Gaussian Process Boosting: Simulation Report

Yuqi YANG

Hong Kong University of Science and Technology

yyangfd@connect.ust.hk

Contents

1. Overview

1.1 Why is it interesting

Standard boosting algorithm excels at solving complex non-linear modeling problems while staying robust over various scenarios in supervised learning settings. However, boosting algorithms always assume the independence relationships across different response variables when the predictors are given.

While Gaussian process, on the other hand, often starts with a zero or linear function prior assumption. Gaussian process or mixed effects modeling techniques can account for the residual correlations by design, but the prior assumption sometimes can be not necessarily correct and lead to a degradation of performance.

Therefore, these pros and cons of two different models call for a combined effort to model the possible intricate non-linearity and capture the residual correlations at the same time. This would be useful, and predictably, will achieve superior performance over stand-alone models. The results from the paper also have shown the feasibility and State-of-the-Art prediction accuracy of the proposed methodology.

1.2 What are the major challenges to solve

To combine the two models, we need to consider how to relax the constraint imposed by them. More specifically, we need to come up with a solution to relax both the linear prior or zero prior setting in mixed effects models and the independence setting in boosting architecture.

There have been literature trying to solve the problem in a three-step fashion:

1. Use some learning algorithms to come up with a additive predictor function $f_{k+1}(\cdot)$
2. In every iteration, try to estimate the covariance parameters
3. In every iteration, try to estimate the random effects

These procedures would work. However, the cost of computation is high, since in each round both step two and three requires recursive estimation and there would be rooms to optimize and level up the efficiency.

1.3 How the proposed method solve the problem

1.3.1 Algorithm prototype

We formalize our goal to minimizing the following loss function by choosing a suitable modeling function $F(\cdot)$ from the function space $\mathcal{H}$ for the boosting component, and suitable parameter θ from the covariance parameter space Θ of the Gaussian process, a

negative log likelihood function is used as the loss function:

$$\hat{F}, \hat{\theta} = \arg\min_{F,\theta \in \mathcal{H},\Theta} L(y, F, \theta)\big|_{F=F(X)}$$

Algorithm 1 *Gaussian Process Boosting Prototype*

Input: Initialized θ_0, learning rate η, number of iterations K
Initialize $F_0() = \arg\min_{c \in \mathbb{R}} L(y, c \cdot \mathbb{I}, \theta_0)$
for $k = 1$ to K **do**
    Find $\theta_{k+1} = \arg\min_{\theta \in \Theta} L(y, F_k, \theta)$
    Conducting boosting step for finding the additive structure $f_{k+1}(\cdot)$
    Update $F_{k+1}(\cdot) = F_k(\cdot) + \eta f_{k+1}(\cdot)$
end for
return Predictor function $\hat{F}(\cdot)$, covariance parameters $\hat{\theta}$

1.3.2 Update rule in the Boosting step

When updating the predictor function $F(\cdot)$ in boosting, there are several choices available. One is using the Netwon step, which offers one-step of functional gradient followed by one-step optimization of parameters, and is therefore, efficient from the computational point of view:

$$f_{k+1}(\cdot) = \arg\min_{f(\cdot)} (y - F_k - f)^T \Sigma_{k+1}^{-1} (y - F_k - f)$$

but suffering from the overfitting problem, the boosting mechanisms often opt for an early stopping for the algorithm, which is not guaranteed to be optimally convergent.

The alternative counterpart, known as the block descent technique, will optimize the function then parameter in an iterative manner, which would impose a great demand of computation, if we try to optimize both the function and parameter until the convergence condition is matched for the stopping of optimization in one iteration:

$$f_{k+1}(\cdot) = \arg\min_{f(\cdot)} \|\Sigma_{k+1}^{-1}(F_k - y) - f\|^2$$

Hence, inspired by both techniques, we could mitigate the inefficiency by opting for a fused updating rule: we use the Newton step to approximate the gradient of the function $F(\cdot)$ and then do a block descent, or coordinate descent to update the parameter θ. In this manner, we would be more likely to achieve the optimal convergence while not compromising the overall efficiency.

