Modern approaches for component-wise boosting:

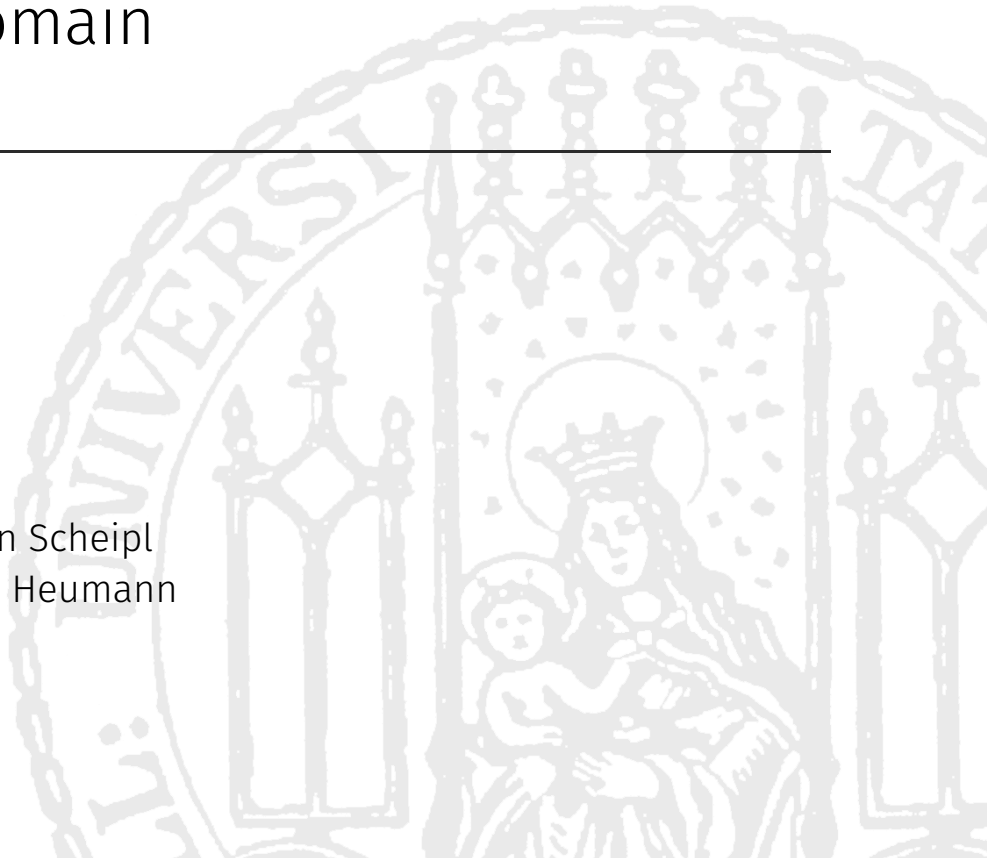
Automation, efficiency, and distributed computing with application to the medical domain

---

Daniel Schalk

March 24, 2023

Supervisor: Prof. Dr. Bernd Bischl
Reviewers: Prof. Dr. Matthias Schmid, PD Dr. Fabian Scheipl
Chair of the examination panel: Prof. Dr. Christian Heumann



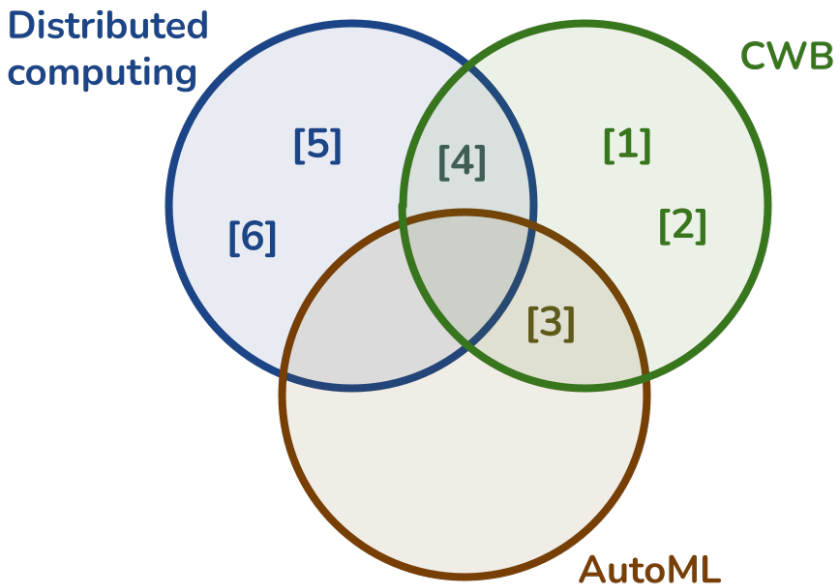
About the dissertation

Overview

Focus of the dissertation:

Discuss and propose modern directions for component-wise gradient boosting (CWB; Bühlmann and Yu, 2003).

Contribution and topics:

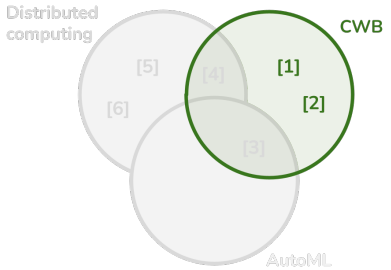


Publications

Part I - Efficiency (CWB)

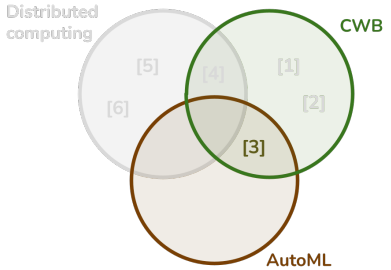
[1] Schalk, D., Thomas, J., and Bischl, B. (2018). compboost: Modular framework for component-wise boosting. *Journal of Open Source Software*, 3(30):967

[2] Schalk, D., Bischl, B., and Rügamer, D. (2022a). Accelerated componentwise gradient boosting using efficient data representation and momentum-based optimization. *Journal of Computational and Graphical Statistics*



Part II - Automation

[3] Coors, S., Schalk, D., Bischl, B., and Rügamer, D. (2021). Automatic componentwise boosting: An interpretable automl system. *ECML-PKDD Workshop on Automating Data Science*

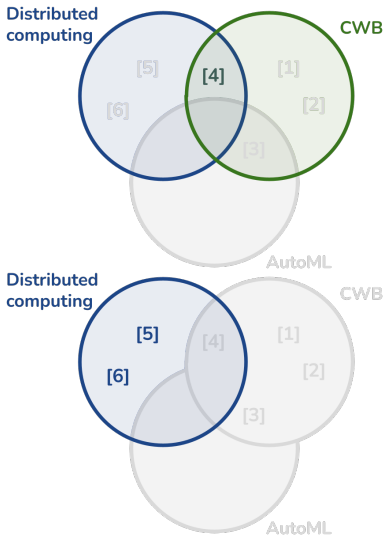


Part III - Distributed computing

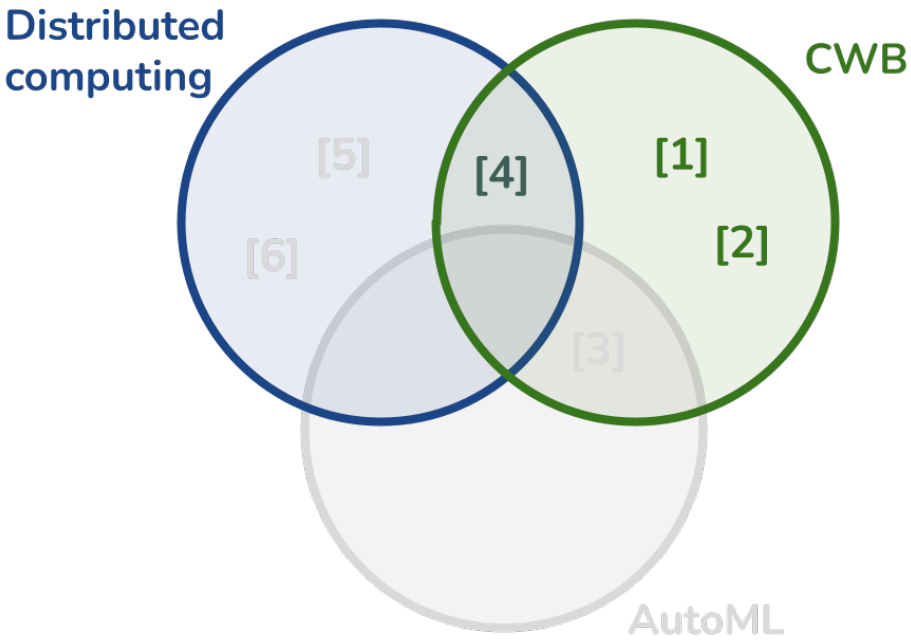
[4] Schalk, D., Bischl, B., and Rügamer, D. (2023a). Privacy-preserving and lossless distributed estimation of high-dimensional generalized additive mixed models. *arXiv preprint arXiv:2210.07723*. [Currently under review in the journal *Statistics and Computing*]

[5] Schalk, D., Hoffmann, V. S., Bischl, B., and Mansmann, U. (2022b). Distributed non-disclosive validation of predictive models by a modified roc-glm. *arXiv preprint arXiv:2203.10828* [Currently under review in the journal *BMC Medical Research Methodology*]

[6] Schalk, D., Hoffmann, V. S., Bischl, B., and Mansmann, U. (2023b). dsBinVal: Conducting distributed roc analysis using datashield. *Journal of Open Source Software*, 8(82):4545



Overview



The focus of this presentation is on CWB adjustments [1], [2], and [4]. The other contributions are briefly summarized at the end of the presentation.

