBM, whereas the claim payment (C)GBM essentially coincides with the intercept model, corresponding to the 45 degree line. It is, however, interesting to see that the risk ordering based on variance predictions of the squared residuals finds a clear signal for the CGBM for both claim count and claim payment data. This illustrates a benefit of using the CGBM instead of the simpler GBM. Regarding the concentration curves based variance predictions, since the mean predictions are very similar between CGBM and GBM, the concentration curves for the variance predictions look very similar when basing the squared residuals instead on the CGBM's mean predictions. For more on the use of concentration curves when the predictors are not auto-calibrated, see Wüthrich (2023).

Again note that the CGBM works reasonably well also when using Negative binomial models with non-convex losses.

6. Comments on extensions and concluding remarks

The general multi-parametric method described in Algorithm 2 is intentionally simple. It is, however, straight forward to define, e.g. multi-parametric versions of XGBoost, which rely on a second order Taylor approximation of the empirical loss. This results in a tree fitting procedure closely related to a standard Newton step. Moreover, the consistency analysis from Section 3.1 applies to this model as well if one replaces the original loss with the corresponding Taylor approximated loss. It is, however, important to note that XGBoost uses Hessians, due to the second order approximation, and if the original loss is not convex, this must be done with care, since an

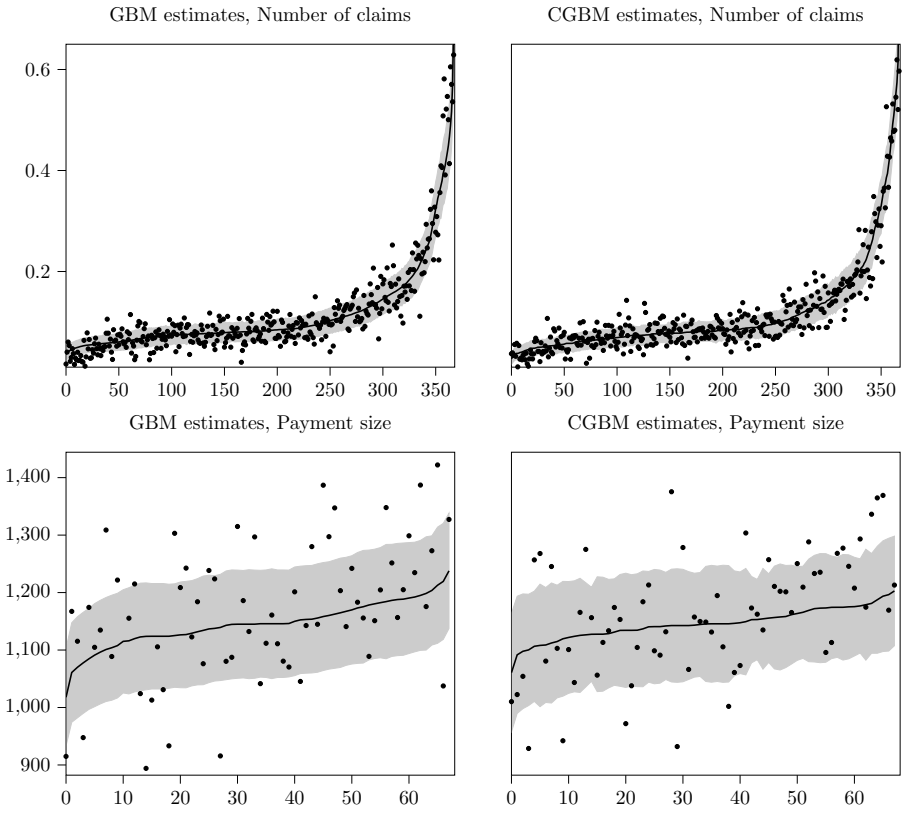

FIGURE 2. Claim count and total claim cost data from the the `freMTPL2` dataset. Binned response plot where the test dataset of size m has been split into $[\sqrt{m}]$ bins based on the model's sorted mean predictions using standardised exposure. Dots correspond to the observed bin averages according to (4). The thick black line represents the model's mean predictions per bin and the grey area corresponds to $\pm$ one standard deviation predicted by the model within each bin calculated according to (5).

indefinite Hessian may change the direction of the search. Suggestions on how to deal with this is discussed in e.g. Lee (2020, 2021) for the closely related Asymptotic Delta boosting method. Other alternatives to circumvent this could be to consider quasi-Newton methods using expected Hessians. Note that this will not be an issue when using Algorithm 2, since each iteration will always result in a loss improvement. Further, by not requiring explicit Hessian calculations Algorithm 2 will be easier to implement compared to e.g. multi-parametric XGBoost or Asymptotic Delta-boost.