1.3.3 Gaussian Process Boosting

By joining the updating rule into the algorithm prototype, hence we get the vanilla Gaussian Process Boosting algorithm.

Algorithm 2 *Gaussian Process Boosting*

Input: Initialized θ_0, learning rate η, number of iterations K
Initialize $F_0() = \arg\min_{c \in \mathbb{R}} L(y, c \cdot \mathbb{I}, \theta_0)$
for $k = 1$ to K **do**
    Find $\theta_{k+1} = \arg\min_{\theta \in \Theta} L(y, F_k, \theta)$
    Find $\beta_{k+1} = \hat{\beta}$, $(\hat{\beta}, \hat{\nu}) = \arg\min_{(\beta,\nu): f() = h(;\beta)^T \nu \in \mathcal{S}} \|\Sigma_{k+1}^{-1}(F_k - y) - f\|^2$
    Calculate $\nu_{k+1} = (h_{\beta_{k+1}}^T \Sigma_{k+1}^{-1} h_{\beta_{k+1}})^{-1} h_{\beta_{k+1}}^T \Sigma_{k+1}^{-1}(y - F_k)$
    Set $f_{k+1}() = h(;\beta_{k+1})^T \nu_{k+1}$
    Update $F_{k+1}(\cdot) = F_k(\cdot) + \eta f_{k+1}(\cdot)$
end for
return Predictor function $\hat{F}(\cdot)$, covariance parameters $\hat{\theta}$

2. Simulation Result

We conduct the simulation study based on the same setup as the simulated dataset mentioned in the Section 4.1[1] of the paper (Sigrist, 2022). The reproduced code can be found in the GitHub link provided below.[2]

2.1 Simulation Setting

We adopt the following mixed effects model with random effects Zb, both grouped mixed effects and Gaussian Process are incorporated under this setting:

$$y = F(X) + Zb + \epsilon, \quad b \sim \mathcal{N}(0, \Sigma), \quad \epsilon \sim \mathcal{N}(0, \sigma^2 I_n)$$

a sample size of $n = 500$ for spatial dataset generated is used. The number is chosen to be relatively small to enable exact calculations during intermediate steps without doing large scale approximations.

To generate the spatial data, we use a exponential variance Gaussian process model, where the covariance function $k(s, s')$ is given by

$$k(s, s') = \sigma_1^2 \exp\left(-\frac{\|s - s'\|}{\rho}\right)$$

here, the locations are constraint to be $[0, 1]^2$ and $\rho = 0.1$. $\sigma_1^2 = 1$ is set as the marginal variance, and $\sigma^2 = 1$ as the error variance. By doing this, we can control the signal-to-noise ratio between the random effects Zb and the error term ϵ to be 1.

For the fixed effects part, we consider the same simulation setting as the author referenced (Ahlem Hajjem & Larocque, 2014), where $F(x)$ is chosen to be

$$F(x) = C \cdot \left(2x_1 + x_2^2 + 4\mathbb{I}_{\{x_3 > 0\}} + 2\log(|x_1|x_3)\right), \quad x = (x_1, \ldots, x_9)^T, \quad x \sim \mathcal{N}(0, I_9)$$

[1] The overall methodology of the artificial derivation of the dataset is adopted while some of the hyper parameters could be different from the ones proposed in the paper.