Structure of the talk

Background

Efficiency

Distributed computing

Further contributions

Conclusion and outlook

Background

Data

Example throughout this presentation is a subset of a WHO data set[1] about life expectation in years per country:

Life.expectancy in years	**Country**	**Year**	...	**BMI** in feet/inches	**Adult.Mortality** per 1000 citizens
53.2	ETH	2002	...	12.9	369
81.0	GER	2015	...	62.3	68
76.8	USA	2000	...	6.1	114
⋮	⋮	⋮		⋮	⋮
79.1	USA	2014	...	69.1	14
58.9	ZAF	2011	...	47.9	413
69.0	ZAF	2013	...	49.5	371

- $n = 75$ (for `Country` $\in \{$GER, USA, SWE, ZAF, ETH$\}$, $n = 2938$ for all countries)
- $p = 20$

[1] Available at `kaggle.com/datasets/kumarajarshi/life-expectancy-who`

Background

Component-wise gradient boosting

GB algorithm

Algorithm 1 Gradient boosting (GB) algorithm

Input Train data $\mathcal{D}$, number of boosting iterations M, loss function L, base learner b
Output Model $\hat{f} = \hat{f}^{[M]}$

procedure GB($\mathcal{D}, M, L, b$)
  Initialize: $f_0 = \hat{f}^{[0]}(\mathbf{x}) = \textbf{arg min}_{c \in \mathcal{Y}} \mathcal{R}_{\text{emp}}(c|\mathcal{D})$
  while $m \leq M$ **do**
    $r^{[m](i)} = - \left. \frac{\partial L(y^{(i)}, f(\mathbf{x}^{(i)}))}{\partial f(\mathbf{x}^{(i)})} \right|_{f = \hat{f}^{[m-1]}}, \quad \forall i \in \{1, \ldots, n\}$
    $\hat{\boldsymbol{\theta}}^{[m]} = \textbf{arg min}_{\boldsymbol{\theta}} \sum_{i=1}^{n} (r^{[m](i)} - b(\mathbf{x}^{(i)}|\boldsymbol{\theta}))^2$
    $\nu_m = \textbf{arg min}_{\nu \in \mathbb{R}} \sum_{i=1}^{n} L(r^{[m](i)}, \hat{f}^{[m]} + \nu \hat{b}^{[m]}(\mathbf{x}^{(i)}|\hat{\boldsymbol{\theta}}^{[m]}))$
    $\hat{f}^{[m]}(\mathbf{x}) = \hat{f}^{[m-1]}(\mathbf{x}) + \nu_m \hat{b}^{[m]}(\mathbf{x}|\hat{\boldsymbol{\theta}}^{[m]})$
  return $\hat{f} = \hat{f}^{[M]}$

GB algorithm

Algorithm 1 Gradient boosting (GB) algorithm

Input Train data $\mathcal{D}$, number of boosting iterations M, loss function L, base learner b
Output Model $\hat{f} = \hat{f}^{[M]}$

procedure GB($\mathcal{D}, M, L, b$)
  Initialize: $f_0 = \hat{f}^{[0]}(\mathbf{x}) = \arg\min_{c \in \mathcal{Y}} \mathcal{R}_{\text{emp}}(c|\mathcal{D})$
  while $m \leq M$ **do**
    $r^{[m](i)} = - \left. \frac{\partial L(y^{(i)}, f(\mathbf{x}^{(i)}))}{\partial f(\mathbf{x}^{(i)})} \right|_{f = \hat{f}^{[m-1]}}, \quad \forall i \in \{1, \ldots, n\}$
    $\hat{\boldsymbol{\theta}}^{[m]} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} (r^{[m](i)} - b(\mathbf{x}^{(i)}|\boldsymbol{\theta}))^2$
    $\nu_m = \arg\min_{\nu \in \mathbb{R}} \sum_{i=1}^{n} L(r^{[m](i)}, \hat{f}^{[m]} + \nu \hat{b}^{[m]}(\mathbf{x}^{(i)}|\hat{\boldsymbol{\theta}}^{[m]}))$
    $\hat{f}^{[m]}(\mathbf{x}) = \hat{f}^{[m-1]}(\mathbf{x}) + \nu_m \hat{b}^{[m]}(\mathbf{x}|\hat{\boldsymbol{\theta}}^{[m]})$
  return $\hat{f} = \hat{f}^{[M]}$

The pseudo residuals $\boldsymbol{r}^{[m]}$ contain the information in which direction to move $\hat{f}^{[m-1]}$ to better fit the data.

GB algorithm

Algorithm 1 Gradient boosting (GB) algorithm

Input Train data $\mathcal{D}$, number of boosting iterations M, loss function L, base learner b
Output Model $\hat{f} = \hat{f}^{[M]}$

1: **procedure** GB($\mathcal{D}, M, L, b$)
2: Initialize: $f_0 = \hat{f}^{[0]}(\mathbf{x}) = \arg\min_{c \in \mathcal{Y}} \mathcal{R}_{\text{emp}}(c|\mathcal{D})$
3: **while** $m \leq M$ **do**
4: $r^{[m](i)} = - \left. \frac{\partial L(y^{(i)}, f(\mathbf{x}^{(i)}))}{\partial f(\mathbf{x}^{(i)})} \right|_{f = \hat{f}^{[m-1]}}, \quad \forall i \in \{1, \ldots, n\}$
5: $\hat{\boldsymbol{\theta}}^{[m]} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} (r^{[m](i)} - b(\mathbf{x}^{(i)}|\boldsymbol{\theta}))^2$
6: $\nu_m = \arg\min_{\nu \in \mathbb{R}} \sum_{i=1}^{n} L(r^{[m](i)}, \hat{f}^{[m]} + \nu \hat{b}^{[m]}(\mathbf{x}^{(i)}|\hat{\boldsymbol{\theta}}^{[m]}))$
7: $\hat{f}^{[m]}(\mathbf{x}) = \hat{f}^{[m-1]}(\mathbf{x}) + \nu_m \hat{b}^{[m]}(\mathbf{x}|\hat{\boldsymbol{\theta}}^{[m]})$
8: **return** $\hat{f} = \hat{f}^{[M]}$

One base learner b (often a tree) is fitted to $\boldsymbol{r}^{[m]}$.

GB algorithm

Algorithm 1 Gradient boosting (GB) algorithm

Input Train data $\mathcal{D}$, number of boosting iterations M, loss function L, base learner b
Output Model $\hat{f} = \hat{f}^{[M]}$

procedure GB($\mathcal{D}, M, L, b$)
  Initialize: $f_0 = \hat{f}^{[0]}(\mathbf{x}) = \arg\min_{c \in \mathcal{Y}} \mathcal{R}_{\text{emp}}(c|\mathcal{D})$
  while $m \leq M$ **do**
    $r^{[m](i)} = - \left. \frac{\partial L(y^{(i)}, f(\mathbf{x}^{(i)}))}{\partial f(\mathbf{x}^{(i)})} \right|_{f = \hat{f}^{[m-1]}}, \quad \forall i \in \{1, \ldots, n\}$
    $\hat{\boldsymbol{\theta}}^{[m]} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} (r^{[m](i)} - b(\mathbf{x}^{(i)}|\boldsymbol{\theta}))^2$
    $\nu_m = \arg\min_{\nu \in \mathbb{R}} \sum_{i=1}^{n} L(r^{[m](i)}, \hat{f}^{[m]} + \nu \hat{b}^{[m]}(\mathbf{x}^{(i)}|\hat{\boldsymbol{\theta}}^{[m]}))$
    $\hat{f}^{[m]}(\mathbf{x}) = \hat{f}^{[m-1]}(\mathbf{x}) + \nu_m \hat{b}^{[m]}(\mathbf{x}|\hat{\boldsymbol{\theta}}^{[m]})$
  return $\hat{f} = \hat{f}^{[M]}$

The fitted base learner is added to the ensemble by conducting a functional gradient descent step.

