Latent Gaussian Model Boosting

Fabio Sigrist*
Lucerne University of Applied Sciences and Arts

May 20, 2021

Abstract

Latent Gaussian models and boosting are widely used machine learning techniques. Tree-boosting shows excellent predictive accuracy on many data sets, but potential drawbacks are that it assumes conditional independence of samples, produces discontinuous predictions for, 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 that allow for making probabilistic predictions. However, existing latent Gaussian models usually assume either a zero or a linear prior mean function. 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 predictive accuracy compared to existing approaches in both simulated and real-world data experiments.

1 Introduction

Boosting [Freund and Schapire, 1996, Friedman, 2001] is a machine learning technique that achieves state-of-the-art predictive accuracy [Chen and Guestrin, 2016]. 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" [Johnson and Zhang, 2013]. In boosting, and in many other supervised machine learning algorithms, it is assumed that a potentially complex 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 classes of latent Gaussian models are Gaussian process [Williams and Rasmussen, 2006] and grouped, or clustered, random effects models [Pinheiro and Bates, 2006]. For instance, Gaussian process models are used for modeling, time series, spatial,

*Email: fabio.sigrist@hslu.ch. Address: Lucerne University of Applied Sciences and Arts, Suurstoffi 1, 6343 Rotkreuz, Switzerland.

and spatio-temporal data. Further, grouped random effects models are used for modeling data with a potentially nested or crossed 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 prior models have the advantage that they are probabilistic models which allows for making probabilistic predictions. A drawback of existing latent Gaussian models is that the prior mean is often assumed to be either zero or to consist of a linear function of predictor variables. Both the zero-mean and the linearity assumption can be unrealistic, and higher predictive 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 by an ensemble of base learners, such as regression trees [Breiman et al., 1984], learned in a stage-wise manner using a boosting algorithm, 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 to relax both the independence assumption in boosting and the linearity assumption in latent Gaussian models in a flexible non-parametric way. Further, it allows to obtain 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 can be interpreted 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 predictive accuracy compared to both standard independent boosting 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 result in an estimate $\hat{F}(\cdot)$ that has both low bias and low variance. Intuitively, a low bias estimator $\hat{F}(\cdot)$ can have high variance if the complexity of the function $F(\cdot)$ is large in relation to the sample size. Examples of this 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 from of regularization in order to avoid high variance in $\hat{F}(\cdot)$. General-purpose regularization options for boosting include early stopping 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 can be interpreted as applying a form 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 [Tobler, 1970]. 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. Tree-boosting, on the other hand, cannot fully incorporate this prior information due to the discontinuity of trees. 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 prior 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 regularization in boosting algorithms.

Intuitively, we conjecture that the improvement in predictive 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 in relation 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, such as boosting and random forest, include Hajjem et al. [2011], Sela and Simonoff [2012], Fu and Simonoff [2015], Hajjem et al. [2014], and Sigrist [2020].

For non-Gaussian data, there exists very little prior work on combining non-linear machine learning methods, such as tree-based algorithms, with latent Gaussian models. For the special case of grouped random effects, Hajjem et al. [2017], Fokkema et al. [2018], and Speiser et al. [2020] propose algorithms that use regression trees to model the function $F(\cdot)$. Speiser et al. [2019] and Pellagatti et al. [2020] extend these algorithms by replacing trees with random forests. All of these methods are very heuristically motivated. In particular, it is unclear which objective functions these algorithms minimize[1] and whether and to which values they converge.

A straightforward alternative 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 $F(\cdot)$. A special example of this is the approach Hothorn et al. [2010] 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 discontinuity 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,

[1] They do not maximize a marginal or approximate marginal log-likelihood neither in a component-wise way nor using an EM algorithm.

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.

Note that the linearity assumption in mixed effects models can also be relaxed by using splines or generalized additive models [Hastie and Tibshirani, 1986, Wood, 2017] for modeling the predictor function $F(\cdot)$; see e.g. Tutz and Reithinger [2007] and Groll and Tutz [2012]. 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 $y = (y_1, \dots, y_n)^T \in \mathbb{R}^n$ 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 $\xi \in \Xi \subset \mathbb{R}^r$. Examples of $p(y|\mu, \xi)$ include Bernoulli and Gamma densities for binary classification and Gamma regression. Further, the parameter μ is related to the sum of a deterministic 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 $F(\cdot) : \mathbb{R}^p \to \mathbb{R}$, $F(X) = (F(X_1), \dots, F(X_n))^T$, where $X_i = (X_{i1} \dots, X_{ip})^T \in \mathbb{R}^p$ is the i-th row of $X \in \mathbb{R}^{n \times p}$ containing predictor variables for observation i, $i = 1, \dots, n$. In addition, 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 auxiliary parameters.