[2] https://github.com/AragornBFRer/GPBoost

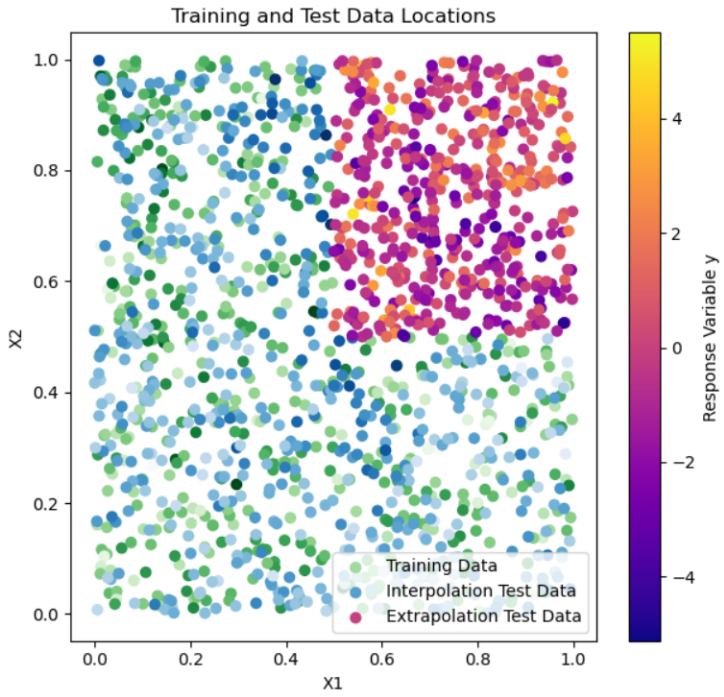

Figure 1: Demonstration of the simulated spatial data, where the *training* set is sampled uniformly from $[0,1]^2$ excluding $[0.5,1]^2$, so as the *interpolation* testing set. The *extrapolation* testing set is simulated from $[0.5,1]^2$, as shown in the reddish color.

here, C is a normalization constant to control the variance of $F(x)$ to be approximately 1. In other words, the signal strength of the fixed effects part and the mixed effects part are about the same.

The dataset are split into different areas for training and testing purposes. For convenience, we term the testing dataset located in the same spot as the training dataset as the *interpolation*, and the ones that are out of current training sample are termed as *extrapolation*. In the spatial data setting we choose, the training data are sampled uniformly from $[0,1]^2$ excluding $[0.5,1]^2$. And the *extrapolation* part are sampled uniformly from $[0.5,1]^2$, as illustrated in Figure 1. The reproduction and implementation of the data simulator can be found in the corresponding GitHub link.[3]

2.2 Empirical Results

We carry out the comparison between the performance of proposed GPBoost model and the performance of `LightGBM` (Ke et al., 2017) and `CatBoost` models.

As we can see from Figure 3, the GPBoost outperforms LightGBM and CatBoost under the similar hyper parameter setting, showcasing the effectiveness of this algorithm. The result stays robust outside of current trained regions.[4]

We additionally visualize the solution path of the GPBoost algorithm and CatBoost algorithm as a comparative study. As shown in the Figure 2 below, the convergence speed

[3] https://github.com/AragornBFRer/GPBoost/tree/main/code

[4] Note that the terming of *extrapolation* and *interpolation* is just for naming convenience, since it's "some sort of extrapolation" under spatial setting as the author suggests.

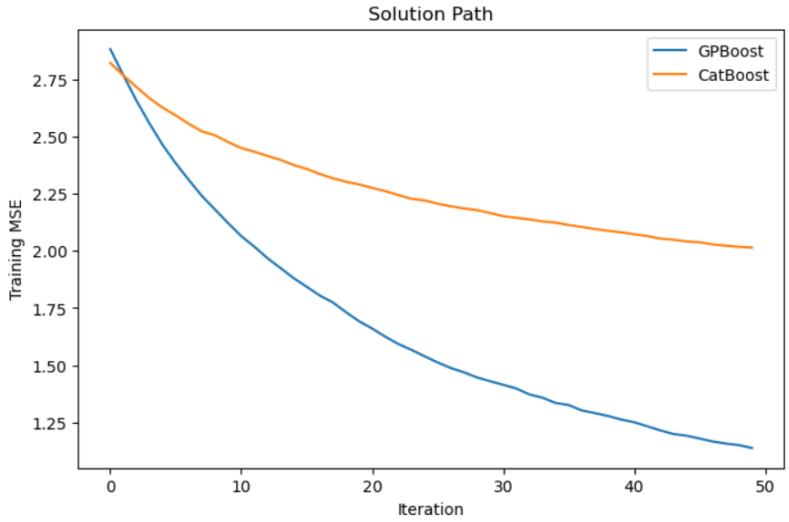

Figure 2: This is the visualization of the solution path of GPBoost and CatBoost under the same number of iterations $num\_iter = 50$.

of the proposed GPBoost algorithm is much faster than the CatBoost. This shows the superior efficiency of GPBoost under the hybrid updating step setting.