CWB basics

- Compared to GB, CWB can choose from a set of K base learners $b \in \{b_1, \ldots, b_K\}$.
- Often, $b_1, \ldots, b_K$ are chosen to be (interpretable) statistical models and hence f corresponds to a generalized additive model (GAM; Hastie, 2017):

$$f(\mathbf{x}) = f_0 + \sum_{k=1}^{K} b_k(\mathbf{x}|\boldsymbol{\theta}_k), \quad \text{intercept } f_0$$

- Advantages of CWB:
  - Feasible to get fit in high-dimensional feature spaces ($p \gg n$).
  - An inherent (unbiased) feature selection.
  - Interpretable/explainable partial feature effects (depending on the choice of base learners).

GB algorithm

Algorithm 1 GB algorithm

Input Train data $\mathcal{D}$, number of boosting iterations M, loss function L, base learner b
Output Model $\hat{f} = \hat{f}^{[M]}$

1: **procedure** GB($\mathcal{D}, M, L, b$)
2: Initialize: $f_0 = \hat{f}^{[0]}(\mathbf{x}) = \arg\min_{c \in \mathcal{Y}} \mathcal{R}_{\text{emp}}(c|\mathcal{D})$
3: **while** $m \leq M$ **do**
4: $r^{[m](i)} = - \left. \frac{\partial L(y^{(i)}, f(\mathbf{x}^{(i)}))}{\partial f(\mathbf{x}^{(i)})} \right|_{f = \hat{f}^{[m-1]}}, \quad \forall i \in \{1, \ldots, n\}$
5: $\hat{\boldsymbol{\theta}}^{[m]} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} (r^{[m](i)} - b(\mathbf{x}^{(i)}|\boldsymbol{\theta}))^2$
6: $\nu_m = \arg\min_{\nu \in \mathbb{R}} \sum_{i=1}^{n} L(r^{[m](i)}, \hat{f}^{[m]} + \nu \hat{b}^{[m]}(\mathbf{x}^{(i)}|\hat{\boldsymbol{\theta}}^{[m]}))$
7: $\hat{f}^{[m]}(\mathbf{x}) = \hat{f}^{[m-1]}(\mathbf{x}) + \nu_m \hat{b}^{[m]}(\mathbf{x}|\hat{\boldsymbol{\theta}}^{[m]})$
8: **return** $\hat{f} = \hat{f}^{[M]}$

CWB algorithm

Algorithm 2 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 $\hat{f} = \hat{f}^{[M]}$

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

Algorithm 2 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 $\hat{f} = \hat{f}^{[M]}$

procedure CWB($\mathcal{D}, \nu, M, L, b_1, \ldots, b_K$)

  Initialize: $f_0 = \hat{f}^{[0]}(\mathbf{x}) = \textbf{arg min}_{c \in \mathcal{Y}} \mathcal{R}_{\text{emp}}(c|\mathcal{D})$

  while $m \leq M$ **do**

    $r^{[m](i)} = - \left. \frac{\partial L(y^{(i)}, f(\mathbf{x}^{(i)}))}{\partial f(\mathbf{x}^{(i)})} \right|_{f = \hat{f}^{[m-1]}}, \quad \forall i \in \{1, \ldots, n\}$

    for $k \in \{1, \ldots, K\}$ **do**

      $\hat{\boldsymbol{\theta}}_k^{[m]} = \left(Z_k^\top Z_k + K_k\right)^{-1} Z_k^\top r^{[m]}$

      $\text{SSE}_k = \sum_{i=1}^{n} (r^{[m](i)} - b_k(\mathbf{x}^{(i)}|\hat{\boldsymbol{\theta}}_k^{[m]}))^2$

    $k^{[m]} = \textbf{arg min}_{k \in \{1,\ldots,K\}} \text{SSE}_k$

    $\hat{f}^{[m]}(\mathbf{x}) = \hat{f}^{[m-1]}(\mathbf{x}) + \nu b_{k^{[m]}}(\mathbf{x}|\hat{\boldsymbol{\theta}}_i^{[m]} k^{[m]})$

  return $\hat{f} = \hat{f}^{[M]}$

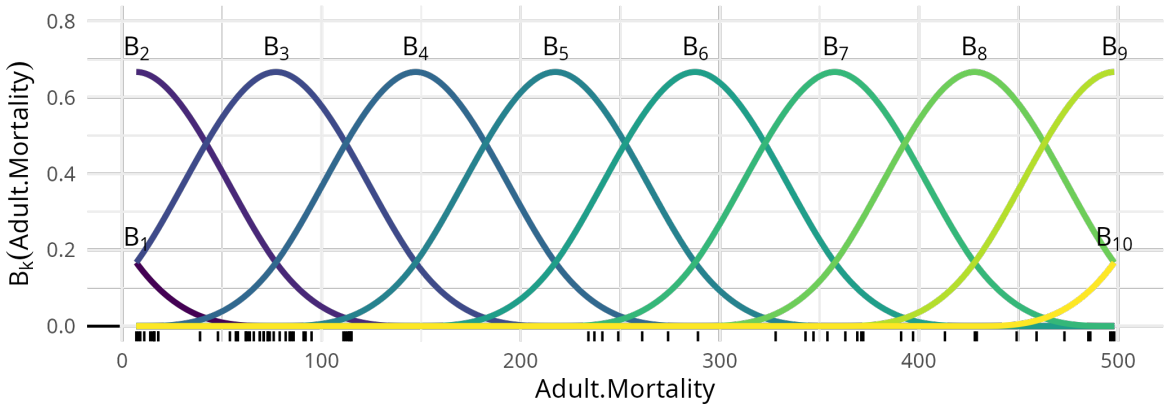
Base learner

- Each base learner b_k has a basis transformation $g_k : \mathcal{X} \to \mathbb{R}^{d_k}$ with
$$g_k(\mathbf{x}) = (g_{k,1}(\mathbf{x}), \ldots, g_{k,d_k}(\mathbf{x}))^\top.$$
and is linear in the parameters: $b_k(\mathbf{x}|\boldsymbol{\theta}_k) = g_k(\mathbf{x})^\top \boldsymbol{\theta}_k$
- For n data points $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$, each base learner defines a design matrix:
$$\boldsymbol{Z}_k = \begin{pmatrix} g_k^\top(\mathbf{x}^{(1)}) \\ \vdots \\ g_k^\top(\mathbf{x}^{(n)}) \end{pmatrix} \in \mathbb{R}^{n \times d_k}$$
- Each base learner can have a penalty matrix $\boldsymbol{K}_k$ (e.g. $\boldsymbol{K}_k = \lambda_k \boldsymbol{I}_{d_k}$ for ridge regression) and is fitted using the least squares estimator
$$\hat{\boldsymbol{\theta}}_k = (\boldsymbol{Z}_k^\top \boldsymbol{Z}_k + \boldsymbol{K}_k)^{-1} \boldsymbol{Z}_k^\top \mathbf{y}.$$

B/P-Spline base learner

$$g_k(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$$

B-spline basis B of a pre-defined degree (Eilers and Marx, 1996).