We assume that $F(\cdot)$ is a function in a function space $\mathcal{H}$ which 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 can consist of, e.g., linear functions [Bühlmann et al., 2006], smoothing splines [Bühlmann and Yu, 2003], wavelets [Saberian et al., 2011], reproducing kernel Hilbert space (RKHS) regression functions [Sigrist, 2021b], and regression trees [Breiman et al., 1984], 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 [Gelfand et al., 2003]. 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 deterministic 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 predictive accuracy. Moreover, existing boosting algorithms assume that the data y is independent conditional on $F(X)$ and thus ignore any potential residual correlation. Further, in existing latent Gaussian models, it is usually assumed that $F(X) = X\beta$ is a linear function.

Note that 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 [2020]. Further, for notational simplicity, we assume that only one parameter μ of the data distribution 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 variable is straightforward.

As a side note, 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 [McCulloch and Neuhaus, 2011]. Further, 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.

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

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

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 algorithm 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 $-\log(p(y|F, \theta, \xi))$. 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 [Tierney and Kadane, 1986] 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) [Minka, 2001]. Depending on the data distribution, for instance, for binary classification, this can lead to more accurate approximations [Kuss and Rasmussen, 2005], but it is computationally more demanding than the Laplace approximation. Nickisch and Rasmussen [2008], e.g., report that EP is ten times slower 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

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

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 $-\log(p(y|F,\theta,\xi))$ is given by

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

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 $p(b|y,\theta,\xi)$:

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

2.2.1 Gradients

For the boosting algorithm introduced in the following, we need to have expressions for $\frac{\partial L^{LA}(y,F,\theta,\xi)}{\partial F}$, $\frac{\partial L^{LA}(y,F,\theta,\xi)}{\partial \theta}$, and $\frac{\partial L^{LA}(y,F,\theta,\xi)}{\partial \xi}$. These are obtained as follows.

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

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

$$\begin{aligned}\frac{\partial L^{LA}(y,F,\theta,\xi)}{\partial \theta_k} =& -\frac{1}{2}\tilde{b}^T\Sigma^{-1}\frac{\partial \Sigma}{\partial \theta_k}\Sigma^{-1}\tilde{b} + \frac{1}{2}tr\left(\left(\Sigma + (Z^T\tilde{W}Z)^{-1}\right)^{-1}\frac{\partial \Sigma}{\partial \theta_k}\right) \\ &+ \left(\frac{\partial L^{LA}(y,F,\theta,\xi)}{\partial \tilde{b}}\right)^T\frac{\partial \tilde{b}}{\partial \theta_k}, \qquad k = 1,\ldots,q,\end{aligned} \tag{9}$$

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

where

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

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

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

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

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

Proof of Proposition 2.1. The derivation is similar as in Williams and Rasmussen [2006, 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

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

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

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

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

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

with respect to F_i and obtain

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

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 i-th 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, for computational efficiency and stability, we use different versions of the above result depending on the specific latent Gaussian model. If Zb consists of only grouped random effects, we use the version presented in Proposition 2.1 except that in (9), we replace $\left(\Sigma + (Z^T\tilde{W}Z)^{-1}\right)^{-1} \frac{\partial\Sigma}{\partial\theta_k}$ with the equivalent expression $\left(Z^T\tilde{W}Z + \Sigma^{-1}\right)^{-1}\Sigma^{-1}\frac{\partial\Sigma}{\partial\theta_k}Z^T\tilde{W}Z$. 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

$$\left(Z^T\tilde{W}Z + \Sigma^{-1}\right)^{-1} = \Sigma - \Sigma Z^T\tilde{W}^{1/2}\left(I_m + \tilde{W}^{1/2}Z\Sigma Z^T\tilde{W}^{1/2}\right)^{-1}\tilde{W}^{1/2}Z\Sigma,$$

factorize the matrix $I_m + \tilde{W}^{1/2}Z\Sigma Z^T\tilde{W}^{1/2}$, and, similarly as in Williams and Rasmussen [2006, Chapter 5.5.1], adapt all calculations in Proposition 2.1 accordingly.