2.3 Ablation Study

To testify the effectiveness and necessity of the proposed algorithm, we can adjust the types of updating rule used in the boosting step to see if the performance would change. As a result, the hybrid update step achieves the best performance under the above setting, while the demonstrated result in Figure 4 and Figure 5, which used gradient algorithm and newton step for updating, appears to be worse. With hybrid step, the implemented algorithm can be better off compared to LightGBM and CatBoost model, while stand alone update rules may not.

3. Conclusion

The Gaussian Process Boosting model proposed in this paper cleverly combine the boosting structure with the mixed effects model to achieve better results for both individual models. The idea of relaxing both side of the assumptions to combine the models, i.e., relaxing the independence assumption in the boosting algorithm by introducing the correlated structures from the Gaussian Process model and relaxing the linear prior assumption of the mixed effects model via introducing non-linearity from the additive structure of boosting methods. This opens another realm of research thinking for me, to rethink the pros and cons of of each model from the bottom up, and exploit full potential of existing methods.

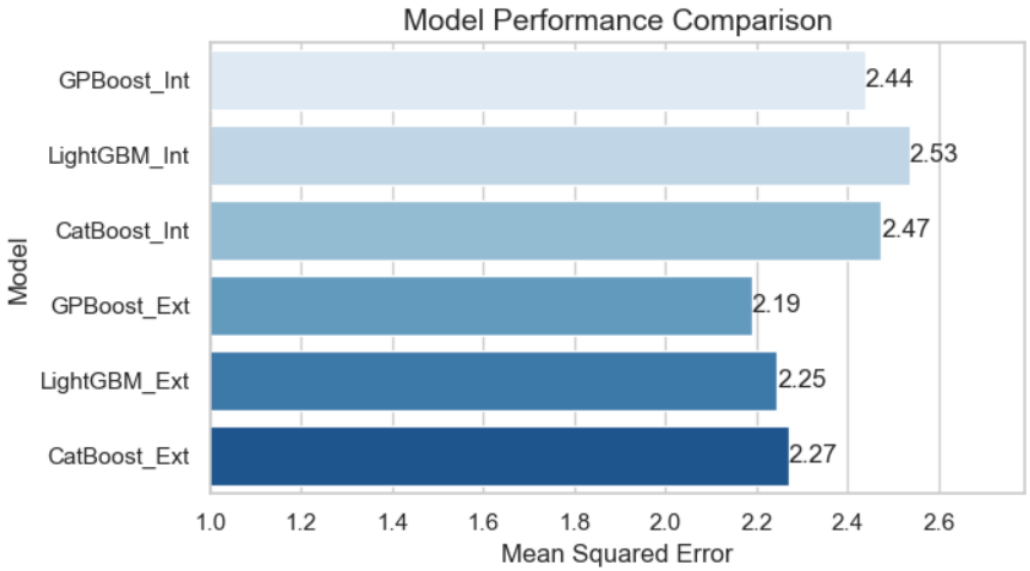

Figure 3: Demonstration of the performance of GPBoost model with a hybrid of gradient algorithm and newton step used in the boosting step, measured in MSE. Where *_Int* stands for interpolation and *_Ext* stands for extrapolation.

4. Disclaimer

I have used ChatGPT to facilitate my reproduction of the code. But I have not referred to other code sources.

References

Ahlem Hajjem, F. B., & Larocque, D. (2014). Mixed-effects random forest for clustered data. *Journal of Statistical Computation and Simulation*, 84(6), 1313–1328. https://doi.org/10.1080/00949655.2012.741599

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 30). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

Sigrist, F. (2022). Gaussian process boosting. *Journal of Machine Learning Research*, 23(232), 1–46. http://jmlr.org/papers/v23/20-322.html