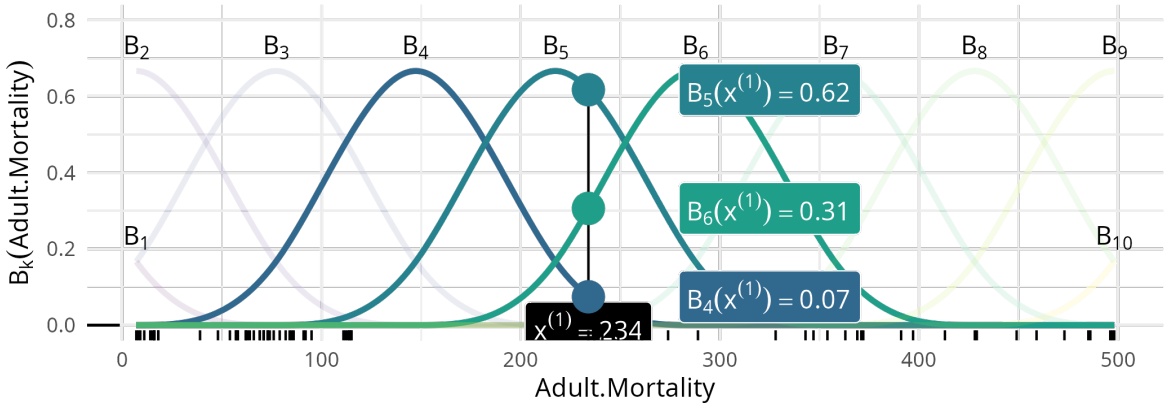


$$Z_k = \begin{pmatrix} g_k^\top(\mathbf{x}^{(1)}) \\ \vdots \\ g_k^\top(\mathbf{x}^{(n)}) \end{pmatrix} = \begin{pmatrix} B_{k,1}(\mathbf{x}^{(1)}) & \ldots & B_{k,d_k}(\mathbf{x}^{(1)}) \\ \vdots & & \vdots \\ B_{k,1}(\mathbf{x}^{(n)}) & \ldots & B_{k,d_k}(\mathbf{x}^{(n)}) \end{pmatrix} \in \mathbb{R}^{n \times d_k}$$

B/P-spline base learner

$$g_k(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$$

B-spline basis B of a pre-defined degree (Eilers and Marx, 1996).



$$Z_k = \begin{array}{cccccccccc} B_1 & B_2 & B_3 & B_4 & B_5 & B_6 & B_7 & B_8 & B_9 & B_{10} \\ \left(\right. \end{array}$$

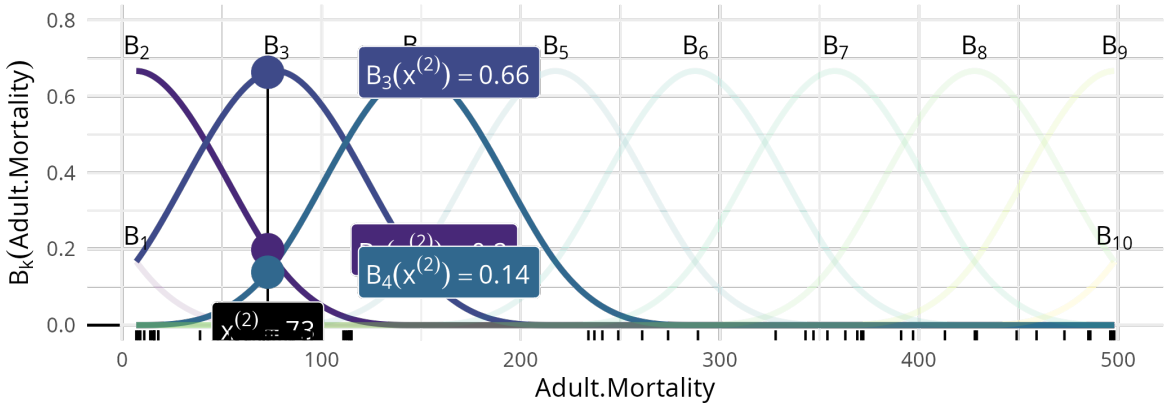
B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}
0.00	0.00	0.00	0.07	0.62	0.31	0.00	0.00	0.00	0.00

$\left. \right)$

B/P-spline base learner

$$g_k(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$$

B-spline basis B of a pre-defined degree (Eilers and Marx, 1996).

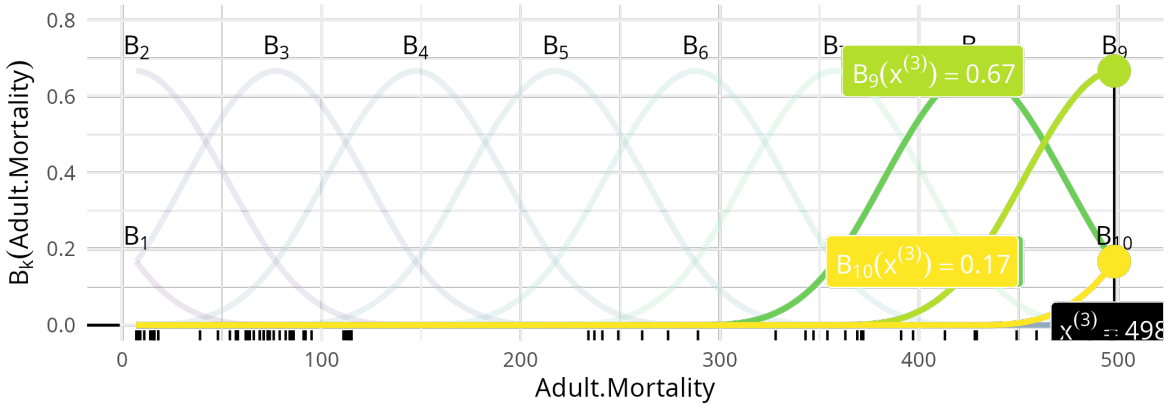


$$Z_k = \begin{array}{c} \begin{array}{cccccccccc} B_1 & B_2 & B_3 & B_4 & B_5 & B_6 & B_7 & B_8 & B_9 & B_{10} \end{array} \\ \begin{pmatrix} 0.00 & 0.00 & 0.00 & 0.07 & 0.62 & 0.31 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.20 & 0.66 & 0.14 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ & & & & & & & & & \\ & & & & & & & & & \\ & & & & & & & & & \end{pmatrix} \end{array}$$

B/P-spline base learner

$$g_k(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$$

B-spline basis B of a pre-defined degree (Eilers and Marx, 1996).

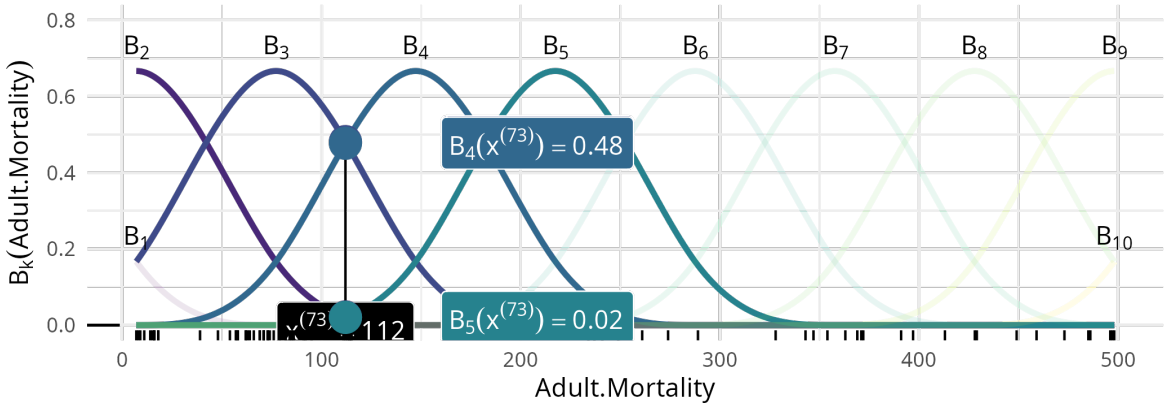


$$Z_k = \begin{array}{c} \begin{array}{cccccccccc} B_1 & B_2 & B_3 & B_4 & B_5 & B_6 & B_7 & B_8 & B_9 & B_{10} \end{array} \\ \begin{pmatrix} 0.00 & 0.00 & 0.00 & 0.07 & 0.62 & 0.31 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.20 & 0.66 & 0.14 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.17 & 0.67 & 0.17 \end{pmatrix} \end{array}$$

B/P-spline base learner

$$g_k(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$$

B-spline basis B of a pre-defined degree (Eilers and Marx, 1996).

