Latent Gaussian Model Boosting

Fabio Sigrist

Abstract—Latent Gaussian models and boosting are widely used techniques in statistics and machine learning. Tree-boosting shows excellent prediction accuracy on many data sets, but potential drawbacks are that it assumes conditional independence of samples, produces discontinuous predictions for, e.g., spatial data, and it can have difficulty with high-cardinality categorical variables. Latent Gaussian models, such as Gaussian process and grouped random effects models, are flexible prior models which explicitly model dependence among samples and which allow for efficient learning of predictor functions and for making probabilistic predictions. However, existing latent Gaussian models usually assume either a zero or a linear prior mean function which can be an unrealistic assumption. This article introduces a novel approach that combines boosting and latent Gaussian models to remedy the above-mentioned drawbacks and to leverage the advantages of both techniques. We obtain increased prediction accuracy compared to existing approaches in both simulated and real-world data experiments.

Index Terms—Machine learning, boosting, mixed effects models, gaussian processes

1 Introduction

Boosting [1], [2] is a machine learning technique that achieves state-of-the-art prediction accuracy [3], [4]. This is reflected in statements such as "[i]n general 'boosted decision trees' is regarded as the most effective off-the-shelf non-linear learning method for a wide range of application problems" [5]. In boosting, and in many other supervised machine learning algorithms, it is assumed that a potentially complex predictor function $F(\cdot)$ relates a set of predictor variables to a response variable, and that conditional on $F(\cdot)$ evaluated at the predictor variables, different samples are independent. Apart from this potentially unrealistic independence assumption, tree-boosting can have difficulty with high-cardinality categorical variables, and it produces discontinuous predictions. The latter is often unrealistic for spatial and spatio-temporal data.

Latent Gaussian models are a broad class of flexible prior models in which, conditional on latent Gaussian variables, a response variable is assumed to follow a known parametric distribution, and parameters of this distribution are related to the latent Gaussian variables. Two widely known types of latent Gaussian models are Gaussian process [6] and grouped, or clustered, random effects models [7]. Gaussian process models are used for modeling, for instance, time series, spatial, and spatio-temporal data. Further, grouped random effects models are used for modeling data with a grouping structure. In particular, grouped random effects models can be seen as an approach for modeling categorical variables with possibly high-cardinality, as every categorical variable corresponds to a grouping and vice versa. Latent Gaussian models have the advantage that they are probabilistic models which allows for making probabilistic predictions. Besides, the explicit modeling of dependence allows for efficient learning of the predictor function $F(\cdot)$. A drawback of existing latent Gaussian models is that the prior mean is often assumed to be either zero or to be a linear function of predictor variables. Both the zero-mean and the linearity assumption can be unrealistic, and higher prediction accuracy can be obtained by relaxing these assumptions; see, e.g., our experiments in Sections 4 and 5.

The goal of this article is to combine boosting and latent Gaussian models for non-Gaussian data distributions. Specifically, we consider a class of models where the response variable follows a known parametric distribution, and a parameter of this distribution is related to the sum of a non-parametric function and a latent Gaussian variable. We propose to model the predictor function $F(\cdot)$ by an ensemble of base learners, such as regression trees [8], learned in a stage-wise manner by doing functional gradient descent steps in a boosting framework, and the hyperparameters of the covariance structure of the latent Gaussian model are jointly estimated with the predictor function; see Section 3 for more details.

Our novel approach allows for relaxing both the independence assumption in boosting and the linearity assumption in latent Gaussian models in a flexible non-parametric way. Further, it allows for obtaining continuous, or smooth, predictions for predictor variables such as spatial coordinates while at the same time being able to capture non-linearities, discontinuities, and interactions for predictor variables for which this is desirable. In addition, the use of grouped random effects is as a way for dealing with high-cardinality categorical variables in tree-boosting. As we show in our experiments in Sections 4 and 5, our novel approach leads to higher prediction accuracy compared to both existing boosting algorithms and linear latent Gaussian models.

1.1 The View of Latent Gaussian Models as Priors and Regularizers

An algorithm for learning a predictor function $F(\cdot)$, which relates predictor variables to a response variable, should

• *The author is with the Lucerne University of Applied Sciences and Arts, 6343 Rotkreuz, Switzerland. E-mail: fabio.sigrist@hslu.ch.*

Manuscript received 14 May 2021; revised 24 December 2021; accepted 9 April 2022. Date of publication 19 April 2022; date of current version 6 January 2023. This work was supported in part by the Swiss Innovation Agency - Innosuisse under Grant 55463.1 IP-IC.
(Corresponding author: Fabio Sigrist.)
Recommended for acceptance by S. Zilles.
Digital Object Identifier no. 10.1109/TPAMI.2022.3168152

result in an estimate $\hat{F}(\cdot)$ that has both low bias and low variance. Intuitively, a low bias estimator $\hat{F}(\cdot)$, such as a flexible machine learning model, can have high variance if the complexity of the function $F(\cdot)$ is large relative to the sample size. Examples of data for which this can occur include, first, time series, spatial, and spatio-temporal data where the amount of variation over space and/or time is large relative to the sample size and, second, data with high-cardinality categorical variables where the number of categories is large relative to the sample size.

Modern supervised machine learning approaches such as deep neural networks and tree-boosting typically have low bias but need to apply some form of regularization to avoid high variance in $\hat{F}(\cdot)$. General-purpose regularization options for boosting include early stopping, learning rate shrinkage, and restrictions on the base learners such as the depth of trees and the minimal number of samples per leaf. However, as we argue in this article, for applications involving, e.g., spatial data or high-cardinality categorical variables, it can be advantageous to apply problem-specific regularization which incorporates available prior knowledge instead of relying on agnostic general-purpose regularization.

Prior models such as latent Gaussian models which explicitly model residual dependence among data can be interpreted as applying a form of regularization. For instance, an important prior assumption of Gaussian processes is that observations that are close together in space and/or time, or any other feature that defines a Gaussian process, are "more similar to each other than distant samples". For spatial data, this prior assumption is often referred to as Tobler's first law of geography [9]. Such a prior model implies regularization in the sense that predictions for points that are close together are similar, and that the amount of similarity varies in a continuous, or potentially smooth, manner with distance. Further, heuristically, a prior assumption of grouped random effects models is that different group effects are similar to some degree, and deviations from a global average are stochastic and identically distributed. Crucially, important characteristics of a latent Gaussian model such as the speed at which the dependency decays over space and/or time, the smoothness, the amount of variation over space and/or categories, and thus the amount of regularization implied by the prior is characterized by hyperparameters which can be learned from data. Our proposed approach allows for incorporating this reasonable prior knowledge and thus for applying explicit data-specific regularization in boosting algorithms.

Intuitively, we conjecture that the improvement in prediction accuracy of our novel approach over classical independent tree-boosting is the larger, the more categories a categorical variable has and the faster the covariance decays over space and/or time or, in other words, the higher the complexity of $F(\cdot)$ is compared to the sample size since appropriate regularization is more important in these cases. This hypothesis is confirmed in simulated experiments in Section 4.1.

1.2 Related Work

For Gaussian data, existing approaches for combining Gaussian process and grouped random effects models with machine learning algorithms include Hajjem *et al.* [10], Sela and Simonoff [11], Fu and Simonoff [12], Hajjem *et al.* [13], Sigrist [14], Griesbach *et al.* [15], Rabinowicz and Rosset [16], and Saha *et al.* [17].

For non-Gaussian data, there exists little prior work on combining non-linear machine learning methods with latent Gaussian models. For the special case of grouped random effects, Hajjem *et al.* [18], Fokkema *et al.* [19], and Speiser *et al.* [20] propose algorithms that use regression trees to model the function $F(\cdot)$. Speiser *et al.* [21] and Pellagatti *et al.* [22] extend these algorithms by replacing trees with random forests. However, all of these methods are heuristically motivated. In particular, it is unclear which objective functions these algorithms minimize – they do not maximize a marginal or approximate marginal log-likelihood neither in a component-wise way nor using an EM algorithm – and whether and to which values they converge.

A straightforward alternative to the use of Gaussian processes and grouped random effects is to simply include the variables that define the latent Gaussian model, such as spatial coordinates, time points, and categorical variables, in the deterministic predictor function $F(\cdot)$ of a statistical or machine learning model. A special example of this is the approach Hothorn *et al.* [23] where splines are used to model spatial effects and ridge regression is used to model grouped random effects. However, while the adoption of splines avoids discontinuities in predictions, this approach has several drawbacks compared to using latent Gaussian models. First, the hyperparameters, and thus the amount of regularization or smoothing, cannot be learned from data and need to be chosen using, e.g., cross-validation and, second, since the base learners are deterministic, probabilistic predictions cannot be made. Further, splines have the disadvantage that they suffer from the so-called "curse of dimensionality" when the dimension of the "locations" is large and the locations are thus sparse in space. This approach can thus not be used in situations where Gaussian processes are applied to higher-dimensional non-spatial "locations" as is often done in machine learning applications of Gaussian processes.

The linearity assumption in mixed effects models can also be relaxed by using splines or generalized additive models [24], [25] for modeling the predictor function $F(\cdot)$; see, e.g., Tutz and Reithinger [26] and Groll and Tutz [27]. However, one has to assume a certain functional form with only limited possibility for interaction effects for the predictor function by specifying, for instance, main and second-order interaction effects. In general, this can thus result in model misspecification.