3 Latent Gaussian Model Boosting

We propose to do the minimization 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 $F_{m-1}(\cdot)$:

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

where $f_m(\cdot)$, $m = 1, \ldots, M$, is chosen in a way 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 [2021a] 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 $(F_{m-1}(\cdot), I_{X_i}(\cdot))$ and (X_i), $i = 1, \ldots, n$, 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 [2021a] for more information. It is easily seen that the negative Gâteaux derivative of $R^A(F(\cdot), \theta, \xi)$ evaluated at $(F_{m-1}(\cdot), I_{X_i}(\cdot))$ and (X_i) is given by the vector $-\frac{\partial L_m^A(y, F_{m-1}, \theta, \xi)}{\partial F}$. This means that, $f_m(\cdot)$ can be found as a least squares approximation:

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

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 predictive accuracy [Friedman, 2001]. Further, functional gradient descent can also be accelerated using momentum. For instance, Biau et al. [2019] and Lu et al. [2019] propose to use Nesterov acceleration [Nesterov, 2004] 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 reason for this choice of optimization algorithm is outlined in Sigrist [2020].The LaGaBoost Algorithm 1 summarizes our approach.

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 (accelerated) first- or second-order method for convex optimization initialized with $(\theta_{m-1}, \xi_{m-1})$
4: Find $f_m(\cdot) = \underset{f(\cdot) \in \mathcal{S}}{\text{argmin}} \left\| -\frac{\partial L^A(y, F_{m-1}, \theta_m, \xi_m)}{\partial F} - f \right\|^2$
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. I.e., 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, note that 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 required for the gradients in Proposition 2.1 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 while at the same time having excellent generalization properties [Zhang et al., 2017, Wyner et al., 2017, Belkin et al., 2018, 2019]. 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 data. 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 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. Note that when k-fold cross-validation is used, both the function $F(\cdot)$ and the parameters θ and ξ are learned using the full data.

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
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. skipping line 3 in Algorithm 1, to obtain $\hat{F}(\cdot)$

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 observed and latent random variables for which predictions should be made. We have

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

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.

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

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

where

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

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., Williams and Rasmussen, 2006, Chapter 3.9], but for other likelihoods, it needs to be numerically approximated.

3.3 Software Implementation

The LaGaBoost and LaGaBoostOOS algorithms based on the Laplace approximation are implemented in C++ in the `GPBoost` library; see `https://github.com/fabsig/GPBoost` for more information. For linear algebra calculations, we rely on the `Eigen` library [Guennebaud et al., 2010]. 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 [Ke et al., 2017].

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: $y_i \in \{0,1\}$, $P(y_i = 1) = \Phi(\mu_i)$, $i = 1,\dots,n$, 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:

$$F(x) = C_1 + C_2 \cdot \tan^{-1}\left(\frac{x_2x_3 - 1 - \frac{1}{x_2x_4}}{x_1}\right),$$

where $x = (x_1, x_2, x_3, x_4)^T$, $x_1 \sim Unif(0, 100)$, $x_2 \sim Unif(40\pi, 560\pi)$, $x_3 \sim Unif(0, 1)$, and $x_4 \sim Unif(1, 11)$. This function, which we denote by 'friedman3', was first used in Friedman [1991] and has since then often been used to compare non-parametric regression models. 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 with $m = 500$ different groups. I.e., 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 $[0.5, 1]^2$. Figure 1 illustrates this.

Figure 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.

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 boosting

for binary data with the log loss ('LogitBoost') [Friedman et al., 2000], and model-based boosting ('mboost') [Hothorn et al., 2010] with the log loss, i.e. a Bernoulli negative 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 for 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 0.3.0 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 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`[2] since the number of categories is large. The results are slightly worse when considering the grouping variable as a categorical variable (results not tabulated). Concerning the mboost algorithm, we use the `mboost` R package [Hofner et al., 2014] 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 values 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 $M \in \{1, \dots, 1000\}$, the learning rate $\nu \in \{0.1, 0.05, 0.01\}$, the maximal depth of the trees $\in \{1, 5, 10\}$, 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 calculate average test error rates and test log losses for both the "interpolation" and "extrapolation" 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 predictive 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. Concerning predictive accuracy, we observe no clear differences between the LaGaBoost and the LaGaBoostOOS algorithms. However, as expected, the hyperparameter estimates of the LaGaBoostOOS algorithm have smaller RMSEs and biases. Further, in the appendix, we also perform the same simulated experiments for two additional predictor functions $F(\cdot)$: another non-linear function and a linear function. The results for

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

the other non-linear function are qualitatively very similar to the ones reported here. Not surprisingly, a linear latent Gaussian model performs best when the true function is linear.