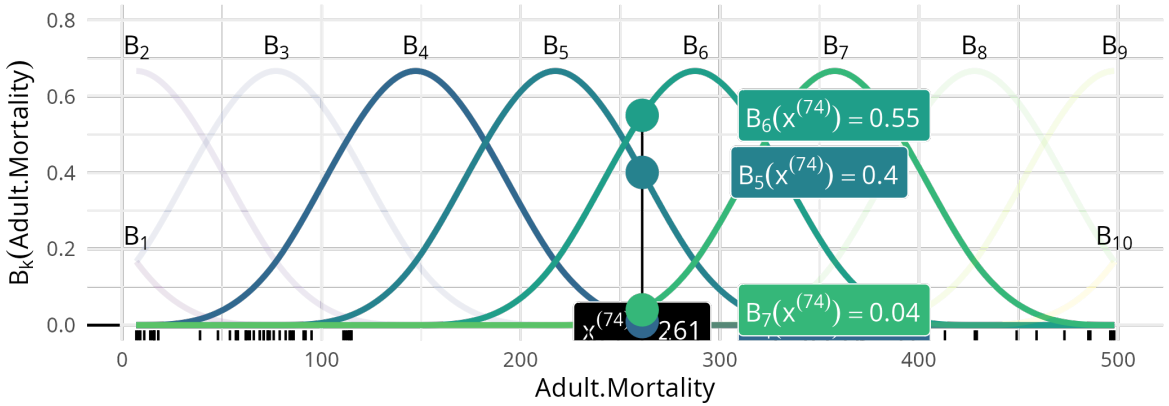


$$Z_k = \begin{array}{c} \begin{matrix} B_1 & B_2 & B_3 & B_4 & B_5 & B_6 & B_7 & B_8 & B_9 & B_{10} \end{matrix} \\ \begin{pmatrix} 0.00 & 0.00 & 0.00 & 0.07 & 0.62 & 0.31 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.20 & 0.66 & 0.14 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.17 & 0.67 & 0.17 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.00 & 0.02 & 0.48 & 0.48 & 0.02 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \end{pmatrix} \end{array}$$

B/P-spline base learner

$$g_k(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$$

B-spline basis B of a pre-defined degree (Eilers and Marx, 1996).

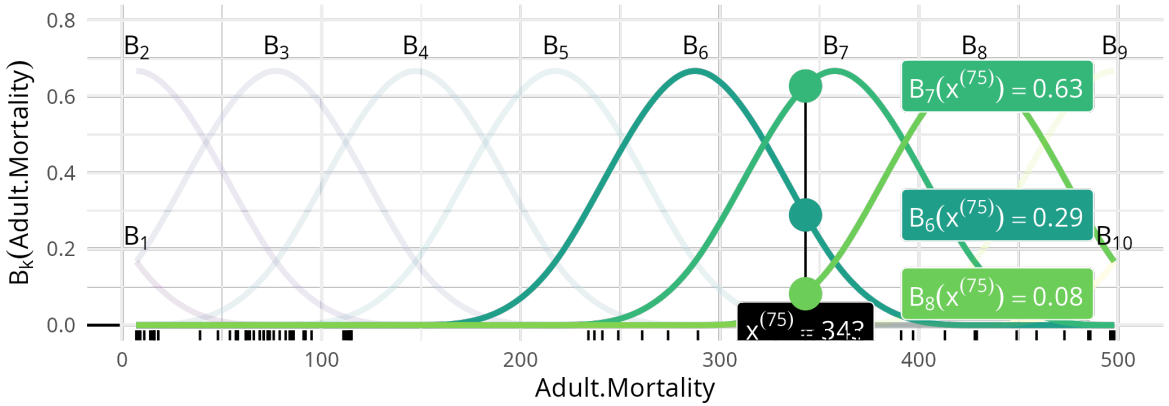


$$Z_k = \begin{array}{c} \begin{array}{cccccccccc} B_1 & B_2 & B_3 & B_4 & B_5 & B_6 & B_7 & B_8 & B_9 & B_{10} \end{array} \\ \begin{pmatrix} 0.00 & 0.00 & 0.00 & 0.07 & 0.62 & 0.31 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.20 & 0.66 & 0.14 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.17 & 0.67 & 0.17 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.00 & 0.02 & 0.48 & 0.48 & 0.02 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.01 & 0.40 & 0.55 & 0.04 & 0.00 & 0.00 & 0.00 \end{pmatrix} \end{array}$$

B/P-spline base learner

$$g_k(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$$

B-spline basis B of a pre-defined degree (Eilers and Marx, 1996).



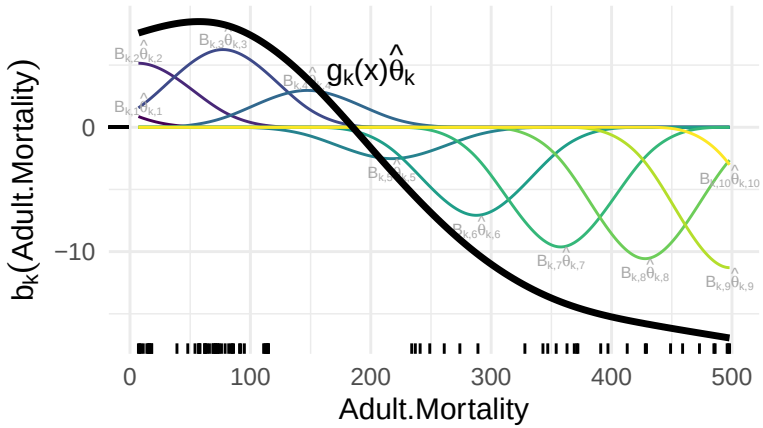
$$Z_k = \begin{pmatrix} B_1 & B_2 & B_3 & B_4 & B_5 & B_6 & B_7 & B_8 & B_9 & B_{10} \\ 0.00 & 0.00 & 0.00 & 0.07 & 0.62 & 0.31 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.20 & 0.66 & 0.14 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.17 & 0.67 & 0.17 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.00 & 0.02 & 0.48 & 0.48 & 0.02 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.01 & 0.40 & 0.55 & 0.04 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.29 & 0.63 & 0.08 & 0.00 & 0.00 \end{pmatrix}$$

B/P-spline base learner

$$g_k(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$$

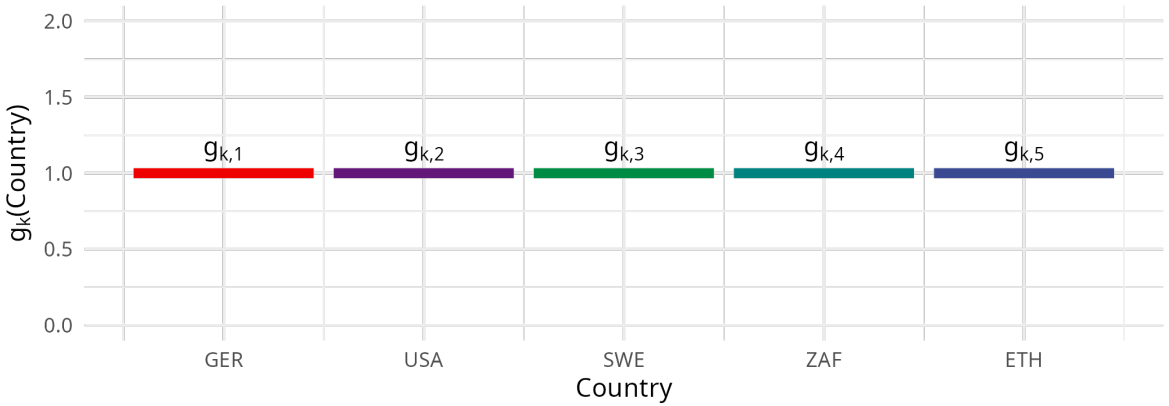
B-spline basis B of a pre-defined degree (Eilers and Marx, 1996).

$$Z_k \quad \Rightarrow \quad \hat{\boldsymbol{\theta}}_k = (Z_k^\top Z_k + K_k)^{-1} Z_k^\top y \quad \Rightarrow$$



Categorical base learner

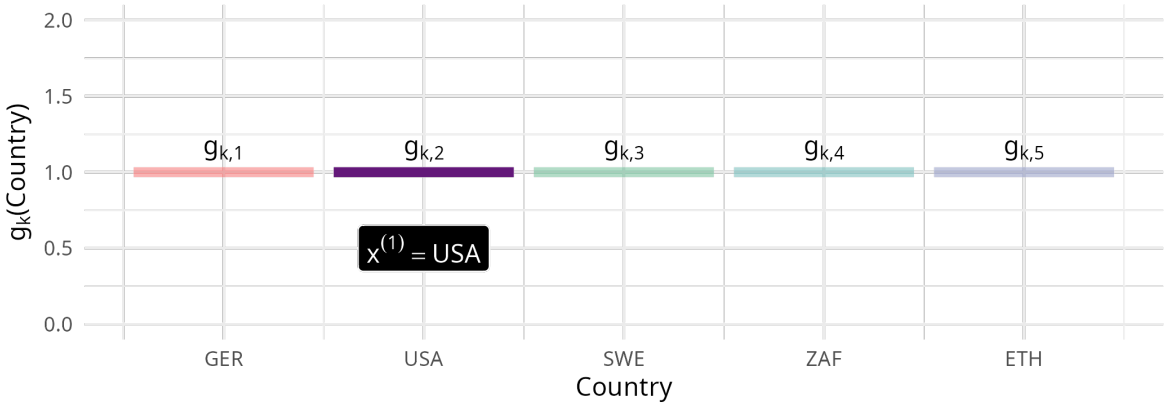
$$g_k(x) = (g_{k,1}(x), \ldots, g_{k,G}(x))^\top = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top, \quad x \in \{1, \ldots, G\}$$



$$Z_k = \begin{pmatrix} g_k^\top(\mathbf{x}^{(1)}) \\ \vdots \\ g_k^\top(\mathbf{x}^{(n)}) \end{pmatrix} = \begin{pmatrix} \mathbb{1}_{\{\mathbf{x}^{(1)}=1\}} & \cdots & \mathbb{1}_{\{\mathbf{x}^{(1)}=G\}} \\ \vdots & & \vdots \\ \mathbb{1}_{\{\mathbf{x}^{(n)}=1\}} & \cdots & \mathbb{1}_{\{\mathbf{x}^{(n)}=G\}} \end{pmatrix} \in \mathbb{R}^{n \times G}$$