2 A Non-parametric Latent Gaussian Model

We assume that the response variable $y = (y_1, \ldots, y_n)^T \in \mathbb{R}^n$ follows a parametric distribution which has a density $p(y|\mu, \xi)$ with respect to a sigma finite product measure with parameters $\mu \in \mathbb{R}^n$ and $\xi \in \Xi \subset \mathbb{R}^r$. The focus of this article is on non-Gaussian densities $p(y|\mu, \xi)$. If $p(y|\mu, \xi)$ is a Gaussian density, calculations simplify as the required marginalization can be done analytically; see Sigrist [14]. Examples of $p(y|\mu, \xi)$ include Bernoulli and Poisson densities for binary classification and Poisson regression. The parameter

μ is related to the sum of a predictor function $F(\cdot)$ evaluated at predictor variables and a latent Gaussian variable:

$$\mu = F(X) + Zb, \quad b \sim \mathcal{N}(0, \Sigma), \tag{1}$$

where $F(X)$ is the row-wise evaluation of a function $F(\cdot) : \mathbb{R}^p \to \mathbb{R}$, $F(X) = (F(X_1), \ldots, F(X_n))^T$, and $X_i = (X_{i1} \ldots, X_{ip})^T \in \mathbb{R}^p$ is the ith row of $X \in \mathbb{R}^{n \times p}$ containing predictor variables for observation i, $i = 1, \ldots, n$. For notational simplicity, we assume that the distribution $p(y|\mu, \xi)$ is parameterized in a way such that $\mu \in \mathbb{R}^n$. Otherwise, if the support of μ is not $\mathbb{R}^n$, the model needs to be reparametrized using, e.g., a so-called link function. Further, we assume that conditional on μ, the data is independent:

$$p(y|\mu, \xi) = \prod_{i=1}^{n} p(y_i|\mu_i, \xi).$$

Any additional, auxiliary or hyper-, parameters of the likelihood $p(y|\mu, \xi)$ are denoted by ξ. In many situations such as classification and Poisson regression, there are no additional parameters.

We assume that $F(\cdot)$ is a function in a function space $\mathcal{H}$ that is the linear span of a set $\mathcal{S}$ of so-called base learners $f_j(\cdot) : \mathbb{R}^p \to \mathbb{R}$. Classes of base learners include, e.g., linear functions [28], smoothing splines [29], wavelets [30], reproducing kernel Hilbert space (RKHS) regression functions [31], and regression trees [8], with the latter being the most popular choice. For defining functional derivatives, we additionally assume that the space $\mathcal{H}$ is normed. For instance, assuming that the X_i's are identically distributed and that all $F \in \mathcal{H}$ are square integrable with respect to the law of X_1, a norm on $\mathcal{H}$ can defined by the inner product $\langle F, G \rangle = E_{X_1}(F(X_1)G(X_1))$ for $F, G \in \mathcal{H}$.

Examples of latent Gaussian variables $b \in \mathbb{R}^m$ include finite-dimensional versions of Gaussian processes and/or grouped random effects. We assume that the covariance matrix $\mathrm{Cov}(b) = \Sigma$ is parametrized by a set of parameters $\theta \in \Theta \subset \mathbb{R}^q$ whose dimensionality is often relatively low, and Σ can depend on predictor variables $S \in \mathbb{R}^{n \times d}$. For instance, for spatial and temporal Gaussian processes, these predictor variables S are locations and time points, respectively. For notational simplicity, we suppress the dependence of Σ on its parameters θ and on S. Further, $Z \in \mathbb{R}^{n \times m}$ are predictor variables which relate the random variable b to μ. Often, Z is simply an incidence matrix with entries in $\{0, 1\}$. For instance, for grouped random effects, Z consists of dummy variables that encode categorical variables. In general, Z can also contain continuous predictor variables, e.g., in the case of random coefficient models [32]. Note that, conditional on $F(X)$ and Z, dependence among the response variable y can arise either due to the matrix Z being non-diagonal or due to the covariance matrix Σ being non-diagonal.

In summary, we distinguish between three sets of predictor variables: X with input variables for the predictor function $F(\cdot)$, S which determines the covariance structure of the random variable b, and Z which relates b to μ and thus also determines the covariance structure of μ and y. Note that these three sets of predictor variables may or may not be over-lapping. If e.g., X and S contain disjoint sets of predictor variables, one assumes that there are no interactions among them. On the other hand, if, for instance, spatial locations in S are also included in X, interactions among locations and other predictor variables in X can be modeled.

In comparison to our approach, existing boosting algorithms, and many supervised machine learning algorithms in general, do not distinguish between the different types of predictor variables X, S, and Z, and one essentially has two options: either ignore the additional predictor variables in S and Z or include them in the set of predictor variables X for the predictor function $F(\cdot)$. It goes without saying that the former is not a good option as potentially important information is neglected. Furthermore, the second option can result in the high variance problem mentioned in the introduction, and this translates into inferior prediction accuracy; see, e.g., our experiments in Sections 4 and 5. Besides, existing boosting algorithms assume that the data y is independent conditional on $F(X)$ and thus ignore any potential residual correlation. Further, in most latent Gaussian models, it is assumed that $F(\cdot)$ is either a linear function, $F(X) = X\beta$, or that $F(\cdot)$ is simply zero, $F(X) = 0$.

For notational simplicity, we assume that only one parameter μ of the data distribution $p(y|\mu, \xi)$ is related to a latent Gaussian variable. However, the extension to multivariate data and/or the situation where multiple parameters depend on potentially multiple Gaussian variables is straightforward. Also note that we assume that the latent variable b follows a Gaussian distribution, but moderate violations of this assumption have been shown to have only a small effect on prediction accuracy in the context of generalized linear mixed models [33].

2.1 Definition of Learners

The marginal density of the response y is given by

$$p(y|F, \theta, \xi) = \int p(y|\mu, \xi)p(b|\theta)db. \tag{2}$$

Ideally, we would like to minimize the empirical risk functional

$$R(F(\cdot), \theta, \xi) : \quad (F(\cdot), \theta, \xi) \;\mapsto\; -\log\left(p(y|F, \theta, \xi)\right)\Big|_{F=F(X)}.$$

If $p(y|\mu, \xi)$ is a Gaussian distribution, the marginalization in (2) can be done analytically. For non-Gaussian data, however, an approximation has to be used. In order that an approximation is applicable for the boosting algorithms presented in this article, it needs to fulfill two requirements. First, one must be able to compute it efficiently as this needs to be done repeatedly. Second, the gradient with respect to $F(\cdot)$ must be computable in an efficient way.

Our goal is thus to find the joint minimizer

$$(\hat{F}(\cdot), \hat{\theta}, \hat{\xi}) = \underset{(F(\cdot), \theta, \xi) \in (\mathcal{H}, \Theta, \Xi)}{\operatorname{argmin}} R^A(F(\cdot), \theta, \xi), \tag{3}$$

where $R^A(F(\cdot), \theta, \xi)$ is an approximate empirical risk functional

$$R^A(F(\cdot), \theta, \xi) : \quad (F(\cdot), \theta, \xi) \;\mapsto\; L^A(y|F, \theta, \xi)\Big|_{F=F(X)}, \tag{4}$$

and $L^A(y|F,\theta,\xi)$ is an approximation to the negative logarithmic marginal likelihood $-\log\left(p(y|F,\theta,\xi)\right)$. Note that $R^A(F(\cdot),\theta,\xi)$ is calculated by evaluating $F(\cdot)$ at X and then calculating $L^A(y|F=F(X),\theta,\xi)$. I.e., the risk functional $R^A(F(\cdot),\theta,\xi)$ is, in general, infinite dimensional in its first argument and finite dimensional in its other arguments.

2.2 The Laplace Approximation

In this article, we focus on the Laplace approximation [34] for approximating the marginal likelihood in (2). However, other approximations that satisfy the above-mentioned requirements can equally well be used. For instance, if $p(y|\mu,\xi)p(b|\theta)$ factors into low-dimensional components, numerical integration, such as adaptive Gauss-Hermite quadrature, can be used to approximate (2). Examples, where this applies, are single-level grouped random effects models. Another potential approximation is expectation propagation (EP) [35]. Depending on the data distribution, for instance, for binary classification, this can lead to more accurate approximations [36], but it is computationally more demanding than the Laplace approximation.

For applying the Laplace approximation, we assume that $p(y_i|\mu_i,\xi)$ is three times differentiable in μ_i. The Laplace approximation for (2) is given by

$$p(y|F,\theta,\xi) \approx p(y|\tilde{\mu},\xi)p(\tilde{b}|\theta) \cdot \det\Big(Z^T\tilde{W}Z+\Sigma^{-1}\Big)^{-1/2}(2\pi)^{m/2}, \tag{5}$$

where $\tilde{b}$ is the mode of $p(y|b,F,\xi)p(b|\theta)$,

$$\begin{aligned}\tilde{b} &= \underset{b}{\operatorname{argmax}}\ p(y|\mu,\xi)p(b|\theta)\\ &= \underset{b}{\operatorname{argmax}}\ \log p(y|\mu,\xi) - \frac{1}{2}b^T\Sigma^{-1}b,\end{aligned}$$

$\tilde{\mu} = F(X)+Z\tilde{b}$, and $\tilde{W}\in\mathbb{R}^{n\times n}$ is a diagonal matrix with entries

$$(\tilde{W})_{ii} = -\frac{\partial^2 \log p(y_i|\mu_i,\xi)}{\partial \mu_i^2}\bigg|_{\mu=\tilde{\mu}}.$$

Note that $\tilde{b}$ depends on $F=F(X)$, θ, and ξ, but we suppress this dependence for notational simplicity. The mode can be found, for instance, using Newton's method.

