Accelerated component-wise gradient boosting using efficient data representation and momentum-based optimization

D. Schalk, B. Bischl, D. Rügamer

17th, Dec. 2022

Background

About the project

- Component-wise boosting (CWB; Bühlmann and Yu, 2003; Bühlmann et al., 2007) is a boosting method based on multiple base learners.
- Utilizing interpretable statistical models as base learners makes the fitted model interpretable.
- Also, CWB is can fit models in high dimensional features spaces and provides an inherent (unbiased) feature selection.
- But, depending on the size of the data, it is infeasible to fit the algorithm w.r.t runtime or memory allocation.

$\Rightarrow$ **Goal:** Fasten the fitting process and reduce the memory consumption without loosing predictive or estimation accuracy.

CWB basics I

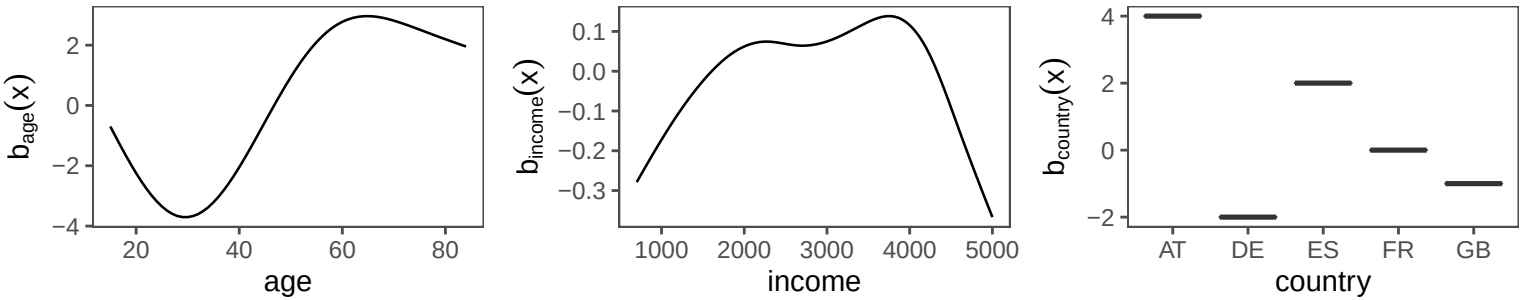
- Feature vector $\boldsymbol{x} = (x_1, \ldots, x_p) \in \mathcal{X}$
- CWB fits a model $\hat{f}$ based on an additive structure

$$\mathbb{E}[Y|\boldsymbol{x}] = \hat{f}(\boldsymbol{x}) = f_0 + \sum_{k=1}^{K} b_k(\boldsymbol{x}).$$

  Hence, CWB can be used as fitting engine for GAMs.
- K additive terms represented by base learner $b_k : \mathcal{X} \to \mathbb{R}$. Often one base learner per feature is used to model univariate effects.

CWB basics II

- Example throughout the talk:
  - Three features x_{age}, x_{country}, and x_{income}
  - Response Y is a numeric score for "happiness"
  - Aim: Model Y with b_{age} a P-spline, b_{income} a P-spline and b_{country} a one-hot encoding



CWB algorithm I

- Given is a data set $\mathcal{D}$ with n observations and p features, a loss function $L(y, \hat{y})$, and a set of base learners $b_1, \ldots, b_K$.
- The gaol of fitting CWB is to minimize the empirical risk $\mathcal{R}_{\text{emp}}(\hat{f}|\mathcal{D}) = n^{-1} \sum_{(x,y)\in\mathcal{D}} L(y, \hat{f}(x))$.
- The algorithm is initialized with a loss-optimal constant c that minimizes the empirical risk:

$$f_0 = \hat{f}^{[0]}(\boldsymbol{x}) = \arg\min_{c\in\mathcal{Y}} \mathcal{R}_{\text{emp}}(c|\mathcal{D})$$

- Model updates are calculated by functional gradient descent:

$$\hat{f}^{[m+1]} = \hat{f}^{[m]} + \nu \hat{b}^{[m]}$$

CWB algorithm II

- The negative functional gradient is expressed by pseudo residuals:

$$r^{[m](i)} = -\nabla_f L(y^{(i)}, f(\boldsymbol{x}^{(i)}))|_{f=\hat{f}^{[m-1]}}, \;\; i \in \{1, \ldots, n\}$$

- In CWB, each base learner $b_1, \ldots, b_K$ is fitted to $\boldsymbol{r}^{[m]}$ and the one with the lowest sum of squared errors (SSE) is chosen as new component $\hat{b}^{[m]}$.
- The last two step are repeated M times.