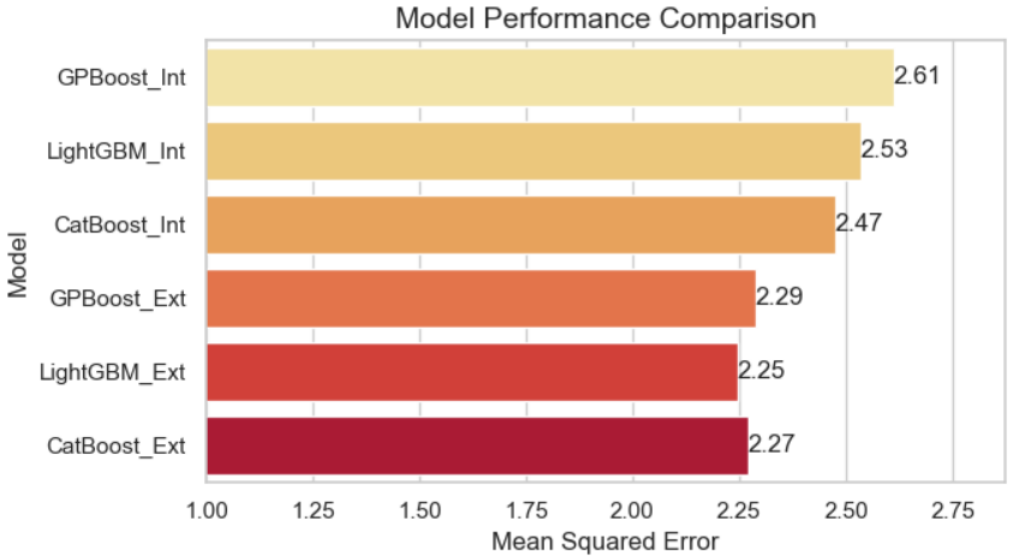

Figure 4: Demonstration of the performance of GPBoost model with gradient algorithm used in the boosting step, measured in MSE. Where $\_Int$ stands for interpolation and $\_Ext$ stands for extrapolation.

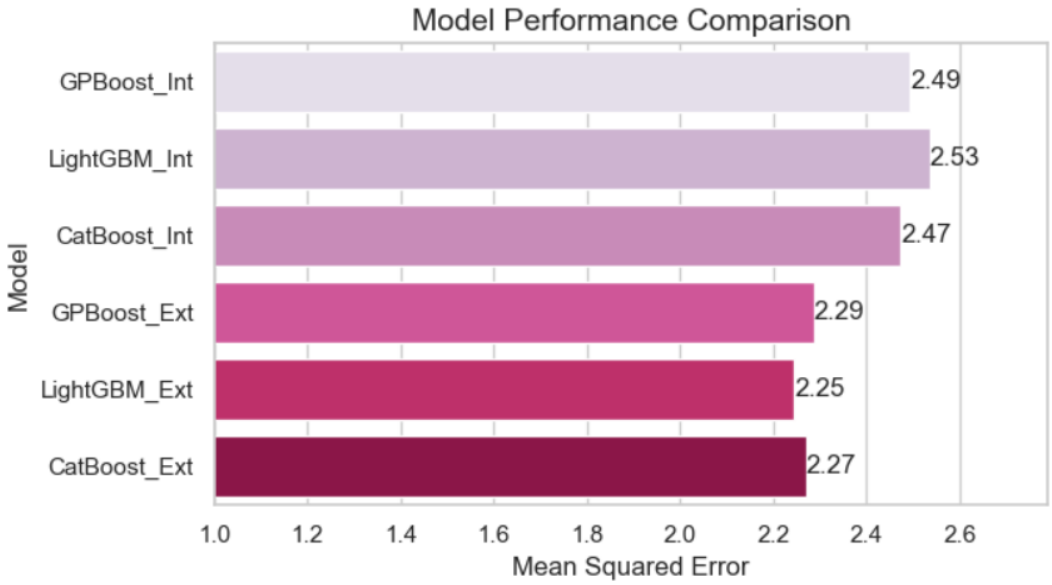

Figure 5: Demonstration of the performance of GPBoost model with a newton step used in the boosting step, measured in MSE. Where $\_Int$ stands for interpolation and $\_Ext$ stands for extrapolation.