Modulo constant terms that do not depend on θ, ξ, or F, the Laplace approximation to the negative log-marginal likelihood $-\log\left(p(y|F,\theta,\xi)\right)$ is given by

$$\begin{aligned}L^{LA}(y,F,\theta,\xi) = &-\log p(y|\tilde{\mu},\xi) + \frac{1}{2}\tilde{b}^T\Sigma^{-1}\tilde{b}\\ &+\frac{1}{2}\log\det\big(\Sigma Z^T\tilde{W}Z+I_m\big).\end{aligned} \tag{6}$$

Since

$$p(y|F,\theta,\xi) = \frac{p(y|F,b,\xi)p(b|\theta)}{p(b|y,\theta,\xi)},$$

the Laplace approximation in (5) is equivalent to the following Gaussian approximation to the posterior $p(b|y,\theta,\xi)$:

$$p(b|y,\theta,\xi) \approx \mathcal{N}\Big(\tilde{b},\Big(Z^T\tilde{W}Z+\Sigma^{-1}\Big)^{-1}\Big). \tag{7}$$

2.2.1 Gradients

For the boosting algorithms introduced in the following, we need to calculate $\frac{\partial L^{LA}(y,F,\theta,\xi)}{\partial F}$ and also $\frac{\partial L^{LA}(y,F,\theta,\xi)}{\partial \theta}$ and $\frac{\partial L^{LA}(y,F,\theta,\xi)}{\partial \xi}$ if, e.g., a first-order optimization method is used for minimizing with respect to θ and ξ. These are obtained as follows.

Proposition 2.1. *The gradients with respect to F, θ, and ξ of the approximate negative logarithmic marginal likelihood of the Laplace approximation $L^{LA}(y,F,\theta,\xi)$ in (6) are given by*

$$\begin{aligned}\frac{\partial L^{LA}(y,F,\theta,\xi)}{\partial F_i} = &-\frac{\partial \log p(y_i|\tilde{\mu}_i,\xi)}{\partial \tilde{\mu}_i}\\ &+\frac{1}{2}\operatorname{tr}\bigg(\Big(Z^T\tilde{W}Z+\Sigma^{-1}\Big)^{-1}Z^T\frac{\partial \tilde{W}}{\partial F_i}Z\bigg)\\ &+\bigg(\frac{\partial L^{LA}(y,F,\theta,\xi)}{\partial \tilde{b}}\bigg)^T\frac{\partial \tilde{b}}{\partial F_i},\end{aligned} \tag{8}$$

$$\begin{aligned}\frac{\partial L^{LA}(y,F,\theta,\xi)}{\partial \theta_k} = &-\frac{1}{2}\tilde{b}^T\Sigma^{-1}\frac{\partial \Sigma}{\partial \theta_k}\Sigma^{-1}\tilde{b}\\ &+\frac{1}{2}\operatorname{tr}\bigg(\Big(\Sigma+(Z^T\tilde{W}Z)^{-1}\Big)^{-1}\frac{\partial \Sigma}{\partial \theta_k}\bigg)\\ &+\bigg(\frac{\partial L^{LA}(y,F,\theta,\xi)}{\partial \tilde{b}}\bigg)^T\frac{\partial \tilde{b}}{\partial \theta_k},\end{aligned} \tag{9}$$

$$\begin{aligned}\frac{\partial L^{LA}(y,F,\theta,\xi)}{\partial \xi_l} = &-\frac{\partial \log p(y|\tilde{\mu},\xi)}{\partial \xi_l}\\ &+\frac{1}{2}\operatorname{tr}\bigg(\Big(Z^T\tilde{W}Z+\Sigma^{-1}\Big)^{-1}Z^T\frac{\partial \tilde{W}}{\partial \xi_l}Z\bigg)\\ &+\bigg(\frac{\partial L^{LA}(y,F,\theta,\xi)}{\partial \tilde{b}}\bigg)^T\frac{\partial \tilde{b}}{\partial \xi_l},\end{aligned} \tag{10}$$

for $i=1,\dots,n$, $k=1,\dots,q$, $l=1,\dots,r$, where

$$\frac{\partial L^{LA}(y,F,\theta,\xi)}{\partial \tilde{b}_j} = \frac{1}{2}\operatorname{tr}\bigg(\Big(Z^T\tilde{W}Z+\Sigma^{-1}\Big)^{-1}Z^T\frac{\partial \tilde{W}}{\partial \tilde{b}_j}Z\bigg), \tag{11}$$

$$\frac{\partial \tilde{b}}{\partial F_i} = -\Big(Z^T\tilde{W}Z+\Sigma^{-1}\Big)^{-1}Z^T\tilde{W}_{\cdot i} \tag{12}$$

$$\frac{\partial \tilde{b}}{\partial \theta_k} = \Big(Z^T\tilde{W}Z+\Sigma^{-1}\Big)^{-1}\Sigma^{-1}\frac{\partial \Sigma}{\partial \theta_k}Z^T\frac{\partial \log p(y|\tilde{\mu},\xi)}{\partial \tilde{\mu}}, \tag{13}$$

$$\frac{\partial \tilde{b}}{\partial \xi_l} = \Big(Z^T\tilde{W}Z+\Sigma^{-1}\Big)^{-1}Z^T\frac{\partial^2 \log p(y|\tilde{\mu},\xi)}{\partial \xi_l \partial \tilde{\mu}}, \tag{14}$$

$\tilde{W}_{\cdot i}$ denotes column i of $\tilde{W}$, $\frac{\partial \tilde{W}}{\partial F_i} = \operatorname{diag}(-\frac{\partial^3 \log p(y_i|\tilde{\mu}_i,\xi)}{\partial \tilde{\mu}_i^3})$, $\frac{\partial \tilde{W}}{\partial \xi_l} = \operatorname{diag}(-\frac{\partial^3 \log p(y_i|\tilde{\mu}_i,\xi)}{\partial \tilde{\mu}_i^2 \partial \xi_l})$, $\frac{\partial \tilde{W}}{\partial \tilde{b}_j} = \operatorname{diag}(-\frac{\partial^3 \log p(y_i|\tilde{\mu}_i,\xi)}{\partial \tilde{\mu}_i^3}Z_{ij})$.

Proof of Proposition 2.1 The derivation is similar as in Williams and Rasmussen [6, Chapter 5.5.1]. All three gradients are sums of the explicit derivatives of $L^{LA}(y,F,\theta,\xi)$ and implicit derivatives through the dependency of $\tilde{b}$ on F, θ, and ξ. The explicit derivatives with respect to F, θ, and ξ, ignoring any dependency through $\tilde{b}$, are given in the first two summands in (8), (9), and (10). For the implicit derivatives, we first note that

$$\frac{\partial L^{LA}(y,F,\theta,\xi)}{\partial \tilde{b}_j} = \frac{1}{2}\operatorname{tr}\bigg(\Big(Z^T\tilde{W}Z+\Sigma^{-1}\Big)^{-1}Z^T\frac{\partial \tilde{W}}{\partial \tilde{b}_j}Z\bigg)$$

where we use the fact that the derivative of the first two terms in (6) with respect to $\tilde{b}$ vanishes, and

$$\frac{\partial \tilde{W}}{\partial \tilde{b}_j} = -\text{diag}\left(\frac{\partial^3 \log p(y_i|\tilde{\mu}_i,\xi)}{\partial \tilde{\mu}_i^3} Z_{ij}\right)$$

since

$$\begin{aligned}\frac{\partial \tilde{W}_{ii}}{\partial \tilde{b}_j} &= \left(\frac{\partial \tilde{W}_{ii}}{\partial \tilde{\mu}}\right)^T \frac{\partial \tilde{\mu}}{\partial \tilde{b}_j} \\ &= \frac{\partial \tilde{W}_{ii}}{\partial \tilde{\mu}_i} Z_{ij} \\ &= -\frac{\partial^3 \log p(y_i|\tilde{\mu}_i,\xi)}{\partial \mu_i^3} Z_{ij}.\end{aligned}$$