	LaGaBoost	LinearME	LogitBoost	mboost	LaGaBoostOOS
Error	**0.2369**	0.292	0.3377	0.2973	0.237
(sd)	(0.00807)	(0.00808)	(0.00887)	(0.00875)	(0.00818)
[p-val]		[4e-93]	[6.1e-100]	[1.9e-84]	[0.55]
Log_loss	**2421**	2823	3037	2822	2420
(sd)	(44.1)	(42.7)	(29)	(23.4)	(44.6)
[p-val]		[9.2e-118]	[3.6e-113]	[4.8e-107]	[1.1e-07]
Error_ext	**0.342**	0.4081	0.3521		0.3421
(sd)	(0.00899)	(0.0135)	(0.0225)		(0.0089)
[p-val]		[8.3e-81]	[9.9e-06]		[0.83]
Log_loss_ext	**3068**	3379	3104		3067
(sd)	(28.3)	(25.1)	(63.1)		(28.7)
[p-val]		[1.2e-111]	[2e-08]		[0.013]
RMSE_sigma2	0.1696	0.3528			0.1534
Bias_sigma2	-0.1506	-0.3476			-0.1322
Time (s)	1.589	1.666	0.06444	15.96	4.947

Table 1: Results for the grouped / high-cardinality data and the mean function F = 'friedman3'. The results for the "extrapolation" test data sets are denoted by '_ext'.

	LaGaBoost	LinearGP	LogitBoost	mboost	LaGaBoostOOS
Error	**0.2907**	0.3308	0.3312	0.3734	0.29
(sd)	(0.0263)	(0.0303)	(0.0321)	(0.0328)	(0.0255)
[p-val]		[4.8e-31]	[4.5e-30]	[3.7e-43]	[0.26]
Log_loss	**281.6**	303.3	304.5	331.5	280.8
(sd)	(13.7)	(12.9)	(11.7)	(5.98)	(14)
[p-val]		[1.5e-44]	[3.3e-46]	[1.6e-65]	[3.8e-18]
Error_ext	**0.3433**	0.3915	0.3699	0.4152	0.3429
(sd)	(0.0556)	(0.0622)	(0.0689)	(0.0888)	(0.0543)
[p-val]		[5.3e-21]	[8.5e-09]	[5.5e-16]	[0.64]
Log_loss_ext	**308.6**	330.7	322.1	3552	308.4
(sd)	(19.5)	(19.1)	(30.1)	(4100)	(19.1)
[p-val]		[1.3e-34]	[9.8e-09]	[3.5e-12]	[0.24]
RMSE_sigma2	0.5742	0.4525			0.478
RMSE_rho	0.07181	0.1323			0.06007
Bias_sigma2	-0.5465	-0.3734			-0.4112
Bias_rho	0.04654	0.07052			0.03293
Time (s)	64.73	43.99	0.01145	0.8435	80.51

Table 2: Results for the spatial data and the mean function F = 'friedman3'. The results for the "extrapolation" test data sets are denoted by '_ext'.

4.1 When Does the LaGaBoost Algorithm Outperform Classical Independent Boosting?

It is relatively obvious that the LaGaBoost algorithm tends to outperform linear latent Gaussian models when the linearity assumption is violated. 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 assume independence among samples. Intuitively, we expect that the improvement in predictive accuracy of our novel approach over classical 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 with different 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, and 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.

Figure 2 reports the relative decrease in the test error of the LaGaBoost algorithm compared to the LogitBoost algorithm for the "interpolation" test data sets. The figure confirms our hypothesis that the improvement in predictive accuracy is the larger, the smaller the number of observations per group is and the faster the covariance decays over space. Or in other words, the higher the complexity of the underlying true function relative to the sample size, the larger is the improvement obtained by the LaGaBoost algorithm. Note that the relative outperformance of the LaGaBoost over the LogitBoost algorithm is almost the same for 10 and 20 samples per group. This is not surprising as in the extreme case with very little data per group, also random effects cannot learn the group effects well, and the accuracy of both approaches will thus be similar.