CWB algorithm III

Algorithm 1 Vanilla CWB algorithm

Input Train data $\mathcal{D}$, learning rate ν, number of boosting iterations M, loss function L, base learners $b_1, \ldots, b_K$
Output Model $\hat{f} = \hat{f}^{[M]}$, estimated coefficient vectors $\hat{\boldsymbol{\theta}}^{(i)}1, \ldots, \hat{\boldsymbol{\theta}}^{(i)}M$

procedure CWB($\mathcal{D}, \nu, L, b_1, \ldots, b_K$)
  Initialize: $f_0 = \hat{f}^{[0]}(\boldsymbol{x}) = \arg\min_{c \in \mathcal{Y}} \mathcal{R}_{\text{emp}}(c|\mathcal{D})$
  while $m \leq M$ **do**
    $r^{[m](i)} = - \left. \frac{\partial L(y^{(i)}, f(\boldsymbol{x}^{(i)}))}{\partial f(\boldsymbol{x}^{(i)})} \right|_{f = \hat{f}^{[m-1]}}, \quad \forall i \in \{1, \ldots, n\}$
    for $k \in \{1, \ldots, K\}$ **do**
      $\hat{\boldsymbol{\theta}}_k^{[m]} = \left(\boldsymbol{Z}_k^\top \boldsymbol{Z}_k + \boldsymbol{K}_k \right)^{-1} \boldsymbol{Z}_k^\top \boldsymbol{r}^{[m]}$
      $\text{SSE}_k = \sum_{i=1}^{n} (r^{[m](i)} - b_k(\boldsymbol{x}^{(i)}|\hat{\boldsymbol{\theta}}_k^{[m]}))^2$
    $k^{[m]} = \arg\min_{k \in \{1, \ldots, K\}} \text{SSE}_k$
    $\hat{f}^{[m]}(\boldsymbol{x}) = \hat{f}^{[m-1]}(\boldsymbol{x}) + \nu b_{k^{[m]}}(\boldsymbol{x}|\hat{\boldsymbol{\theta}}_{k^{[m]}}^{[m]})$
  return $\hat{f} = \hat{f}^{[M]}$

CWB advantages

- Can be fit in high dimensional feature spaces ($n \ll p$).
- Variable selection (unbiased).
- Estimation of partial feature effect that allows an interpretation of the final model.

Goal

For bigger data sets it is often infeasible to fit CWB.

$\Rightarrow$ Increase CWB's efficiency:

- **Acceleration:** Fasten the fitting by using other optimizer.
- **Memory:** Reduce the memory consumption.

Acceleration

Nesterovs momentum

Gradient descent:

Parameter space		Functional space
$\hat{\boldsymbol{\theta}}^{[m+1]} = \hat{\boldsymbol{\theta}}^{[m]} + \nu \nabla_{\boldsymbol{\theta}} \mathcal{R}_{\text{emp}}(\hat{f}(.\vert\hat{\boldsymbol{\theta}}^{[m]})\vert\mathcal{D})$	$\Rightarrow$	$\hat{f}^{[m+1]} = \hat{f}^{[m]} + \nu \hat{b}^{[m]}$

Nesterov momentum:

Parameter space		Functional space
$\boldsymbol{u}^{[m]} = \nabla_{\boldsymbol{\theta}} \mathcal{R}_{\text{emp}}(\hat{f}(.\vert\hat{\boldsymbol{\theta}}^{[m]} - \gamma\hat{\boldsymbol{\vartheta}}^{[m-1]}\vert\mathcal{D})$ $\hat{\boldsymbol{\vartheta}}^{[m]} = \gamma\hat{\boldsymbol{\vartheta}}^{[m-1]} + \nu\boldsymbol{u}^{[m]}$ $\hat{\boldsymbol{\theta}}^{[m+1]} = \hat{\boldsymbol{\theta}}^{[m]} + \hat{\boldsymbol{\vartheta}}^{[m]}$	$\Rightarrow$	???

$\Rightarrow$ **Idea:** Use Nestervos momentum and adjust it for functional updates and CWB.