To find $\frac{\partial \tilde{b}}{\partial F_i}$, we differentiate

$$\begin{aligned}0 &= \frac{\partial}{\partial \tilde{b}}\left(\log p(y|\tilde{\mu},\xi) - \frac{1}{2}\tilde{b}^T\Sigma^{-1}\tilde{b}\right) \\ &= Z^T \frac{\partial \log p(y|\tilde{\mu},\xi)}{\partial \tilde{\mu}} - \Sigma^{-1}\tilde{b} \qquad (15)\end{aligned}$$

with respect to F_i and obtain

$$\begin{aligned}0 &= Z^T \frac{\partial^2 \log p(y|\tilde{\mu},\xi)}{\partial \tilde{\mu}_i \partial \tilde{\mu}} \\ &\quad + \frac{\partial}{\partial \tilde{b}}\left(Z^T \frac{\partial \log p(y|\tilde{\mu},\xi)}{\partial \tilde{\mu}} - \Sigma^{-1}\tilde{b}\right)\frac{\partial \tilde{b}}{\partial F_i} \\ &= -Z^T \tilde{W}_{\cdot i} + \left(-Z^T\tilde{W}Z - \Sigma^{-1}\right)\frac{\partial \tilde{b}}{\partial F_i},\end{aligned}$$

where $\tilde{W}_{\cdot i} = \frac{\partial^2 \log p(y|\tilde{\mu},\xi)}{\partial \tilde{\mu}_i \partial \tilde{\mu}}$ is column i of $\tilde{W}$, i.e., a vector of 0's except for the ith entry which is given by $\frac{\partial^2 \log p(y_i|\tilde{\mu}_i,\xi)}{\partial \tilde{\mu}_i^2}$. The statement in Equation (12) thus follows. Similarly, multiplying Equation (15) with Σ and differentiating it with respect to θ_k gives

$$0 = \frac{\partial \Sigma}{\partial \theta_k} Z^T \frac{\partial \log p(y|\tilde{\mu},\xi)}{\partial \tilde{\mu}} + \left(-\Sigma Z^T \tilde{W} Z - I_m\right)\frac{\partial \tilde{b}}{\partial \theta_k},$$

from which we obtain Equation (13) by multiplying with Σ^{-1}. Equation (14) follows analogously. □

We note that in our software implementation, we use different equivalent versions of the above result depending on the specific latent Gaussian model for computational efficiency and stability. If Zb consists of only grouped random effects, we use the version presented in Proposition 2.1 except that in (9), we replace $(\Sigma + (Z^T\tilde{W}Z)^{-1})^{-1}\frac{\partial \Sigma}{\partial \theta_k}$ with the equivalent expression $(Z^T\tilde{W}Z + \Sigma^{-1})^{-1}\Sigma^{-1}\frac{\partial \Sigma}{\partial \theta_k}Z^T\tilde{W}Z$. In this case, Z and Σ^{-1} are sparse, and the random effects dimension m is smaller than the number of samples n. It follows that a Cholesky factor for $Z^T\tilde{W}Z + \Sigma^{-1}$ can be computed efficiently using sparse matrix algebra, and also the remaining calculations for obtaining the gradients in Proposition 2.1 can be done efficiently. If Zb contains a finite dimensional versions of a Gaussian process, we use the Sherman-Morrison-Woodbury formula $(Z^T\tilde{W}Z + \Sigma^{-1})^{-1} = \Sigma - \Sigma Z^T\tilde{W}^{1/2}(I_m + \tilde{W}^{1/2}Z\Sigma Z^T\tilde{W}^{1/2})^{-1}\tilde{W}^{1/2}Z\Sigma$, factorize the matrix $I_m + \tilde{W}^{1/2}Z\Sigma Z^T\tilde{W}^{1/2}$, and, similarly as in Williams and Rasmussen [6, Chapter 5.5.1], adapt all calculations in Proposition 2.1 accordingly.

3 Latent Gaussian Model Boosting

We propose to do the minimization of the risk functional in (3) using a novel boosting algorithm presented in the following. For known and fixed θ and ξ, boosting finds a minimizer of the approximate empirical risk functional $R^A(F(\cdot),\theta,\xi)$ in a greedy way by sequentially adding an update $f_m(\cdot)$ to the current estimate $F_{m-1}(\cdot)$:

$$F_m(\cdot) = F_{m-1}(\cdot) + f_m(\cdot), \quad f_m \in \mathcal{S}, \qquad (16)$$

where $f_m(\cdot)$, $m = 1,\ldots,M$, is chosen such that its addition results in the minimization of the risk. This minimization cannot be done analytically and an approximation is thus used. In general, such an approximation consists of either a penalized functional first-order or a functional second-order Taylor expansion of the risk around the current estimate $F_{m-1}(\cdot)$. This corresponds to functional gradient descent or functional Newton steps. See Sigrist [37] for more information on the distinction between gradient and Newton boosting.

In our case, we use functional gradient descent. Specifically, $f_m(\cdot)$ is given by the least squares approximation to the vector obtained when evaluating the negative functional gradient of $R^A(F(\cdot),\theta,\xi)$ at $(F_{m-1}(\cdot), I_{X_i}(\cdot))$, $i = 1,\ldots,n$, where $I_{X_i}(\cdot)$ are indicator functions which equal 1 at X_i and 0 otherwise. Equivalently, $f_m(\cdot)$ is the minimizer of a first-order functional Taylor approximation of $R^A(F(\cdot),\theta,\xi)$ around $F_{m-1}(\cdot)$ with an L^2 penalty on $f(\cdot)$ evaluated at (X_i); see, e.g., Sigrist [37] for more information. It is easily seen that the negative Gâteaux derivative of $R^A(F(\cdot),\theta,\xi)$ evaluated at $(F_{m-1}(\cdot), I_{X_i}(\cdot))$ is given by the vector $-\frac{\partial L^A(y,F,\theta,\xi)}{\partial F}\Big|_{F=F_{m-1}(X)}$ which we denote shortly as $-\frac{\partial L^A(y,F_{m-1},\theta,\xi)}{\partial F}$. This means that $f_m(\cdot)$ can be found as the following least squares approximation:

$$f_m(\cdot) = \underset{f(\cdot)\in\mathcal{S}}{\text{argmin}}\left\| -\frac{\partial L^A(y,F_{m-1},\theta,\xi)}{\partial F} - f\right\|^2, \qquad (17)$$

where $f = (f(X_1),\ldots,f(X_n))^T$. Note that $\frac{\partial L^A(y,F_{m-1},\theta,\xi)}{\partial F}$ depends on the approximation used for the marginal log-likelihood. For the Laplace approximation, this is given in Proposition 2.1.

It has been empirically observed that damping the update in (16),

$$F_m(\cdot) = F_{m-1}(\cdot) + \nu f_m(\cdot), \quad \nu > 0,$$

results in higher prediction accuracy [2]. Further, functional gradient descent can also be accelerated using momentum. For instance, Biau *et al.* [38] and Lu *et al.* [39] propose to use Nesterov acceleration [40] for gradient boosting.

To jointly learn $F(\cdot)$ and (θ,ξ), we propose to combine functional boosting updates in the direction of $F(\cdot)$ with coordinate descent steps in θ and ξ. The reasons for this choice are outlined in Sigrist [14]. The LaGaBoost Algorithm 1 summarizes our approach. Note that, despite not being explicitly stated in Algorithm 1, the approximation for the negative logarithmic marginal likelihood needs to be calculated repeatedly in the algorithm whenever $L^A(y,F,\theta,\xi)$ is evaluated or a gradient of it is calculated.

Algorithm 1. LaGaBoost: Latent Gaussian model Boosting

Input : Initial values $\theta_0 \in \Theta$ and, if applicable, $\xi_0 \in \Xi$, learning rate $\nu > 0$, number of boosting iterations $M \in \mathbb{N}$, approximation $L^A(y, F, \theta, \xi)$

Output: Function $\hat{F}(\cdot) = F_M(\cdot)$, hyperparameters $\hat{\theta} = \theta_M$, and auxiliary parameters $\hat{\xi} = \xi_M$

1: Initialize $F_0(\cdot) = \text{argmin}_{c \in \mathbb{R}} L^A(y, c \cdot 1, \theta_0, \xi_0)$
2: **for** $m = 1$ **to** M **do**
3: Find $(\theta_m, \xi_m) = \underset{(\theta,\xi)\in(\Theta,\Xi)}{\text{argmin}}\ L^A(y, F_{m-1}, \theta, \xi)$ using a method for convex optimization initialized with $(\theta_{m-1}, \xi_{m-1})$
4: Find $f_m(\cdot) = \underset{f(\cdot)\in\mathcal{S}}{\text{argmin}} \| - \frac{\partial L^A(y, F_{m-1}, \theta_m, \xi_m)}{\partial F} - f\|^2$
5: Update $F_m(\cdot) = F_{m-1}(\cdot) + \nu f_m(\cdot)$
6: **end for**

If the risk functional $R^A(F(\cdot), \theta, \xi)$ is convex in its arguments and Θ and Ξ are convex sets, then (3) is a convex optimization problem since $\mathcal{H} = span(\mathcal{S})$ is also convex. This means that there exists a unique minimizer and the LaGaBoost algorithm converges to the minimum, as long as the learning rate ν is not too large to avoid overshooting, i.e., that the risk increases when doing too large steps. Further, the computational complexity of the algorithm depends on the specific latent Gaussian variable model and the marginal likelihood approximation used. For instance, for the Laplace approximation, the calculation of Cholesky factors is usually the bottleneck.