Moreover, in Section 5 Algorithm 2 was applied to multivariate response data, here simple multivariate Gaussian data, but Algorithm 2 could of, course be applied to more complex multivariate distributions.

Concerning non-convexity of the loss function, if the loss function is non-convex there are no guarantees for convergence. In this situation one needs to restort to massive grid searches, which is computationally expensive, see e.g. Thomas et al. (2018) and the references therein. Note, however, that this can be less of a problem if one is primarily interested in modelling, e.g. moments, since even if there are multiple minima w.r.t. $\boldsymbol{\theta}(\boldsymbol{x})$, moments can still be stable.

The definition of Algorithm 2 is intended to be simple and generic. One can, however, note that for distributions, such as the Gamma distribution, the line search in the node adjustment in Step (iii) in Algorithm 2 can be replaced with closed form expressions. This will improve speed. Another alternative is to merely ignore the node adjustment and use smaller shrinkage parameters

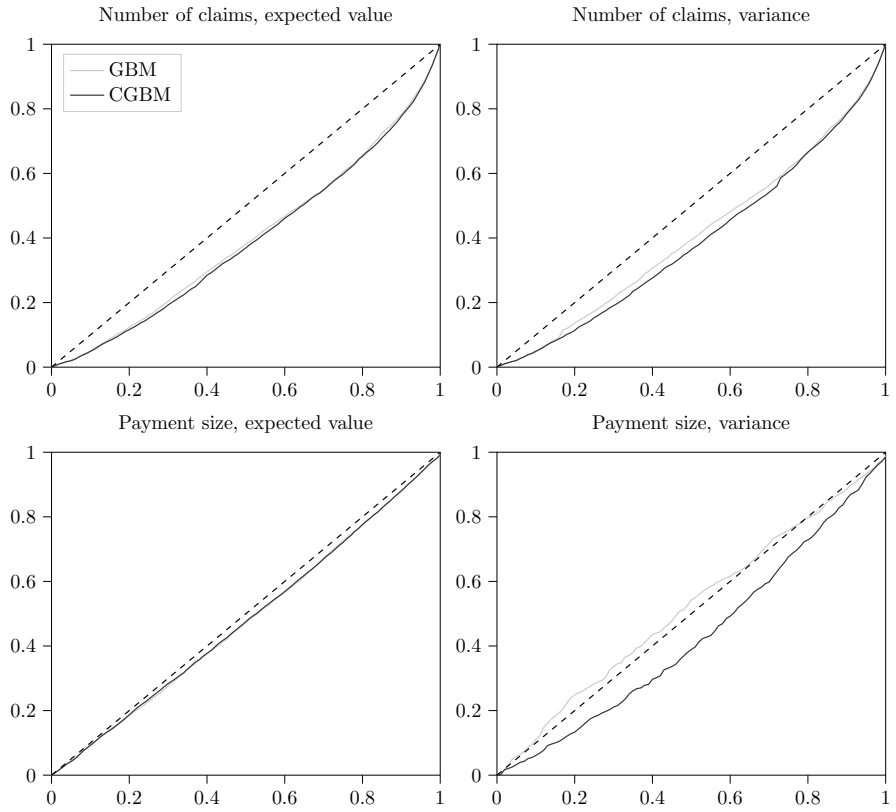

FIGURE 3. Concentration curves for the test data of the `freMTPL2` dataset, for mean and variance predictions, calculated according to (6). The variance plots contain squared residuals based on the GBM estimates of the mean, sorted by the corresponding models' variance estimates.

(ϵ_js). Concerning approximations that may improve speed; see e.g. Chen & Guestrin (2016) and Ke et al. (2017).