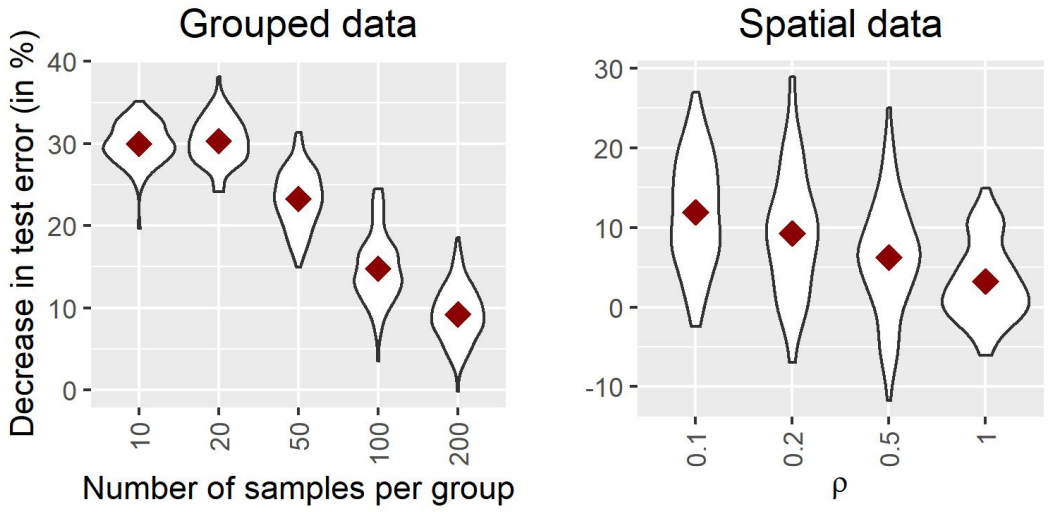

Figure 2: Relative decrease in test error of the LaGaBoost algorithm vs. the LogitBoost algorithm for spatial data with different ρ's and grouped data with varying number of samples per group. The red rhombi represent means.

5 Real-world Applications

In the following, we apply the LaGaBoost algorithm to two real-world binary classification data sets and compare its predictive accuracy to alternative approaches. We consider both a spatial data set and data with a grouped structure, i.e. data with a high-cardinality categorical variable. Concerning the former, 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 [Elith et al., 2020]. The goal is to predict the presence or absence of the species. For the latter, 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, and the goal is to predict the binary poverty indicator. Table 3 summarizes the data sets.

Table 3: Summary of data sets.

Name	Data type	Nb. samples	Freq. of 1's	Nb. features	Nb. categories
Poverty	Grouped / high-cardinality	5755	36.79 %	7 + 1	1151
Species	Spatial	909	23.76 %	13 + 2	

We compare the predictive accuracy of the different approaches using nested 4-fold cross-validation. For the poverty data set with the high-cardinality categorical variable, we perform stratified cross-validation such that each 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 algorithm, does not allow for making predictions for unobserved groups. We compare the LaGaBoost algorithm to the same alternative methods as in the simulation study in Section 4. Tuning parameters are chosen by doing an additional inner 4-fold cross-validation on every of the four training data sets. We consider the same set of tuning parameters and selection criterion as in the simulated experiments. In addition to the test error and the test loss, we also report the test area under the ROC curve (AUC).

The results are reported in Tables 4 and 5. We find that the LaGaBoost algorithm outperforms all alternative methods in all three predictive accuracy metrics for both the spatial and the grouped data, i.e. the high-cardinality categorical variable data. Further, for the spatial data, we observe that when also including the coordinates in the predictor variables X ('LaGaBoost_IC'), the LaGaBoost algorithm performs slightly worse compared to when not including the coordinates in the predictor variables X ('LaGaBoost'). This is an indication that there are no interactions among the locations and the other predictor variables.

	LaGaBoost	LaGaBoost_IC	LinearGP	LogitBoost	mboost
Error	**0.2255**	0.231	0.2398	0.2409	0.2387
AUC	**0.7714**	0.7707	0.7412	0.7382	0.7317
Log_loss	**0.4636**	0.4639	0.4882	0.485	0.4903

Table 4: Comparison of predictive accuracy for the species data. 'LaGaBoost_IC' denotes results for the LaGaBoost algorithm when also including the coordinates in the predictor function $F(\cdot)$.

	LaGaBoost	LinearME	LogitBoost	mboost
Error	**0.2791**	0.2831	0.3211	0.3131
AUC	**0.7275**	0.7227	0.7037	0.7154
Log_loss	**0.5825**	0.586	0.6017	0.5995

Table 5: Comparison of predictive accuracy for the poverty data.