3.1 Out-of-sample Learning for Hyperparameters

It has recently been observed that state-of-the-art machine learning techniques such as neural networks, kernel machines, or boosting can achieve a zero training loss and interpolate the training data while at the same time having excellent generalization properties [41], [42], [43], [44], [45]. Such an interpolation of the training data could be problematic for the hyperparameter estimation in the LaGaBoost algorithm. We propose to circumvent this potential problem by estimating the hyperparameters θ and the auxiliary parameters ξ using out-of-sample validation data obtained by applying cross-validation or by partitioning the data into two disjoint training and validation sets. To avoid that the function $F(\cdot)$ and/or the parameters θ and ξ are only learned on a fraction of the full data, we propose a two-step approach presented in the LaGaBoostOOS Algorithm 2. In brief, the LaGaBoostOOS algorithm first runs the LaGaBoost algorithm on the training data and obtains predictions $\hat{F}_{val}$ for the function $F(\cdot)$ on the left out validation data. The parameters θ and ξ are then estimated on the validation data using the predicted values $\hat{F}_{val}$. Finally, the LaGaBoost algorithm is run a second time on the full data while holding θ and ξ fixed. When k-fold cross-validation is used, both the function $F(\cdot)$ and the parameters θ and ξ are thus learned using the full data.

3.2 Prediction

In the following, we show how predictions can be made. We distinguish between predicting observables variables y_p and latent variables μ_p. Let $y_p \in \mathbb{R}^{n_p}$ and $\mu_p \in \mathbb{R}^{n_p}$ denote the observable and latent random variables for which predictions should be made. The following holds true:

$$\begin{pmatrix} b \\ \mu_p \end{pmatrix} = \begin{pmatrix} 0 \\ F(X_p) \end{pmatrix} + \begin{pmatrix} (I_m, 0_{m\times m_p}) \\ Z_p \end{pmatrix} \begin{pmatrix} b \\ b_p \end{pmatrix}, \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ F(X_p) \end{pmatrix}, \begin{pmatrix} \Sigma & (\Sigma, \Sigma_{op}) Z_p^T \\ Z_p(\Sigma, \Sigma_{op})^T & Z_p \begin{pmatrix} \Sigma & \Sigma_{op} \\ \Sigma_{op}^T & \Sigma_p \end{pmatrix} Z_p^T \end{pmatrix} \right) \tag{18}$$

where $b_p \in \mathbb{R}^{m_p}$ is a latent random variable for which no corresponding data has been observed in y, $(I_m, 0_{m\times m_p}) \in \mathbb{R}^{m\times(m+m_p)}$, $I_m \in \mathbb{R}^{m\times m}$ is an identity matrix, $0_{m\times m_p} \in \mathbb{R}^{n\times m_p}$ is a matrix of zeros, the matrix $Z_p \in \mathbb{R}^{n_p\times(m+m_p)}$ relates the vector of observed and new latent variables $(b^T, b_p^T)^T \in \mathbb{R}^{m+m_p}$ to μ_p, $(\Sigma, \Sigma_{op}) \in \mathbb{R}^{m\times(m+m_p)}$, $\Sigma_{op} = \text{Cov}(b, b_p)$, $\Sigma_p = \text{Cov}(b_p)$, and $X_p \in \mathbb{R}^{n_p\times p}$ is the predictor variable matrix of the predictions.

Algorithm 2. LaGaBoostOOS: Latent Gaussian model Boosting with Out-Of-Sample hyperparameter estimation

Input : Initial values $\theta_0 \in \Theta$ and, if applicable, $\xi_0 \in \Xi$, learning rate $\nu > 0$, number of boosting iterations $M \in \mathbb{N}$, approximation $L^A(y, F, \theta, \xi)$

Output: Function $\hat{F}(\cdot) = F_M(\cdot)$, hyperparameters $\hat{\theta} = \theta_M$, and auxiliary parameters $\hat{\xi} = \xi_M$

1: Partition the data into training and validation sets, e.g., using k-fold cross-validation or by partitioning the data into two disjoint sets
2: Run the LaGaBoost algorithm on the training data and generate predictions $\hat{F}_{val}$ for the function $F(\cdot)$ on the validation data
3: Find $(\hat{\theta}, \hat{\xi}) = \text{argmin}_{(\theta,\xi)\in(\Theta,\Xi)} L^A(y_{val}, \hat{F}_{val}, \theta, \xi)$ using the validation data with response variable y_{val}
4: Run the LaGaBoost algorithm on the full data while holding the hyperparameters θ and auxiliary parameters ξ fixed at $\hat{\theta}$ and $\hat{\xi}$, i.e., by skipping line 3 in Algorithm 1, to obtain $\hat{F}(\cdot)$

By the law of total probability, we have

$$p(\mu_p|y, \theta, \xi) = \int p(\mu_p|b, \theta) p(b|y, \theta, \xi) db$$

and

$$p(y_p|y, \theta, \xi) = \int p(y_p|\mu_p, \xi) p(\mu_p|y, \theta, \xi) d\mu_p. \tag{19}$$

If we apply the Laplace approximation, then by (7), (18), standard results for conditional distributions of multivariate Gaussian distributions, and the law of total variance, we have

$$p(\mu_p|y, \theta, \xi) \approx \mathcal{N}(\omega_p, \Omega_p),$$

where

$$\begin{aligned} \omega_p &= F(X_p) + Z_p(\Sigma, \Sigma_{op})^T \Sigma^{-1} \tilde{b}, \\ &= F(X_p) + Z_p(\Sigma, \Sigma_{op})^T Z^T \frac{\partial \log p(y|\tilde{\mu}, \xi)}{\partial \tilde{\mu}}, \end{aligned}$$

$$\begin{aligned} \Omega_p = Z_p &\begin{pmatrix} \Sigma & \Sigma_{op} \\ \Sigma_{op}^T & \Sigma_p \end{pmatrix} Z_p^T \\ &- Z_p(\Sigma, \Sigma_{op})^T \left(\Sigma + (Z^T \tilde{W} Z)^{-1} \right)^{-1} (\Sigma, \Sigma_{op}) Z_p^T, \end{aligned}$$

where in the last line, we have used the Sherman-Morrison-Woodbury formula.

Further, the integral in (19) is analytically tractable for a Bernoulli likelihood with a probit link [see, e.g., 6, Chapter 3.9], but for other likelihoods, it needs to be numerically approximated. In our software implementation and the experiments below, we use adaptive Gauss-Hermite quadrature [46] as numeric integration technique.

3.3 Software Implementation

The LaGaBoost and LaGaBoostOOS algorithms based on the Laplace approximation are implemented in the `GPBoost` library written in C++ with corresponding Python and R packages; see https://github.com/fabsig/GPBoost for more information. For linear algebra calculations, we rely on the `Eigen` library [47]. Sparse matrix algebra is used, in particular for calculating Cholesky decompositions, whenever covariance matrices are sparse, e.g., in the case of grouped random effects. Further, multi-processor parallelization is done using `OpenMP`. For the tree-boosting part, in particular the tree growing algorithm, we use the `LightGBM` library [48]. The `GPBoost` library allows for modeling Gaussian processes, grouped random effects including nested and crossed ones, random coefficients, and combinations of the former. Further, the `GPBoost` library currently implements gradient descent with optional Nesterov acceleration and the Nelder-Mead method for minimizing with respect to the parameters θ and ξ in line 3 of the LaGaBoost Algorithm 1.

4 Simulated Experiments

In the following, we perform simulated experiments to compare the novel LaGaBoost algorithm to alternative approaches. We simulate binary classification data from a latent Gaussian model as in (1) assuming a Bernoulli likelihood with a probit link function: $y_i \in \{0,1\}$, $P(y_i = 1) = \Phi(\mu_i)$, $i = 1, \ldots, n$, where $\Phi(\cdot)$ denotes the standard normal cumulative distribution function. For the latent Gaussian variable Zb, we consider both grouped random effects with a single grouping level and a spatial Gaussian process model with an exponential covariance function $c(s, s') = \sigma_1^2 \exp(-\|s - s'\|/\rho)$ where the locations s are in $[0,1]^2$ and $\rho = 0.1$. The marginal variance in both models is set to $\sigma^2 = 1$. Concerning the function $F(\cdot)$ and the predictor variables X, we sample independently from

$$F(x) = C_1 + C_2 \cdot (2x_1 + x_2^2 + 4 \cdot 1_{\{x_3 > 0\}} + 2\log(|x_1|)x_3),$$
$$x = (x_1, \ldots, x_9)^T, \quad x \sim \mathcal{N}(0, I_9). \tag{20}$$

This function has been used previously in Hajjem *et al.* [13] and Sigrist [14] to compare non-parametric mixed effects models for Gaussian data. The constant C_1 is chosen such that the mean of $F(x)$ is approximately 0, and C_2 is chosen such that the variance of $F(x)$ equals approximately 1, i.e., that $F(x)$ has the same signal strength as the latent Gaussian variable.

We simulate 100 times training data sets of size n and two test data sets each also of size n. All models are trained on the training data and evaluated on the test data. We use a sample size of $n = 5000$ for the grouped random effects

Fig. 1. Example of locations for training and test data for the spatial data. 'Test' and 'Test_ext' refers to locations of the interpolation and extrapolation test data sets, respectively.

with $m = 500$ different groups. This corresponds to a categorical variable with 500 different categories and 10 samples per category. For the Gaussian process model, we use a sample size of $n = 500$. The reason for using a smaller sample size is that this allows us to avoid any additional approximation error due to a large data approximation. In every simulation run, two test data sets, denoted as "interpolation" and "extrapolation" test sets, are generated as follows. For the grouped random effects model, the interpolation test data set consists of n samples from the same m groups as in the training data, and the extrapolation test data consists of n samples for m new groups that have not been observed in the training data. For the Gaussian process model, training data locations are samples uniformly from $[0,1]^2$ excluding $[0.5,1]^2$, the interpolation test data sets are obtained by also simulating locations uniformly in the same area, and the extrapolation test data contains locations sampled uniformly from the excluded square $[0.5,1]^2$. Fig. 1 illustrates this.