Nesterovs momentum in function space

- Biau et al. (2019) proposed an adjustment of gradient boosting with Nesterovs momentum as optimizer.
- Lu et al. (2020) lined out that this approach may diverge and proposed an corrected algorithm Accelerated Gradient Boosting Machine (AGBM):

$$
\begin{aligned}
g^{[m]} &= (1 - \theta_m) f^{[m]} + \theta_m h^{[m]} \\
f^{[m+1]} &= g^{[m]} + \eta b^{[m]} \\
h^{[m+1]} &= h^{[m]} + \eta / \theta_m b_{\text{cor}}^{[m]}
\end{aligned}
$$

with $\theta_m = 2/(m+1)$ and $h^{[m]}$ the momentum sequence.

Base learners in AGBM

- $b^{[m]}$ is fit to pseudo residuals $\boldsymbol{r}^{[m]}$ w.r.t. $\hat{g}^{[m-1]}$ instead of $\hat{f}^{[m]}$
- $b_{\text{cor}}^{[m]}$ is fit to error-corrected pseudo residuals:

$$c^{[m](i)} = r^{[m](i)} + \frac{m}{m+1}(c^{[m-1](i)} - \hat{b}_{\text{cor}}^{[m-1]}(\boldsymbol{x}^{(i)})),$$

with $i = 1, \ldots, n$, if $m > 1$ and $\boldsymbol{c}^{[m]} = \boldsymbol{r}^{[m]}$ if $m = 0$.

Adapting AGBM for CWB

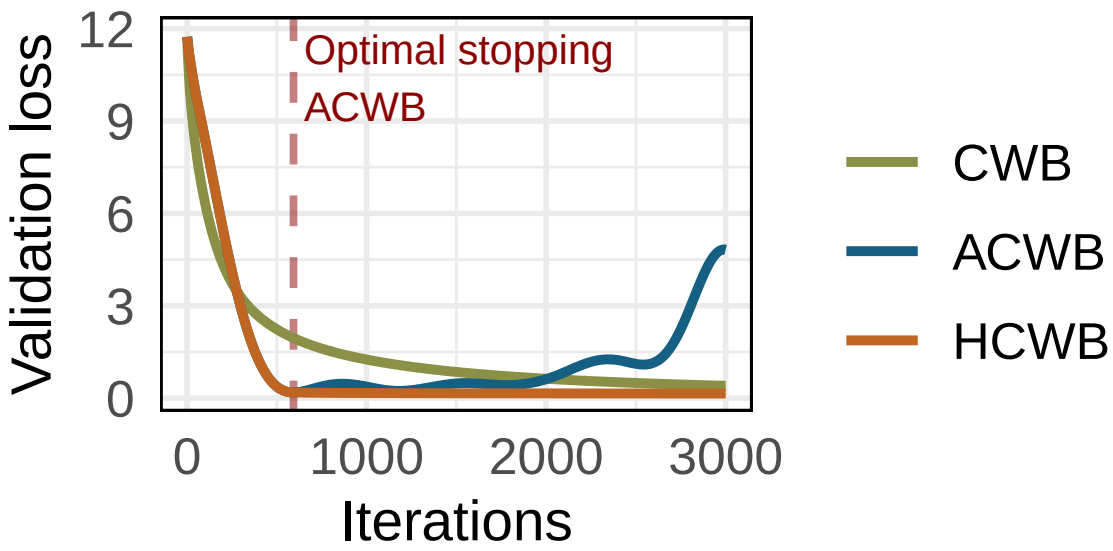
In Schalk et al. (2022), we proposed an accelerated CWB (ACWB) version by incorporating these adaptions to CWB, therefore:

- Both base learners, $b^{[m]}$ and $b_{\text{cor}}^{[m]}$, are the result of a selection process that chooses one of $b_1, \ldots, b_K$ w.r.t. to the minimal SSE on the respective pseudo residuals $\boldsymbol{r}^{[m]}$ and $\boldsymbol{c}^{[m]}$.
- Update the estimated parameters accordingly to allow the estimation of partial feature effects.

Considering these issues allows to maintain all advantages of CWB in ACWB. We refer to the publication for details about the algorithms.