As a final comment, Algorithm 2 produces parameter dimension dependent GBMs, where each parameter dimension may depend differently on the covariates. This construction makes it possible to measure covariate importance using e.g. Shapley values, see e.g. Lundberg & Lee (2017), or feature importance, see Friedman (2001) to explain the covariate influence in each parameter dimension.

ACKNOWLEDGEMENTS

The authors would like to thank M.V. Wüthrich for insightful comments on an earlier draft. M. Lindholm gratefully acknowledges financial support from the Länsförsakringar Alliance project P9/20 "Machine learning methods in non-life insurance".

REFERENCES

Barndorff-Nielsen, O. E. (1978), *Information and exponential families in statistical theory*, Chichester, John Wiley & Sons.

Barndorff-Nielsen, O. E. (1988), *Parametric statistical models and likelihood*, New York, Springer.

Bazaraa, M. S., Sherali, H. D. & Shetty, C. M. (2006), *Nonlinear programming: theory and algorithms*, New York, John Wiley & Sons.

Breiman, L., Friedman, J. H., Olshen, R. A. & Stone, C. J. (1984), *Classification and regression trees*, Belmont, Calif. Wadsworth Inc.

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.

Combettes, P. L. (2001), Quasi-Fejérian analysis of some optimization algorithms, *in* 'Studies in Computational Mathematics', Vol. 8, Elsevier, pp. 115–152.

Denuit, M., Sznajder, D. & Trufin, J. (2019), 'Model selection based on lorenz and concentration curves, gini indices and convex order', *Insurance: Mathematics and Economics* **89**, 128–139.

Friedman, J. H. (2001), 'Greedy function approximation: a gradient boosting machine', *Annals of statistics* **25**(5), 1189–1232.

Friedman, J. H. & Popescu, B. E. (2008), 'Predictive learning via rule ensembles', *The Annals of Applied Statistics* **2**(3), 916–954.

Jørgensen, B. (1997), *The theory of dispersion models*, London, Chapman & Hall.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. & Liu, T.-Y. (2017), 'LightGBM: A highly efficient gradient boosting decision tree', *Advances in neural information processing systems* **30**.

Lee, S. C. (2020), 'Delta boosting implementation of negative binomial regression in actuarial pricing', *Risks* **8**(1), 19.

Lee, S. C. (2021), 'Addressing imbalanced insurance data through zero-inflated Poisson regression with boosting', *ASTIN Bulletin: The Journal of the IAA* **51**(1), 27–55.

Lindholm, M., Lindskog, F. & Palmquist, J. (2022), 'Local bias adjustment, duration-weighted probabilities, and automatic construction of tariff cells', *(To appear) Scandinavian Actuarial Journal* .

Luenberger, D. G. (2016), *Linear and nonlinear programming*, 4th edn, New York, Springer.

Lundberg, S. M. & Lee, S.-I. (2017), 'A unified approach to interpreting model predictions', *Advances in Neural Information Processing Systems* **30**.

Mayr, A., Fenske, N., Hofner, B., Kneib, T. & Schmid, M. (2012), 'Generalized additive models for location, scale and shape for high dimensional data – a flexible approach based on boosting', *Journal of the Royal Statistical Society: Series C (Applied Statistics)* **61**(3), 403–427.

Meng, S., Gao, Y. & Huang, Y. (2022), 'Actuarial intelligence in auto insurance: Claim frequency modeling with driving behavior features and improved boosted trees', *Insurance: Mathematics and Economics* **106**, 115–127.

Nesterov, Y. (2018), *Lectures on convex optimization*, Boston, Mass. Kluwer Acad.

Ohlsson, E. & Johansson, B. (2010), *Non-life insurance pricing with generalized linear models*, EAA lecture notes, Berlin, Springer.

Rigby, R. A. & Stasinopoulos, D. M. (2005), 'Generalized additive models for location, scale and shape', *Journal of the Royal Statistical Society: Series C (Applied Statistics)* **54**(3), 507–554.

Saha, A. & Tewari, A. (2013), 'On the nonasymptotic convergence of cyclic coordinate descent methods', *SIAM Journal on Optimization* **23**(1), 576–601.

Sigrist, F. (2021), 'Gradient and Newton boosting for classification and regression', *Expert Systems With Applications* **167**, 114080.