We compare the LaGaBoost algorithm based on the Laplace approximation to the following alternative approaches: linear Gaussian process and grouped random effects models for binary data with a probit link function and $F(x) = x^T\beta, \beta \in \mathbb{R}^p$, independent Newton boosting for binary data with the log loss ('LogitBoost') [49], and model-based gradient boosting ('mboost') [23] with the log loss, i.e., a negative Bernoulli log-likelihood, and a probit link function. For the LogitBoost algorithm, we include the locations for the spatial data and the categorical grouping variable as additional predictor variables in the function $F(\cdot)$. For all boosting algorithms, we use trees as base learners, except for the grouped and spatial random effects in mboost. Learning and prediction with the LaGaBoost and LaGaBoostOOS algorithms, the linear latent Gaussian models, and LogitBoost is done using the `GPBoost` library version 0.7.0 compiled with the MSVC compiler version 19.24.28315.0 and OpenMP version 2.0. For the linear latent Gaussian models, the LaGaBoost algorithm, and the LaGaBoostOOS algorithm, optima for hyperparameters θ are found using Nesterov accelerated gradient

TABLE 1
Results for the Grouped/High-Cardinality Categorical Variable Data and a Binary Bernoulli Likelihood

	LaGaBoost	LinearME	LogitBoost	mboost	LaGaBoostOOS
Error	**0.2373**	0.287	0.3393	0.2825	0.2373
(sd)	(0.00674)	(0.00885)	(0.00871)	(0.00958)	(0.00677)
[p-val]		[1e-86]	[2.8e-106]	[6.4e-73]	[0.66]
NegLL	**2421**	2785	3030	2917	2421
(sd)	(45.5)	(46.1)	(28.8)	(19.7)	(45.6)
[p-val]		[3.1e-105]	[8.2e-114]	[6.5e-111]	[0.67]
Error_ext	**0.3432**	0.4206	0.3533		0.3434
(sd)	(0.00793)	(0.00898)	(0.015)		(0.00783)
[p-val]		[1.9e-95]	[9.2e-12]		[0.31]
NegLL_ext	**3028**	3295	3078		3029
(sd)	(31.4)	(17.9)	(55)		(31.7)
[p-val]		[1.2e-105]	[7.5e-18]		[0.0036]
RMSE σ^2	0.2099	0.3589			0.2141
Bias σ^2	-0.1953	-0.3536			-0.1941
Time (s)	0.6646	0.03906	0.07785	16.96	3.276

The results for the "extrapolation" test data sets are denoted by '_ext'. 'NegLL' denotes the negative log-likelihood (=log loss for binary data).

descent. Further, for the linear models, the coefficients β are also learned using Nesterov accelerated gradient descent. Note that the gradient of $L^A(y, F, \theta, \xi)$ with respect to β is given by

$$\frac{\partial L^A(y, F, \theta, \xi)}{\partial \beta} = X^T \frac{\partial L^A(y, F, \theta, \xi)}{\partial F}.$$

For LogitBoost applied to the grouped random effects data, we consider the grouping variable as a numeric variable and not as a categorical variable as suggested by the authors of `LightGBM`[1] since the number of categories is large. Concerning the mboost algorithm, we use the `mboost` R package [50] version 2.9-2, where spatial effects are modeled using bivariate P-spline base learner (`bspatial` with `df=6`), grouped random effects are modeled using random effects base learners (`brandom` with `df=4`), and all other predictor variables are modeled using trees as base learners. All calculations are done on a laptop with a 2.9 GHz quad-core processor and 16 GB of random-access memory (RAM).

Tuning parameters are chosen by simulating 10 additional training and test sets and choosing the parameter combinations that minimize the average log loss on the test sets. In doing so, we use the union of both the interpolation and extrapolation test data sets to calculate test losses. For all boosting algorithms, we consider the following grid of tuning parameters: the number of boosting iterations $M \in \{1, \ldots, 1000\}$, the learning rate $\nu \in \{0.1, 0.05, 0.01\}$, the maximal depth of the trees $\in \{1, 2, 5, 10\}$, and the minimal number of samples per leaf $\in \{1, 10, 100\}$.

The results for the grouped and spatial data are reported in Tables 1 and 2. We report average test error rates ('Error') and test log losses ('NegLL') for both the interpolation and extrapolation ('_ext') test sets. Further, we calculate p-values of paired t-tests comparing the LaGaBoost algorithm to the other approaches. We find that the LaGaBoost algorithm significantly outperforms all alternative approaches in all prediction accuracy measures for both the grouped and spatial data. In Tables 1 and 2, we additionally report the results for the LaGaBoostOOS algorithm, root mean square errors (RMSEs) and biases for the hyperparameters, and wall-clock time. Overall, we observe no large differences between the LaGaBoost and the LaGaBoostOOS algorithms. However, for the spatial data, the hyperparameter estimates of the LaGaBoostOOS algorithm have smaller RMSEs and biases compared to the LaGaBoost algorithm in line with our arguments laid out in Section 3.1. As expected, the LaGaBoostOOS algorithm has a higher computational time.

An alternative option to simulating additional training and test sets for choosing tuning parameters is to use cross-validation on the training data sets in every of the 100 simulation runs. However, this is computationally more expensive as the number of simulation runs is relatively large. To investigate the differences between these two options for choosing tuning parameters, we redo the simulated experiments with 10 simulation runs and choose tuning parameters using 4-fold cross-validation on the training data in every simulation run. The results of this are reported in Tables A.1 and A.2 in the appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2022.3168152. Overall, we observe only minor differences.

Next, we also perform the same simulated experiments using a Poisson likelihood with a logarithmic link function instead of a binary Bernoulli likelihood. Specifically, we simulate grouped and spatial random effects as described above with $\sigma^2 = 0.2$, and we simulate $F(X)$ according to (20) with C_1 chosen as described above and C_2 chosen such that the variance of $F(X)$ is approximately 0.2. Response variable data is then simulated from a Poisson distribution with mean equaling $\exp(F(X) + Zb)$. Tuning parameters are chosen similarly as for the binary data by minimizing the test negative Poisson likelihood. Further, we use the RMSE and the negative Poisson likelihood for evaluating prediction accuracy. The results of this are reported in Tables 3 and 4. We find qualitatively very similar results as for the binary data. In particular, the LaGaBoost algorithm

1. https://lightgbm.readthedocs.io/en/latest/Advanced-Topics.html#categorical-feature-support (retrieved on May 11, 2021)

TABLE 2
Results for the Spatial Data and a Binary Bernoulli Likelihood

	LaGaBoost	LinearGP	LogitBoost	mboost	LaGaBoostOOS
Error	**0.3085**	0.3309	0.3501	0.3808	0.3068
(sd)	(0.0286)	(0.0278)	(0.0293)	(0.0336)	(0.027)
[p-val]		[1.4e-22]	[6e-30]	[8.6e-41]	[0.057]
NegLL	**290.5**	302.4	312.1	330.3	288.6
(sd)	(13.3)	(13.5)	(9.72)	(6.32)	(13.5)
[p-val]		[8.8e-31]	[8.8e-40]	[6.2e-55]	[7.3e-16]
Error_ext	**0.3755**	0.3953	0.3986	0.4283	0.3732
(sd)	(0.0419)	(0.0306)	(0.055)	(0.0662)	(0.0396)
[p-val]		[6.8e-07]	[3.3e-07]	[2.4e-14]	[0.044]
NegLL_ext	**320**	328.3	331.4	339.7	319.1
(sd)	(15.8)	(11.3)	(21)	(10.9)	(15.8)
[p-val]		[8.9e-10]	[7.9e-11]	[7e-26]	[0.00027]
RMSE σ^2	0.6237	0.4314			0.4944
RMSE ρ	0.1001	0.06225			0.05211
Bias σ^2	-0.5945	-0.3353			-0.4398
Bias ρ	0.06441	0.02141			0.01426
Time (s)	14.32	1.908	0.03377	0.7835	40.83

See the caption of Table 1 for information on the abbreviations used in this table.

significantly outperforms all alternative approaches in all prediction accuracy measures for both the grouped and the spatial data.

4.1 When Does the LaGaBoost Algorithm Outperform Independent Boosting?

It is relatively obvious that the LaGaBoost algorithm tends to outperform linear latent Gaussian models when there are non-linearities and interactions. It is less clear in which situations the LaGaBoost algorithm outperforms classical boosting algorithms which include categorical variables and/or spatial locations in the predictor variables X for $F(\cdot)$ and, conditionally on this, assume independence among samples. As mentioned in Section 1.1, intuitively, we expect that the improvement in prediction accuracy of our novel approach over independent tree-boosting is the larger, the smaller the number of observations per category of a categorical variable is and the faster the covariance decays over space and/or time. To analyze this, we repeat the above simulated experiments for the binary data with varying numbers of samples per group and varying range parameters ρ. Specifically, for the grouped random effects with $n = 5000$ samples, we consider the following number of samples per group: 10, 20, 50, 100, and 200. For the spatial data, we consider the following range parameters ρ: 0.1, 0.2, 0.5, and 1. Apart from this, we use the same experimental setup as above for the Bernoulli likelihood.