Hybrid CWB

- We observed, that ACWB can overfit if not stopped early.
- Therefore, we combined ACWB with CWB to accelerate the fitting in the beginning and fine-tune the model using CWB:

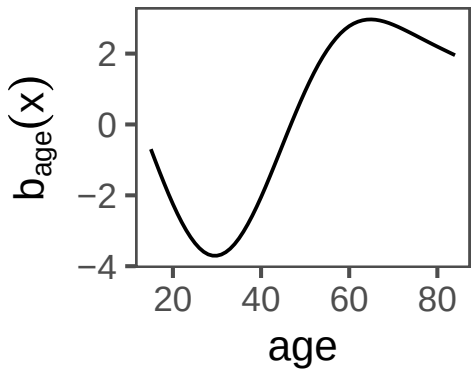


Memory

Base learner design matrix

- Each base learner $b_1, \ldots, b_K$ requires to build a design matrix $\boldsymbol{Z}_k \in \mathbb{R}^{n \times d_k}$ based on the feature vector $\boldsymbol{x}_k$.
- For example:

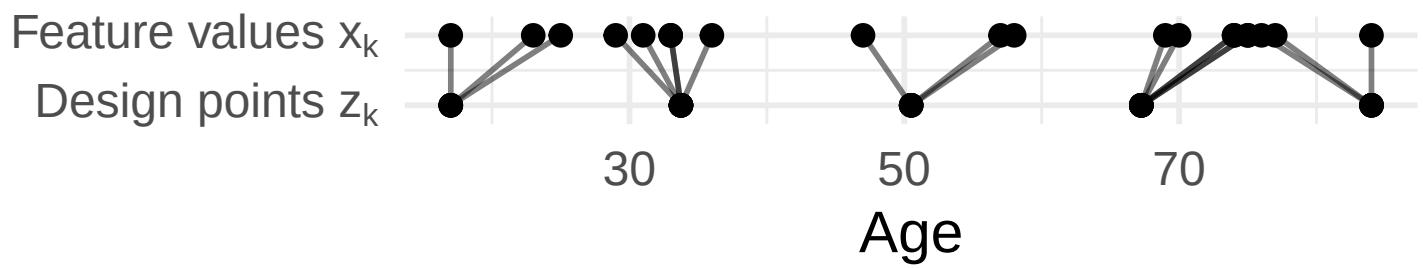
$$\boldsymbol{Z}_{\text{age}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.02 & 0.45 & 0.51 & 0.03 & 0 & 0 & 0 \\ & & & \vdots & & & \\ 0.13 & 0.66 & 0.21 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0.03 & 0.53 & 0.43 & 0.01 \end{pmatrix} \Rightarrow$$



$\Rightarrow$ If n is large, the memory gets filled very fast (especially if p is also large).

Binning

- To reduce the memory consumption, we applied binning to operate on a reduced representation of $\boldsymbol{Z}_k$.
- Binning is a technique that allows to represent the n values $x_k^{(1)}, \ldots, x_k^{(n)}$ of $\boldsymbol{x}_k$ by $n^* < n$ design points $\boldsymbol{z}_k = (z_k^{(1)}, \ldots, z_k^{(n^*)})$.
- The idea is to assign each $x_k^{(i)}$ to the closest design point $z_k^{(i)}$ and store the assignment in a map $\text{ind}_k^{(i)}$: $x_k^{(i)} \approx z_k^{(\text{ind}_k^{(i)})}$