6 Conclusion

We have introduced a novel way for combining latent Gaussian models, such as Gaussian processes and random effects models, with boosting and have obtained increased predictive 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 the predictive accuracy of the LaGaBoost algorithm. Also, it would be interesting to compare the LaGaBoost algorithm to alternative approaches on further data sets, e.g., for other data distributions.

References

M. Belkin, S. Ma, and S. Mandal. To Understand Deep Learning We Need to Understand Kernel Learning. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 541–549, 2018.

M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

G. Biau, B. Cadre, and L. Rouvière. Accelerated gradient boosting. *Machine Learning*, 108(6):971–992, 2019.

L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

P. Bühlmann and B. Yu. Boosting with the L2 loss: regression and classification. *Journal of the American Statistical Association*, 98(462):324–339, 2003.

P. Bühlmann et al. Boosting for high-dimensional linear models. *The Annals of Statistics*, 34(2):559–583, 2006.

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

J. Elith, C. Graham, R. Valavi, M. Abegg, C. Bruce, A. Ford, A. Guisan, R. J. Hijmans, F. Huettmann, L. Lohmann, et al. Presence-only and Presence-absence Data for Comparing Species Distribution Modeling Methods. *Biodiversity Informatics*, 15(2):69–80, 2020.

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. *Behavior research methods*, 50(5):2016–2034, 2018.

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *ICML*, volume 96, pages 148–156. Bari, Italy, 1996.

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

J. H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, pages 1–67, 1991.

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

W. Fu and J. S. Simonoff. Unbiased regression trees for longitudinal and clustered data. *Computational Statistics & Data Analysis*, 88:53–74, 2015.

A. E. Gelfand, H.-J. Kim, C. Sirmans, and S. Banerjee. Spatial modeling with spatially varying coefficient processes. *Journal of the American Statistical Association*, 98(462): 387–396, 2003.

A. Groll and G. Tutz. Regularization for generalized additive mixed models by likelihood-based boosting. *Methods of Information in Medicine*, 51(02):168–177, 2012.

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

A. Hajjem, F. Bellavance, and D. Larocque. Mixed effects regression trees for clustered data. *Statistics & probability letters*, 81(4):451–459, 2011.

A. Hajjem, F. Bellavance, and D. Larocque. Mixed-effects random forest for clustered data. *Journal of Statistical Computation and Simulation*, 84(6):1313–1328, 2014.

A. Hajjem, D. Larocque, and F. Bellavance. Generalized mixed effects regression trees. *Statistics & Probability Letters*, 126:114–118, 2017.

T. Hastie and R. Tibshirani. Generalized Additive Models. *Statistical Science*, 1(3):297–310, 1986.

B. Hofner, A. Mayr, N. Robinzonov, and M. Schmid. Model-based Boosting in R: A Hands-on Tutorial Using the R Package mboost. *Computational Statistics*, 29:3–35, 2014.

T. Hothorn, P. Bühlmann, T. Kneib, M. Schmid, and B. Hofner. Model-based boosting 2.0. *Journal of Machine Learning Research*, 11(Aug):2109–2113, 2010.

R. Johnson and T. Zhang. Learning nonlinear functions using regularized greedy forest. *IEEE transactions on pattern analysis and machine intelligence*, 36(5):942–954, 2013.

G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3149–3157, 2017.

M. Kuss and C. E. Rasmussen. Assessing approximate inference for binary Gaussian process classification. *Journal of machine learning research*, 6(Oct):1679–1704, 2005.

H. Lu, S. P. Karimireddy, N. Ponomareva, and V. Mirrokni. Accelerating gradient boosting machine. *arXiv preprint arXiv:1903.08708*, 2019.

C. E. McCulloch and J. M. Neuhaus. Misspecifying the shape of a random effects distribution: why getting it wrong may not matter. *Statistical science*, pages 388–402, 2011.

T. P. Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 362–369, 2001.

Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2004.

H. Nickisch and C. E. Rasmussen. Approximations for binary Gaussian process classification. *Journal of Machine Learning Research*, 9(Oct):2035–2078, 2008.

M. Pellagatti, C. Masci, F. Ieva, and A. M. Paganoni. Generalized Mixed Effects Random Forest: a flexible approach to predict university student dropout. *MOX-Report*, 36/2020, 2020.

J. Pinheiro and D. Bates. *Mixed-effects models in S and S-PLUS*. Springer Science & Business Media, 2006.

M. J. Saberian, H. Masnadi-Shirazi, and N. Vasconcelos. Taylorboost: First and second-order boosting algorithms with explicit margin control. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2929–2934. IEEE, 2011.