Categorical base learner

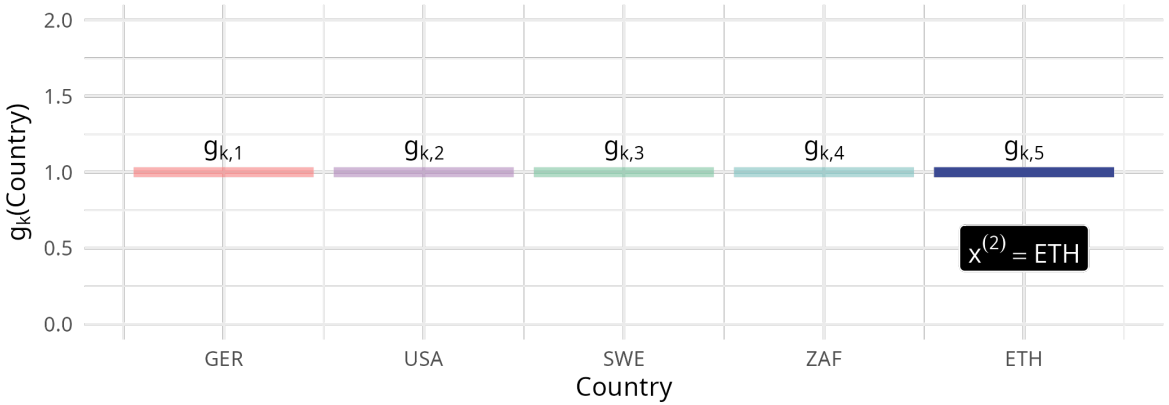
$$g_k(x) = (g_{k,1}(x), \ldots, g_{k,G}(x))^\top = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top, \quad x \in \{1, \ldots, G\}$$



$$Z_k = \begin{array}{c} \begin{array}{ccccc} g_{k,GER} & g_{k,USA} & g_{k,SWE} & g_{k,ZAF} & g_{k,ETH} \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ & & & & \\ & & & & \end{pmatrix} \end{array}$$

Categorical base learner

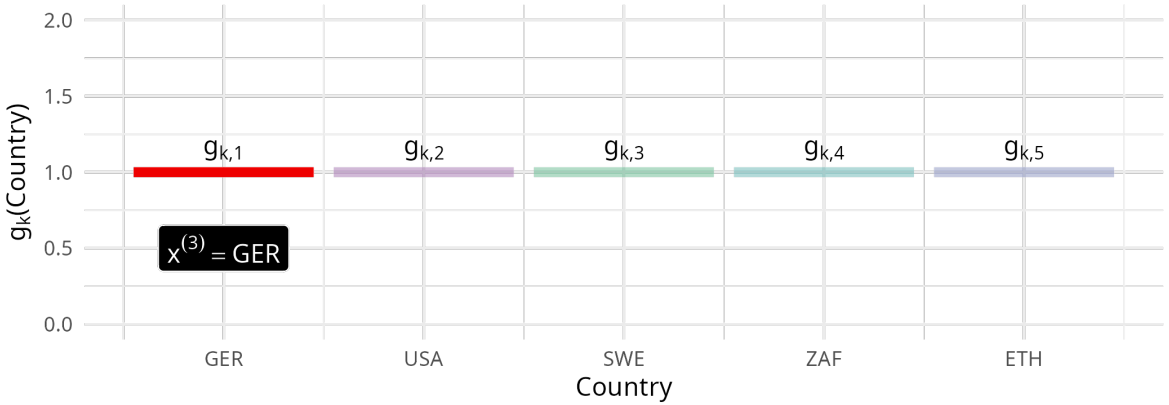
$$g_k(x) = (g_{k,1}(x), \ldots, g_{k,G}(x))^\top = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top, \quad x \in \{1, \ldots, G\}$$



$$Z_k = \begin{array}{c} \begin{array}{ccccc} g_{k,GER} & g_{k,USA} & g_{k,SWE} & g_{k,ZAF} & g_{k,ETH} \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ & & & & \\ & & & & \\ & & & & \end{pmatrix} \end{array}$$

Categorical base learner

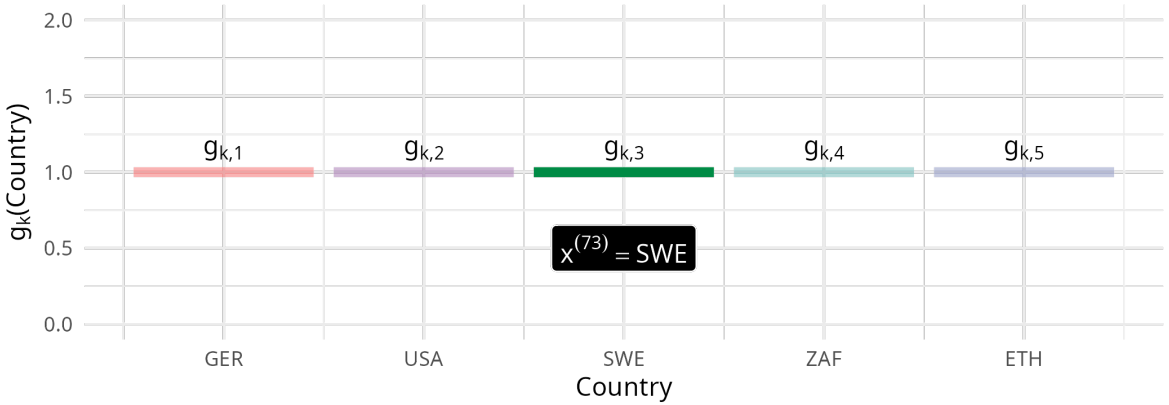
$$g_k(x) = (g_{k,1}(x), \ldots, g_{k,G}(x))^\top = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top, \quad x \in \{1, \ldots, G\}$$



$$Z_k = \begin{array}{c} \begin{array}{ccccc} g_{k,GER} & g_{k,USA} & g_{k,SWE} & g_{k,ZAF} & g_{k,ETH} \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ & & & & \\ & & & & \end{pmatrix} \end{array}$$

Categorical base learner

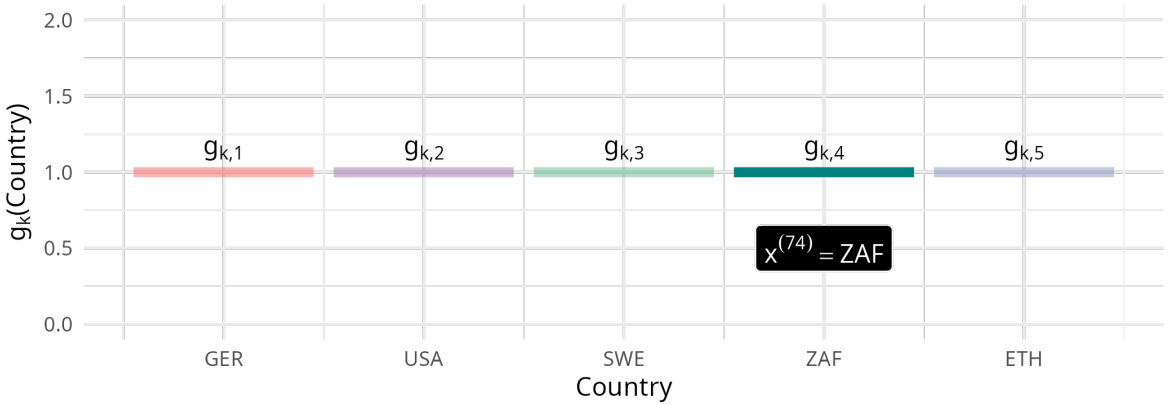
$$g_k(x) = (g_{k,1}(x), \ldots, g_{k,G}(x))^\top = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top, \quad x \in \{1, \ldots, G\}$$



$$Z_k = \begin{pmatrix} g_{k,GER} & g_{k,USA} & g_{k,SWE} & g_{k,ZAF} & g_{k,ETH} \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Categorical base learner