Binning in GLMs

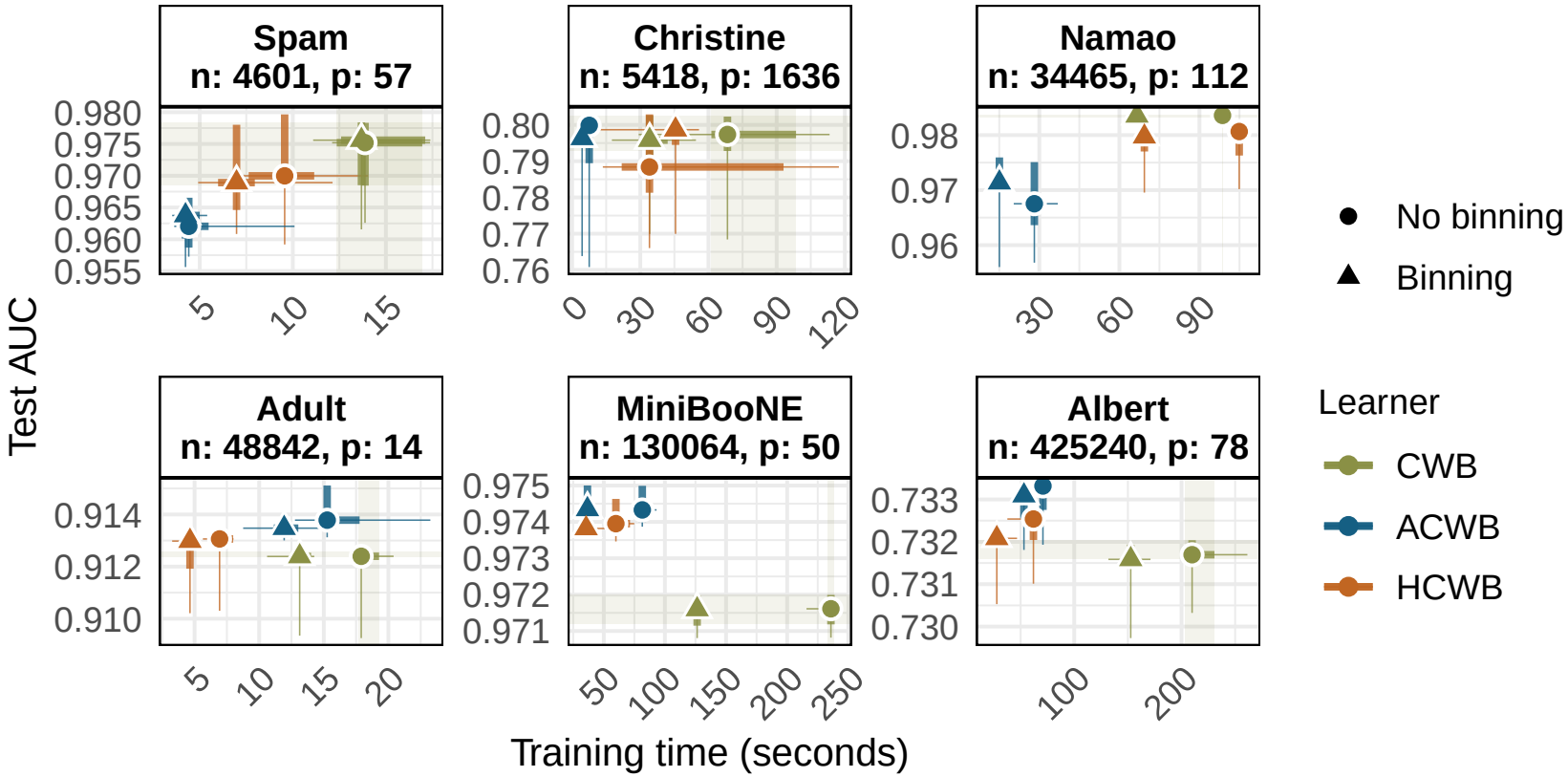
- Lang et al. (2014) used binning to discretize feature vectors to increase the efficiency of multilevel structured additive regression.
- Wood et al. (2017) applied binning to fit GAMs to gigadata and argue that the best approximation is achieved by setting $n^* = \sqrt{n}$.
- Li and Wood (2020) presented optimized cross-product operations of binned design matrices to also speed up the fitting.

Binning in CWB

- Represent numerical features $\boldsymbol{x}_k$ by n^* design points $\boldsymbol{z}_k$.
- Build the design matrix $\boldsymbol{Z}_k$ based on $\boldsymbol{z}_k$ which requires to store $n^* d_k$ values instead of $n d_k$.
- Use optimized cross-product operations to estimate the parameters $\hat{\boldsymbol{\theta}}_k^{[m]} = (\boldsymbol{Z}_k^\top \boldsymbol{Z}_k + \boldsymbol{K}_k)^{-1} \boldsymbol{Z}_k^\top \boldsymbol{r}^{[m]}$ of a base learner to also speed up the fitting.