Fig. 2 reports the relative decrease in the test error of the LaGaBoost algorithm compared to the LogitBoost algorithm as well as the average test error of the two algorithms for the interpolation test data sets. These results confirm our hypothesis that the improvement in prediction accuracy is the larger, the smaller the number of observations per group is and the faster the covariance decays over space. In other words, the higher the complexity of the underlying true

TABLE 3
Results for the Grouped/High-Cardinality Categorical Variable Data and a Poisson Likelihood

	LaGaBoost	LinearME	PoissonBoost	mboost	LaGaBoostOOS
RMSE	**1.465**	1.614	1.575	1.528	1.445
(sd)	(0.514)	(0.531)	(0.503)	(0.51)	(0.5)
[p-val]		[2.5e-12]	[4e-55]	[9.6e-09]	[1.3e-05]
NegLL	**7062**	7504	7515	7293	7033
(sd)	(215)	(285)	(235)	(239)	(179)
[p-val]		[3.1e-62]	[6.8e-77]	[2.9e-39]	[8.8e-07]
RMSE_ext	**1.516**	1.599	1.536		1.504
(sd)	(0.482)	(0.481)	(0.481)		(0.468)
[p-val]		[3.9e-75]	[1.1e-07]		[0.0066]
NegLL_ext	**7488**	7884	7546		7466
(sd)	(212)	(286)	(218)		(181)
[p-val]		[1.2e-66]	[1.3e-14]		[7.9e-05]
RMSE σ^2	0.05751	0.02506			0.04392
Bias σ^2	-0.05382	-0.00339			-0.03661
Time (s)	0.3154	0.02697	0.0606	8.992	5.368

See the caption of Table 1 for information on the abbreviations used in this table.

TABLE 4
Results for the Spatial Data and a Poisson Likelihood

	LaGaBoost	LinearGP	PoissonBoost	mboost	LaGaBoostOOS
RMSE	**1.415**	1.465	1.452	1.494	1.421
(sd)	(0.324)	(0.324)	(0.322)	(0.33)	(0.323)
[p-val]		[4.7e-24]	[1.2e-17]	[4.6e-28]	[7.7e-05]
NegLL	**747.2**	767	764.1	787.5	749.6
(sd)	(53.5)	(56.5)	(55)	(62.2)	(53)
[p-val]		[2.4e-30]	[9.2e-22]	[1.9e-34]	[4.5e-05]
RMSE_ext	**1.505**	1.528	1.525	1.554	1.507
(sd)	(0.624)	(0.623)	(0.623)	(0.621)	(0.622)
[p-val]		[2.7e-09]	[1e-08]	[2.1e-21]	[0.17]
NegLL_ext	**773.4**	785.6	783.1	799.1	774.2
(sd)	(94)	(96.1)	(95.8)	(92.1)	(91.1)
[p-val]		[4.6e-09]	[2.4e-09]	[3.8e-24]	[0.36]
RMSE σ^2	0.09264	0.1217			0.1256
RMSE ρ	0.06708	0.06526			0.06721
Bias σ^2	-0.06304	0.08664			0.08688
Bias ρ	-0.003803	-0.05122			-0.05566
Time (s)	10.16	2.022	0.04036	0.689	26.77

See the caption of Table 1 for information on the abbreviations used in this table.

function relative to the sample size, the larger is the improvement obtained by the LaGaBoost algorithm. We conjecture that this is not just due to more accurate learning of the random effects themselves but also more efficient learning of the remaining part of the predictor function $F(\cdot)$. Fig. 2 also shows that, as expected, average test errors of both algorithms decrease when having fewer categories for the categorical grouping variables and when the correlation decays slower of space.

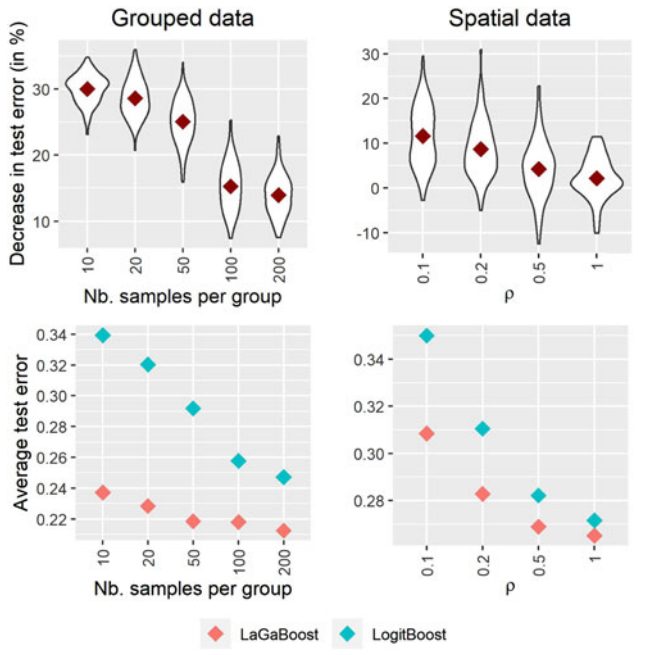

Fig. 2. Comparison of the LaGaBoost and LogitBoost algorithms for grouped data with varying number of samples per group and spatial data with varying range parameters ρ. The top row shows the relative decrease in test error of the LaGaBoost versus the LogitBoost algorithm visualized using violin plots. The red rhombi representing means over the simulation runs. The bottom row shows the average test error of the two algorithms.

5 Real-World Applications

In the following, we apply the LaGaBoost algorithm to two real-world binary classification data sets and compare its prediction accuracy to alternative approaches. We consider both grouped data with a high-cardinality categorical variable and a spatial data set. For the former, we consider data on poverty among young females in the US collected by the National Longitudinal Study of Youth (NLSY) and available from https://www3.nd.edu/~rwilliam/statafiles/teenpovxt.dta. Here, the person id number is the high-cardinality categorical variable that determines the grouped random effects, and the goal is to predict the binary poverty indicator. As a spatial data set, we consider species distribution data. Specifically, we use presence-absence data on rainforest understorey vascular plants in North-east New South Wales, Australia, species "nsw43" obtained from the `disdat` R package [51]. The goal is to predict the presence or absence of the species. Table 5 summarizes the data sets.

We compare the LaGaBoost algorithm to the same alternative approaches as in the simulated experiments in Section 4 using nested 4-fold cross-validation. For the grouped poverty data, we perform stratified cross-validation such that every fold contains approximately the same amount of data for every category of the grouping variable. A reason for doing this is that one of the alternative approaches, the `mboost` R package, does not allow for making predictions for unobserved groups. Tuning parameters are chosen by doing an additional inner 4-fold cross-validation on every

TABLE 5
Summary of Real-World Data Sets

Name	Data type	# data	Freq. of 1's	# features	# cat.
Poverty	Grouped	5755	36.79 %	7 + 1	1151
Species	Spatial	909	23.76 %	13 + 2	

'# data' denotes the sample size, '# features' the number of predictor variables, and '# cat.' the number of groups of the high-cardinality categorical variable.

TABLE 6
Results for the Real-World Data Sets

	LaGaBoost	Linear	LogitBoost	mboost
Poverty (grouped/high-cardinality categorical data)				
Error	**0.2792**	0.2848	0.3239	0.3197
AUC	**0.7318**	0.7313	0.7001	0.7028
Log_loss	**0.5789**	0.5878	0.6013	0.6081
Species (spatial data)				
Error	**0.2365**	0.3003	0.2717	0.2728
AUC	**0.7383**	0.7061	0.683	0.635
Log_loss	**0.4933**	0.5625	0.527	0.5662

'Linear' denotes the linear grouped mixed effects and linear Gaussian process models.

of the four training data sets.[2] We consider the same set of tuning parameters and selection criterion as in the simulated experiments

The results are reported in Table 6. In addition to the test error and the test log loss, we also report the test area under the ROC curve (AUC). We find that the LaGaBoost algorithm outperforms all alternative methods in all three prediction accuracy metrics for both the grouped data with a high-cardinality categorical variable and the spatial data.

6 Conclusion

We have introduced a novel way for combining latent Gaussian models, such as Gaussian processes and random effects models, with boosting. This is done by applying functional gradient descent to the negative logarithmic marginal likelihood of a generalized mixed effects model in a boosting framework while jointly learning hyperparameters. We have obtained increased prediction accuracy compared to existing approaches in both simulated and real-world data experiments. Future research can investigate how the approximation used for the marginal likelihood impacts properties such as prediction accuracy and computational time of the LaGaBoost algorithm.

Acknowledgments

We are thankful to the anonymous reviewers for their valuable comments which helped to improve the paper.