R. J. Sela and J. S. Simonoff. RE-EM trees: a data mining approach for longitudinal and clustered data. *Machine learning*, 86(2):169–207, 2012.

F. Sigrist. Gaussian Process Boosting. *arXiv preprint arXiv:2004.02653*, 2020.

F. Sigrist. Gradient and newton boosting for classification and regression. *Expert Systems With Applications*, 167:114080, 2021a.

F. Sigrist. KTBoost: Combined kernel and tree boosting. *Neural Processing Letters*, 53(2): 1147–1160, 2021b.

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 and Intelligent Laboratory Systems*, 185:122–134, 2019.

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. *Communications in Statistics-Simulation and Computation*, 49(4):1004–1023, 2020.

L. Tierney and J. B. Kadane. Accurate approximations for posterior moments and marginal densities. *Journal of the american statistical association*, 81(393):82–86, 1986.

W. R. Tobler. A computer movie simulating urban growth in the Detroit region. *Economic geography*, 46(sup1):234–240, 1970.

G. Tutz and F. Reithinger. A boosting approach to flexible semiparametric mixed models. *Statistics in medicine*, 26(14):2872–2900, 2007.

C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*. MIT Press Cambridge, MA, 2006.

S. N. Wood. *Generalized additive models: an introduction with R*. Chapman and Hall/CRC, 2017.

A. J. Wyner, M. Olson, J. Bleich, and D. Mease. Explaining the success of adaboost and random forests as interpolating classifiers. *Journal of Machine Learning Research*, 18 (48):1–33, 2017.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

A Appendix

Results for the Simulated Experiments for Two Additional Predictor Functions

We also perform the same simulated experiments as in Section 4 for the following two additional predictor functions $F(\cdot)$:

$$
\begin{aligned}
F(x) &= C_1 + C_2 \cdot (2x_1 + x_2^2 + 4 \cdot 1_{\{x_3>0\}} + 2\log(|x_1|)x_3), \quad (\text{'hajjem'}),\\
&\qquad x = (x_1, \ldots, x_9)^T, \quad x \sim \mathcal{N}(0, I_9),\\
F(x) &= C_1 + C_2 \cdot (x_1 + x_2), \quad (\text{'linear'}),\\
&\qquad x = (x_1, x_2)^T, \quad x_1, x_2 \overset{iid}{\sim} Unif(0,1).
\end{aligned}
$$

The function 'hajjem' has been used in Hajjem et al. [2014] to compare non-parametric mixed effects models for Gaussian data. We also include a linear function in order to investigate how our approach compares to a latent Gaussian model with a linear function $F(\cdot)$ when the data generating process is linear. As in the simulated experiments in Section 4, 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. The results for these two functions $F(\cdot)$ are reported in the following tables.

	LaGaBoost	LinearME	LogitBoost	mboost	LaGaBoostOOS
Error	**0.2366**	0.2866	0.3389	0.2802	0.2371
(sd)	(0.00683)	(0.00802)	(0.00791)	(0.0112)	(0.00677)
[p-val]		[1.6e-88]	[1.3e-106]	[5.8e-68]	[0.015]
Log_loss	**2417**	2783	3023	2915	2419
(sd)	(38.2)	(44.6)	(31.3)	(22.8)	(38.7)
[p-val]		[1.7e-101]	[3.4e-116]	[9e-121]	[0.00026]
Error_ext	**0.3429**	0.4186	0.3535		0.3426
(sd)	(0.00727)	(0.00866)	(0.0147)		(0.00748)
[p-val]		[5e-94]	[1e-11]		[0.26]
Log_loss_ext	**3028**	3296	3081		3030
(sd)	(28.5)	(28.8)	(67.8)		(28.1)
[p-val]		[1.1e-95]	[5.4e-14]		[0.0017]
RMSE_sigma2	0.2105	0.3594			0.2074
Bias_sigma2	-0.198	-0.3528			-0.1953
Time (s)	1.172	0.04448	0.03739	17	5.583

Table A.1: Results for the grouped / high-cardinality data and the mean function F = 'hajjem'. The results for the "extrapolation" test data sets are denoted by '_ext'.