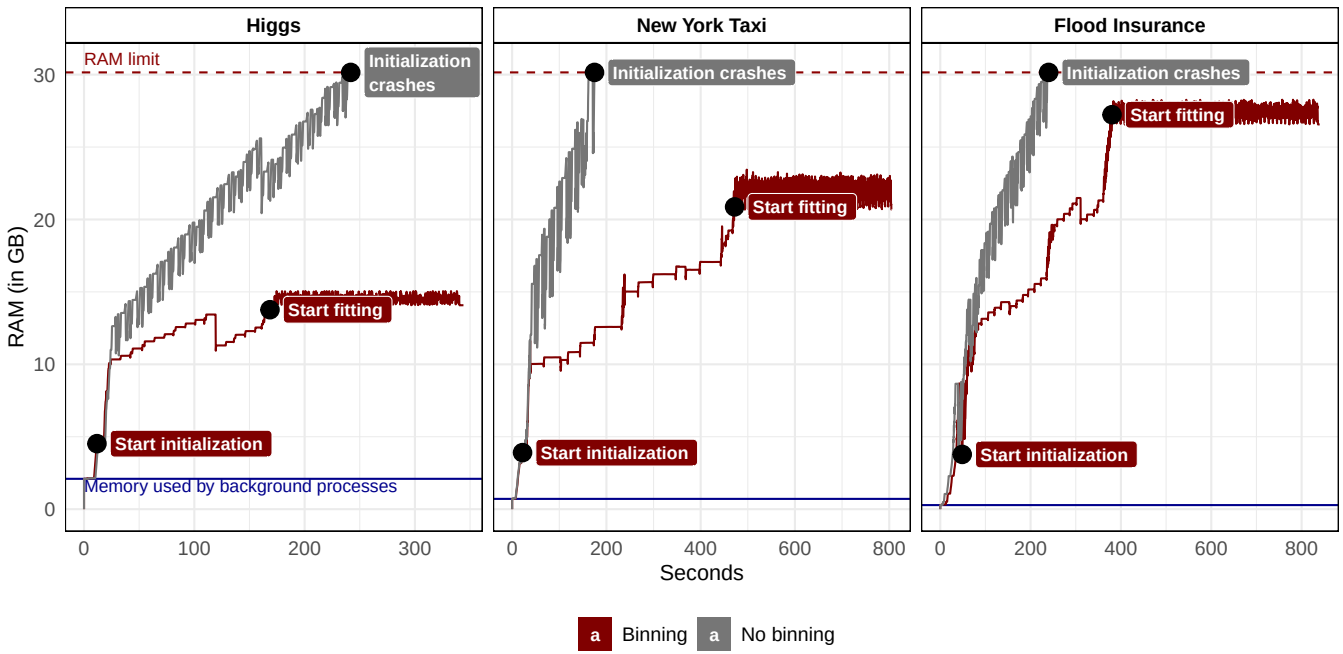
Application to data

Benchmark


Spam
n: 4601, p: 57
Christine
n: 5418, p: 1636
Namao
n: 34465, p: 112
Adult
n: 48842, p: 14
MiniBooNE
n: 130064, p: 50
Albert
n: 425240, p: 78
Test AUC
Training time (seconds)
No binning
Binning
Learner
CWB
ACWB
HCWB

Binning with big data

- HIGGS: 11×10^6 observations, 29 features, 2.4 GB
- NYC Yellow Taxi Trip: 24.3×10^6 observations, 22 features, 3.3 GB
- FEMA's National Flood Insurance Policy Database: 14.5×10^6 observations, 50 features, 3.4 GB



References

References

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

Bühlmann, P., Hothorn, T., et al. (2007). Boosting algorithms: Regularization, prediction and model fitting. *Statistical science*, 22(4):477–505.

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

Lang, S., Umlauf, N., Wechselberger, P., Harttgen, K., and Kneib, T. (2014). Multilevel structured additive regression. *Statistics and Computing*, 24(2):223–238.

Li, Z. and Wood, S. N. (2020). Faster model matrix crossproducts for large generalized linear models with discretized covariates. *Statistics and Computing*, 30(1):19–25.

Lu, H., Karimireddy, S. P., Ponomareva, N., and Mirrokni, V. (2020). Accelerating gradient boosting machines. In *International Conference on Artificial Intelligence and Statistics*, pages 516–526. PMLR.

Schalk, D., Bischl, B., and Rügamer, D. (2022). Accelerated componentwise gradient boosting using efficient data representation and momentum-based optimization. *Journal of Computational and Graphical Statistics*, 0(ja):1–27.

Wood, S. N., Li, Z., Shaddick, G., and Augustin, N. H. (2017). Generalized additive models for gigadata: Modeling the u.k. black smoke network daily data. *Journal of the American Statistical Association*, 112(519):1199–1210.