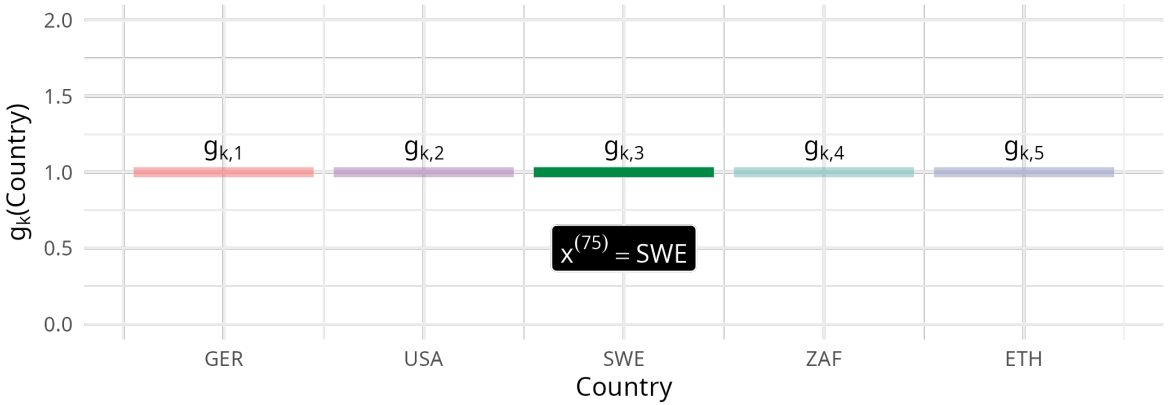
$$g_k(x) = (g_{k,1}(x), \ldots, g_{k,G}(x))^\top = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top, \quad x \in \{1, \ldots, G\}$$



$$Z_k = \begin{array}{c} \begin{array}{ccccc} g_{k,GER} & g_{k,USA} & g_{k,SWE} & g_{k,ZAF} & g_{k,ETH} \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

Categorical base learner

$$g_k(x) = (g_{k,1}(x), \ldots, g_{k,G}(x))^\top = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top, \quad x \in \{1, \ldots, G\}$$

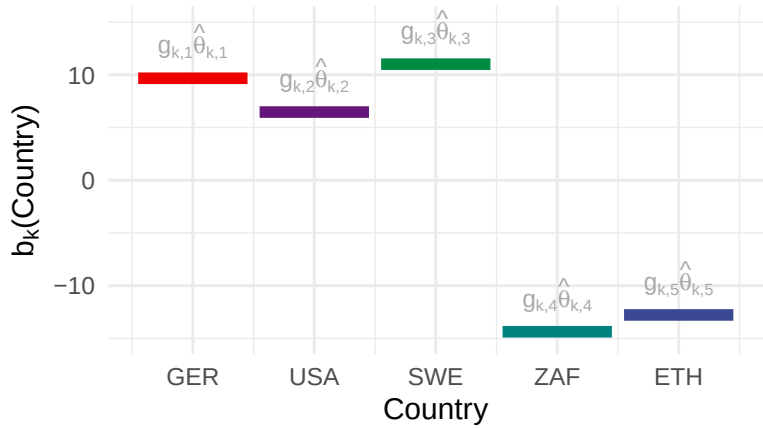


$$Z_k = \begin{array}{c} \begin{matrix} g_{k,GER} & g_{k,USA} & g_{k,SWE} & g_{k,ZAF} & g_{k,ETH} \end{matrix} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Categorical base learner

$$g_k(x) = (g_{k,1}(x), \ldots, g_{k,G}(x))^\top = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top, \quad x \in \{1, \ldots, G\}$$

$$Z_k \quad \Rightarrow \quad \hat{\boldsymbol{\theta}}_k = (Z_k^\top Z_k + K_k)^{-1} Z_k^\top y \quad \Rightarrow$$



(Row-wise) tensor product (RWTP) base learner

Combination (interaction) $b_k \odot b_l$ between to base learners b_k and b_l:

$$g_k(\mathbf{x}) \otimes g_l(\mathbf{x}) = (g_{k,1}(\mathbf{x})g_l(\mathbf{x})^\top, \ldots, g_{k,d_k}(\mathbf{x})g_l(\mathbf{x})^\top)^\top$$

Design matrix:

$$Z_k \odot Z_l = \begin{pmatrix} (g_k(\mathbf{x}^{(1)}) \otimes g_l(\mathbf{x}^{(1)}))^\top \\ \vdots \\ (g_k(\mathbf{x}^{(n)}) \otimes g_l(\mathbf{x}^{(n)}))^\top \end{pmatrix} \in \mathbb{R}^{n \times d_k d_l}$$

RWTP base learner

Example:

- b_k encodes the country: $g_k(x) = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top$
- b_l uses a B-spline basis for BMI: $g_l(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$

$$Z_k \odot Z_l = \begin{array}{c} \begin{array}{ccccc} g_{k,GER} & g_{k,USA} & g_{k,SWE} & g_{k,ZAF} & g_{k,ETH} \end{array} \\ \begin{pmatrix} 0 & g_l(x^{(1)}) & 0 & 0 & 0 \\ & & & & \end{pmatrix} \end{array}$$

RWTP base learner

Example:

- b_k encodes the country: $g_k(x) = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top$
- b_l uses a B-spline basis for BMI: $g_l(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$

$$Z_k \odot Z_l = \begin{array}{c} \begin{array}{ccccc} g_{k,GER} & g_{k,USA} & g_{k,SWE} & g_{k,ZAF} & g_{k,ETH} \end{array} \\ \begin{pmatrix} 0 & g_l(x^{(1)}) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_l(x^{(2)}) \end{pmatrix} \end{array}$$

RWTP base learner

Example:

- b_k encodes the country: $g_k(x) = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top$
- b_l uses a B-spline basis for BMI: $g_l(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$

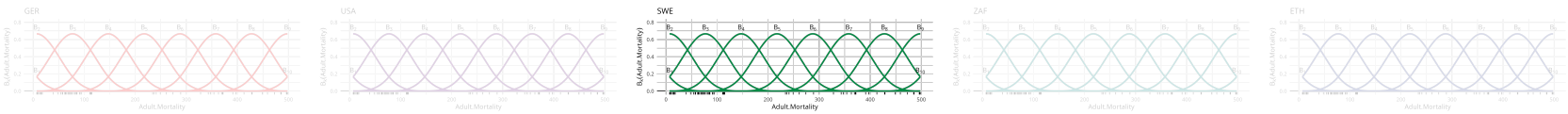
$$Z_k \odot Z_l = \begin{array}{c} \begin{array}{ccccc} g_{k,GER} & g_{k,USA} & g_{k,SWE} & g_{k,ZAF} & g_{k,ETH} \end{array} \\ \begin{pmatrix} 0 & g_l(x^{(1)}) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_l(x^{(2)}) \\ g_l(x^{(3)}) & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

RWTP base learner

Example:

- b_k encodes the country: $g_k(x) = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top$
- b_l uses a B-spline basis for BMI: $g_l(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$

$$Z_k \odot Z_l = \begin{array}{c} \begin{array}{ccccc} g_{k,GER} & g_{k,USA} & g_{k,SWE} & g_{k,ZAF} & g_{k,ETH} \end{array} \\ \begin{pmatrix} 0 & g_l(x^{(1)}) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_l(x^{(2)}) \\ g_l(x^{(3)}) & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & g_l(x^{(73)}) & 0 & 0 \end{pmatrix} \end{array}$$



RWTP base learner

Example:

- b_k encodes the country: $g_k(x) = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top$
- b_l uses a B-spline basis for BMI: $g_l(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$

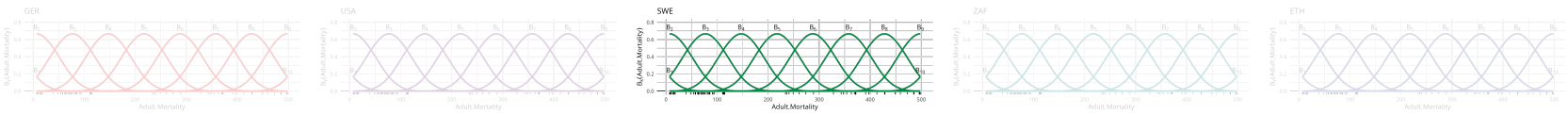
$$Z_k \odot Z_l = \begin{pmatrix} g_{k,GER} & g_{k,USA} & g_{k,SWE} & g_{k,ZAF} & g_{k,ETH} \\ 0 & g_l(x^{(1)}) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_l(x^{(2)}) \\ g_l(x^{(3)}) & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & g_l(x^{(73)}) & 0 & 0 \\ 0 & 0 & 0 & g_l(x^{(74)}) & 0 \end{pmatrix}$$