	LaGaBoost	LinearGP	LogitBoost	mboost	LaGaBoostOOS
Error	**0.318**	0.3338	0.3493	0.397	0.3144
(sd)	(0.029)	(0.0269)	(0.0302)	(0.0338)	(0.0289)
[p-val]		[7.7e-14]	[3e-22]	[2.4e-43]	[0.079]
Log_loss	**296.2**	304.4	311.9	336.5	294.4
(sd)	(11.6)	(12.6)	(11.4)	(5.43)	(12.5)
[p-val]		[2.9e-28]	[2.2e-31]	[6.8e-62]	[0.0059]
Error_ext	**0.3717**	0.3921	0.3906	0.4286	0.3699
(sd)	(0.0438)	(0.039)	(0.0507)	(0.0856)	(0.0447)
[p-val]		[2.1e-14]	[2.3e-07]	[3.8e-12]	[0.41]
Log_loss_ext	**320.1**	328.2	330.9	3158	319.6
(sd)	(15.4)	(15.4)	(21.8)	(4050)	(14.9)
[p-val]		[1.2e-17]	[2.3e-12]	[2.9e-10]	[0.46]
RMSE_sigma2	0.5943	0.4622			0.5039
RMSE_rho	0.05778	0.05834			0.05285
Bias_sigma2	-0.5659	-0.375			-0.4451
Bias_rho	0.02944	0.02185			0.02064
Time (s)	12.91	3.765	0.02225	0.7543	32.29

Table A.2: Results for the spatial data and the mean function F = 'hajjem'. The results for the "extrapolation" test data sets are denoted by '_ext'.

	LaGaBoost	LinearME	LogitBoost	mboost	LaGaBoostOOS
Error	0.2219	**0.2168**	0.3074	0.2771	0.222
(sd)	(0.0065)	(0.00638)	(0.007)	(0.0068)	(0.0064)
[p-val]		[2e-30]	[5.6e-102]	[3.1e-94]	[0.8]
Log_loss	2296	**2246**	2904	2759	2295
(sd)	(36)	(39.6)	(30.3)	(23.7)	(36.2)
[p-val]		[4.1e-70]	[6.1e-120]	[9.6e-118]	[6.2e-42]
Error_ext	0.3085	**0.3049**	0.3162		0.3086
(sd)	(0.0078)	(0.00762)	(0.0131)		(0.00775)
[p-val]		[7.5e-12]	[3.2e-10]		[0.12]
Log_loss_ext	2900	**2872**	2947		2900
(sd)	(35.9)	(38.9)	(67.8)		(35.9)
[p-val]		[2.7e-33]	[1.6e-12]		[1.5e-14]
RMSE_sigma2	0.193	0.1165			0.1767
Bias_sigma2	-0.1785	-0.01651			-0.1617
Time (s)	8.907	0.244	0.06671	19.28	47.37

Table A.3: Results for the grouped / high-cardinality data and the mean function F = 'linear'. The results for the "extrapolation" test data sets are denoted by '_ext'.

	LaGaBoost	LinearGP	LogitBoost	mboost	LaGaBoostOOS
Error	0.2777	**0.2586**	0.3125	0.3786	0.2767
(sd)	(0.0219)	(0.0226)	(0.023)	(0.0234)	(0.0209)
[p-val]		[8.6e-26]	[5.8e-35]	[4.6e-67]	[0.27]
Log_loss	270.9	**256.4**	292.7	329.4	269.2
(sd)	(11.9)	(12.6)	(9.7)	(4.79)	(12.2)
[p-val]		[2.6e-47]	[1.2e-50]	[1.6e-75]	[7.5e-17]
Error_ext	0.3192	**0.2975**	0.3402	0.4295	0.3191
(sd)	(0.0362)	(0.0307)	(0.0375)	(0.0843)	(0.0348)
[p-val]		[7e-22]	[4.4e-10]	[8.4e-26]	[0.93]
Log_loss_ext	296.4	**283.9**	308.4	3509	295.7
(sd)	(19.2)	(17.8)	(19.8)	(4230)	(17.8)
[p-val]		[9.6e-35]	[1.8e-13]	[1.7e-11]	[0.094]
RMSE_sigma2	0.6238	0.5971			0.427
RMSE_rho	0.1072	0.09143			0.05435
Bias_sigma2	-0.5959	-0.5618			-0.3557
Bias_rho	0.0734	0.05386			0.02379
Time (s)	84.18	21.61	0.01426	1.141	221.1

Table A.4: Results for the spatial data and the mean function F = 'linear'. The results for the "extrapolation" test data sets are denoted by '_ext'.