References

[1] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. 13th Int. Conf. Mach. Learn.*, 1996, pp. 148–156.
[2] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, pp. 1189–1232, 2001.
[3] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 785–794.
[4] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need," 2021, *arXiv:2106.03253*.
[5] R. Johnson and T. Zhang, "Learning nonlinear functions using regularized greedy forest," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 5, pp. 942–954, May 2014.
[6] C. K. Williams and C. E. Rasmussen, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.
[7] J. Pinheiro and D. Bates, *Mixed-Effects Models in S and S-PLUS*. Berlin, Germany: Springer Science & Business Media, 2006.
[8] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*. Boca Raton, FL, USA: CRC Press, 1984.
[9] W. R. Tobler, "A computer movie simulating urban growth in the Detroit region," *Econ. Geogr.*, vol. 46, no. sup1, pp. 234–240, 1970.
[10] A. Hajjem, F. Bellavance, and D. Larocque, "Mixed effects regression trees for clustered data," *Statist. Probability Lett.*, vol. 81, no. 4, pp. 451–459, 2011.
[11] R. J. Sela and J. S. Simonoff, "RE-EM trees: A data mining approach for longitudinal and clustered data," *Mach. Learn.*, vol. 86, no. 2, pp. 169–207, 2012.
[12] W. Fu and J. S. Simonoff, "Unbiased regression trees for longitudinal and clustered data," *Comput. Statist. Data Anal.*, vol. 88, pp. 53–74, 2015.
[13] A. Hajjem, F. Bellavance, and D. Larocque, "Mixed-effects random forest for clustered data," *J. Statist. Comput. Simul.*, vol. 84, no. 6, pp. 1313–1328, 2014.
[14] F. Sigrist, "Gaussian process boosting," 2020, *arXiv:2004.02653*.
[15] C. Griesbach, B. Säfken, and E. Waldmann, "Gradient boosting for linear mixed models," *Int. J. Biostatist.*, vol. 17, no. 2, pp. 317–329, 2021.
[16] A. Rabinowicz and S. Rosset, "Trees-based models for correlated data," 2021, *arXiv:2102.08114*.
[17] A. Saha, S. Basu, and A. Datta, "Random forests for spatially dependent data," *J. Amer. Statist. Assoc.*, pp. 1–19, 2021. [Online]. Available: https://doi.org/10.1080/01621459.2021.1950003
[18] A. Hajjem, D. Larocque, and F. Bellavance, "Generalized mixed effects regression trees," *Statist. Probability Lett.*, vol. 126, pp. 114–118, 2017.
[19] M. Fokkema, N. Smits, A. Zeileis, T. Hothorn, and H. Kelderman, "Detecting treatment-subgroup interactions in clustered data with generalized linear mixed-effects model trees," *Behav. Res. Methods*, vol. 50, no. 5, pp. 2016–2034, 2018.
[20] J. L. Speiser, B. J. Wolf, D. Chung, C. J. Karvellas, D. G. Koch, and V. L. Durkalski, "BiMM tree: A decision tree method for modeling clustered and longitudinal binary outcomes," *Commun. Statist.-Simul. Comput.*, vol. 49, no. 4, pp. 1004–1023, 2020.
[21] J. L. Speiser, B. J. Wolf, D. Chung, C. J. Karvellas, D. G. Koch, and V. L. Durkalski, "BiMM forest: A random forest method for modeling clustered and longitudinal binary outcomes," *Chemometrics Intell. Lab. Syst.*, vol. 185, pp. 122–134, 2019.
[22] M. Pellagatti, C. Masci, F. Ieva, and A. M. Paganoni, "Generalized mixed effects random forest: A flexible approach to predict university student dropout," *Stat. Anal. Data Mining: ASA Data Sci. J.*, vol. 14, no. 3, pp. 241–257, 2021.
[23] T. Hothorn, P. Bühlmann, T. Kneib, M. Schmid, and B. Hofner, "Model-based boosting 2.0," *J. Mach. Learn. Res.*, vol. 11, no. Aug, pp. 2109–2113, 2010.
[24] T. Hastie and R. Tibshirani, "Generalized additive models," *Statist. Sci.*, vol. 1, no. 3, pp. 297–310, 1986.
[25] S. N. Wood, *Generalized Additive Models: An Introduction With R*. London, U.K.: Chapman and Hall/CRC, 2017.
[26] G. Tutz and F. Reithinger, "A boosting approach to flexible semiparametric mixed models," *Statist. Med.*, vol. 26, no. 14, pp. 2872–2900, 2007.
[27] A. Groll and G. Tutz, "Regularization for generalized additive mixed models by likelihood-based boosting," *Methods Inf. Med.*, vol. 51, no. 02, pp. 168–177, 2012.
[28] P. Bühlmann *et al.*, "Boosting for high-dimensional linear models," *Ann. Statist.*, vol. 34, no. 2, pp. 559–583, 2006.
[29] P. Bühlmann and B. Yu, "Boosting with the L2 loss: Regression and classification," *J. Amer. Statist. Assoc.*, vol. 98, no. 462, pp. 324–339, 2003.
[30] M. J. Saberian, H. Masnadi-Shirazi, and N. Vasconcelos, "TaylorBoost: First and second-order boosting algorithms with explicit margin control," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 2929–2934.
[31] F. Sigrist, "KTBoost: Combined kernel and tree boosting," *Neural Process. Lett.*, vol. 53, no. 2, pp. 1147–1160, 2021.
[32] A. E. Gelfand, H.-J. Kim, C. Sirmans, and S. Banerjee, "Spatial modeling with spatially varying coefficient processes," *J. Amer. Statist. Assoc.*, vol. 98, no. 462, pp. 387–396, 2003.

2. Note that one has to be careful when doing cross-validation for dependent data to avoid biased estimates of the generalization error as pointed out by, e.g., Rabinowicz and Rosset [52]. However, apart from the fact that the validation and test data sets are of slightly different sizes, our cross-validation setting preserves the distributional relation between the inner fold training and validation data sets and the training and test data sets and, consequently, no bias is introduced.

[33] C. E. McCulloch and J. M. Neuhaus, "Misspecifying the shape of a random effects distribution: Why getting it wrong may not matter," *Statist. Sci.*, vol. 26, pp. 388–402, 2011.
[34] L. Tierney and J. B. Kadane, "Accurate approximations for posterior moments and marginal densities," *J. Amer. Statist. Assoc.*, vol. 81, no. 393, pp. 82–86, 1986.
[35] T. P. Minka, "Expectation propagation for approximate Bayesian inference," in *Proc. 17th Conf. Uncertainty Artif. Intell.*, 2001, pp. 362–369.
[36] M. Kuss and C. E. Rasmussen, "Assessing approximate inference for binary Gaussian process classification," *J. Mach. Learn. Res.*, vol. 6, no. Oct, pp. 1679–1704, 2005.
[37] F. Sigrist, "Gradient and Newton boosting for classification and regression," *Expert Syst. Appl.*, vol. 167, 2021, Art. no. 114080.
[38] G. Biau, B. Cadre, and L. Rouvière, "Accelerated gradient boosting," *Mach. Learn.*, vol. 108, no. 6, pp. 971–992, 2019.
[39] H. Lu, S. P. Karimireddy, N. Ponomareva, and V. Mirrokni, "Accelerating gradient boosting machine," 2019, *arXiv: 1903.08708*.
[40] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, vol. 87. Berlin, Germany; Springer Science & Business Media, 2004.
[41] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *Commun. ACM*, vol. 64, no. 3, pp. 107–115, 2021.
[42] A. J. Wyner, M. Olson, J. Bleich, and D. Mease, "Explaining the success of AdaBoost and random forests as interpolating classifiers," *J. Mach. Learn. Res.*, vol. 18, no. 48, pp. 1–33, 2017.
[43] M. Belkin, S. Ma, and S. Mandal, "To understand deep learning we need to understand kernel learning," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 541–549.
[44] M. Belkin, D. Hsu, S. Ma, and S. Mandal, "Reconciling modern machine-learning practice and the classical bias–variance trade-off," *Proc. Nat. Acad. Sci. USA*, vol. 116, no. 32, pp. 15 849–15 854, 2019.
[45] P. L. Bartlett, P. M. Long, G. Lugosi, and A. Tsigler, "Benign overfitting in linear regression," *Proc. Nat. Acad. Sci. USA*, vol. 117, no. 48, pp. 30 063–30 070, 2020.
[46] Q. Liu and D. A. Pierce, "A note on gauss–hermite quadrature," *Biometrika*, vol. 81, no. 3, pp. 624–629, 1994.
[47] G. Guennebaud *et al.*, "Eigen v3," 2010. [Online]. Available: http://eigen.tuxfamily.org
[48] G. Ke *et al.*, "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 3149–3157.
[49] J. Friedman *et al.*, "Additive logistic regression: A statistical view of boosting (with discussion and a rejoinder by the authors)," *Ann. Statist.*, vol. 28, no. 2, pp. 337–407, 2000.
[50] B. Hofner, A. Mayr, N. Robinzonov, and M. Schmid, "Model-based Boosting in R: A hands-on tutorial using the R package mBoost," *Comput. Statist.*, vol. 29, pp. 3–35, 2014.
[51] J. Elith *et al.*, "Presence-only and presence-absence data for comparing species distribution modeling methods," *Biodiversity Informat.*, vol. 15, no. 2, pp. 69–80, 2020.
[52] A. Rabinowicz and S. Rosset, "Cross-validation for correlated data," *J. Amer. Statist. Assoc.*, pp. 1–14, 2020. [Online]. Available: https://doi.org/10.1080/01621459.2020.1801451

Fabio Sigrist received the MSc degree in Mathematics and the PhD degree in Statistics from the Swiss Federal Institute of Technology in Zurich (ETH Zurich). He is Professor of Applied Statistics and Data Science at the Lucerne University of Applied Sciences and Arts, Switzerland.