Smyth, G. K. (1989), 'Generalized linear models with varying dispersion', *Journal of the Royal Statistical Society: Series B (Methodological)* **51**(1), 47–60.

Thomas, J., Mayr, A., Bischl, B., Schmid, M., Smith, A. & Hofner, B. (2018), 'Gradient boosting for distributional regression: faster tuning and improved variable selection via noncyclical updates', *Statistics and Computing* **28**(3), 673–687.

Wüthrich, M. V. (2023), 'Model selection with gini indices under auto-calibration', *European Actuarial Journal* pp. 1–9.

Zhang, T. & Yu, B. (2005), 'Boosting with early stopping: Convergence and consistency', *The Annals of Statistics* **33**(4), 1538–1579.

TABLE 2. Parameter functions for the simulated data

Distribution	Parameter	Functional form
Gamma	μ	$\exp\left\{a_1x_1 + a_2x_2^2 + a_3\lvert x_3\rvert \sin\left(a_4x_3\right) + a_5x_4x_5 + a_6x_5^2x_6\right\}$
	ϕ	$\exp\left\{b_0 + b_1x_2 + b_2\lvert x_1\rvert\right\}$
Beta Prime	μ	$\exp\left\{a_0 + a_1\sin\left(a_2x_2\right) + a_3x_3x_4\right\}$
	ν	$\exp\left\{b_0 + b_1x_2 + b_2\min\left(b_3, b_4\lvert x_3\rvert\right) + b_5x_6x_3\right\}$
Inverse Gaussian	μ	$\exp\left\{a_1x_1x_2x_3 + a_2\lvert x_4\rvert\right\}$
	λ	$\exp\left\{b_0 + b_1 \cdot 1_{\{x_1>0\}} + b_2x_2 \cdot 1_{\{x_5>0\}}\right\}$
Gaussian	μ	$a_1x_1 + a_2x_2^2 + a_3\lvert x_3\rvert \sin\left(a_4x_2\right) + a_5x_4x_5^2$
	σ^2	$\exp\left\{b_0 + b_1x_2 + b_2\lvert x_1\rvert \sin\left(b_3x_2\right) + b_4 \cdot 1_{\{x_2>0\}}\right\}$
Negative Binomial	μ	$\exp\left\{a_0 + a_1\min\left(a_2, x_5\right)^2 + a_3\min\left(a_4, x_2\right) + a_5\sin\left(a_6x_3\right)\right\}$
	θ	$\exp\left\{b_0 + b_1 \cdot 1_{\{x_2>0\}} + b_2\lvert x_3\rvert \cdot 1_{\{x_6>0\}} + b_3x_4\right\}$
Multivariate Gaussian	μ	$a_1x_1 + a_2x_2^2 + a_3\lvert x_3\rvert \sin\left(a_4x_2\right) + a_5x_4x_5^2$
	σ^2	$\exp\left\{b_0 + b_1x_2 + b_2\lvert x_1\rvert \sin\left(b_3x_2\right) + b_4 \cdot 1_{\{x_2>0\}}\right\}$
	ρ	$\text{sigm}\left\{c_0 + c_1x_4 + c_2x_3^2\right\}$

TABLE 3. Parameter functions for the simulated data in the numerical illustration

Feature	Description	Type
Brand	Brand of car	Categorical (7)
Gas	Gas used by car	Categorical (2)
Density	Population density in car-owners city	Continuous
Area	Area of car	Categorical
Region	Region of car	Categorical
BonusMalus	Bonus/Malus level of driver	Continuous
Power	Power level of car	Ordinal (12)
Vehicle age	Age of the car in years	Continuous
Driver age	Age of driver in years	Continuous

TABLE 4. Features used in the real data example.

APPENDIX A. TABLES

ŁUKASZ DELONG
SGH WARSAW SCHOOL OF ECONOMICS
INSTITUTE OF ECONOMETRICS
LUKASZ.DELONG@SGH.WAW.PL

MATHIAS LINDHOLM
DEPARTMENT OF MATHEMATICS
STOCKHOLM UNIVERSITY
LINDHOLM@MATH.SU.SE

HENNING ZAKRISSON
DEPARTMENT OF MATHEMATICS
STOCKHOLM UNIVERSITY
ZAKRISSON@MATH.SU.SE