RWTP base learner

Example:

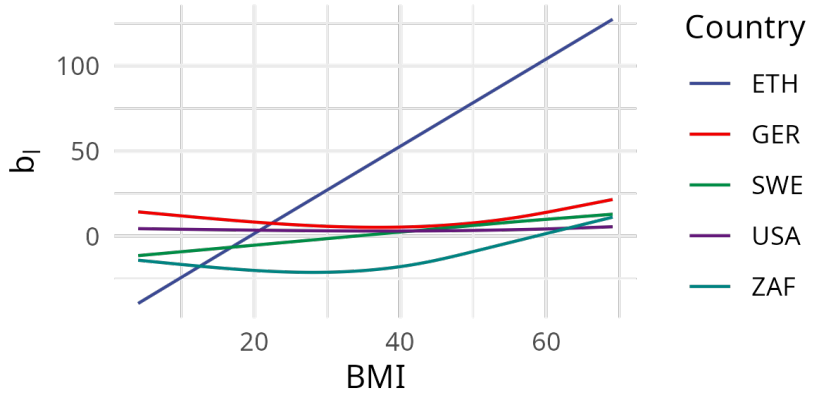
- b_k encodes the country: $g_k(x) = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top$
- b_l uses a B-spline basis for BMI: $g_l(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$

$$Z_k \odot Z_l = \begin{array}{c} \begin{matrix} g_{k,GER} & g_{k,USA} & g_{k,SWE} & g_{k,ZAF} & g_{k,ETH} \end{matrix} \\ \begin{pmatrix} 0 & g_l(x^{(1)}) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_l(x^{(2)}) \\ g_l(x^{(3)}) & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & g_l(x^{(73)}) & 0 & 0 \\ 0 & 0 & 0 & g_l(x^{(74)}) & 0 \\ 0 & 0 & g_l(x^{(75)}) & 0 & 0 \end{pmatrix} \end{array}$$

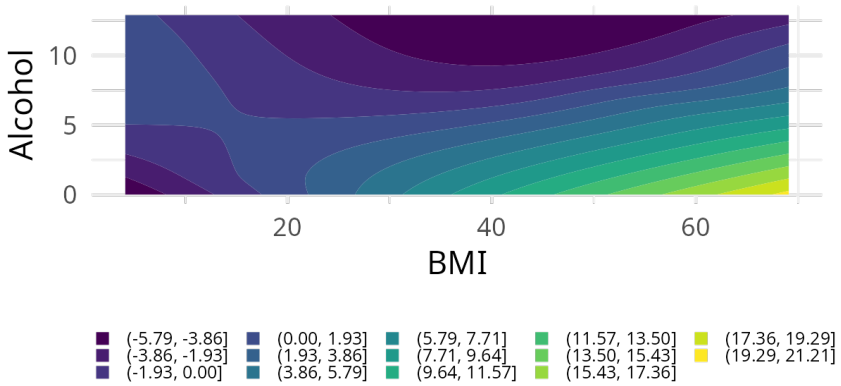


RWTP base learner

- Categoric - numeric base learner combination:



- Numeric - numeric base learner combination:



Efficiency

Efficiency problems of CWB

W.r.t. runtime:

- The base learner selection in each iteration is an extensive task.
- Gradient descent converges rather slowly compared to the optimizer like momentum or Nesterov's momentum.

  $\rightarrow$ Fitting the algorithm can be time consuming.

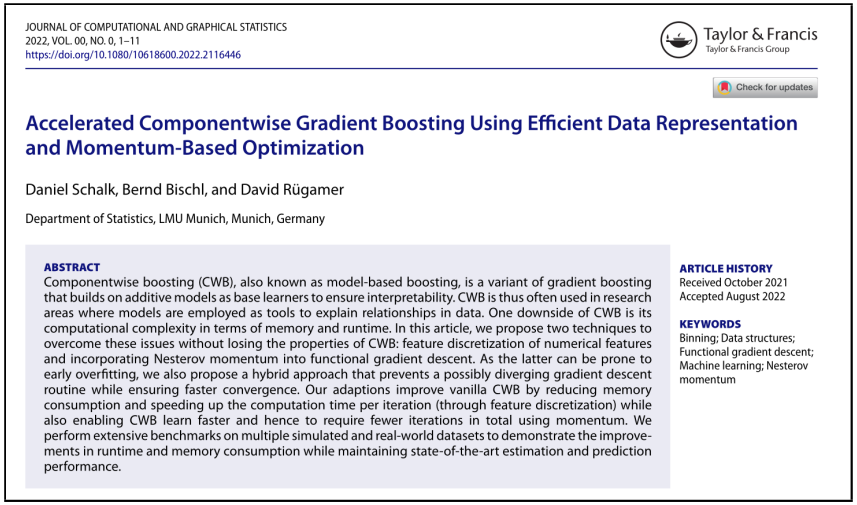
W.r.t. memory:

- Each base learner is required to store a design matrix.
- Additional information in each iteration about pseudo residuals, predicted scores, parameters, etc. may be stored.

  $\rightarrow$ The RAM is filled very fast.

$\Rightarrow$ Less attractive or infeasible to use CWB for medium- to large-scale applications.

Publication (Schalk et al., 2022a)



JOURNAL OF COMPUTATIONAL AND GRAPHICAL STATISTICS
2022, VOL. 00, NO. 0, 1–11
https://doi.org/10.1080/10618600.2022.2116446

Taylor & Francis
Taylor & Francis Group

Check for updates

Accelerated Componentwise Gradient Boosting Using Efficient Data Representation and Momentum-Based Optimization

Daniel Schalk, Bernd Bischl, and David Rügamer

Department of Statistics, LMU Munich, Munich, Germany

ABSTRACT
Componentwise boosting (CWB), also known as model-based boosting, is a variant of gradient boosting that builds on additive models as base learners to ensure interpretability. CWB is thus often used in research areas where models are employed as tools to explain relationships in data. One downside of CWB is its computational complexity in terms of memory and runtime. In this article, we propose two techniques to overcome these issues without losing the properties of CWB: feature discretization of numerical features and incorporating Nesterov momentum into functional gradient descent. As the latter can be prone to early overfitting, we also propose a hybrid approach that prevents a possibly diverging gradient descent routine while ensuring faster convergence. Our adaptions improve vanilla CWB by reducing memory consumption and speeding up the computation time per iteration (through feature discretization) while also enabling CWB learn faster and hence to require fewer iterations in total using momentum. We perform extensive benchmarks on multiple simulated and real-world datasets to demonstrate the improvements in runtime and memory consumption while maintaining state-of-the-art estimation and prediction performance.

ARTICLE HISTORY
Received October 2021
Accepted August 2022

KEYWORDS
Binning; Data structures; Functional gradient descent; Machine learning; Nesterov momentum

Contributions:

- **Accelerate the fitting process** of CWB by incorporating Nesterov's momentum.
- **Reduce the memory load** by implementing a more efficient data representation for numerical features.

Efficiency

Accelerating component-wise boosting

Accelerated gradient boosting machine

- Applying Nesterov's momentum in GB was already proposed by Biau et al. (2019) and refined in an algorithm called Accelerated Gradient Boosting Machine (AGBM) by Lu et al. (2020):

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

Accelerated gradient boosting machine

- Applying Nesterov's momentum in GB was already proposed by Biau et al. (2019) and refined in an algorithm called Accelerated Gradient Boosting Machine (AGBM) by Lu et al. (2020):

$$
\begin{aligned}
g^{[m]} &= (1-\theta_m) f^{[m]} + \theta_m h^{[m]} \\
f^{[m+1]} &= g^{[m]} + \nu b^{[m]} && \text{primary model} \\
h^{[m+1]} &= h^{[m]} + \gamma\nu/\theta_m b^{[m]}_{\text{cor}} && \text{momentum model}
\end{aligned}
$$

Accelerated gradient boosting machine

- Applying Nesterov's momentum in GB was already proposed by Biau et al. (2019) and refined in an algorithm called Accelerated Gradient Boosting Machine (AGBM) by Lu et al. (2020):

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

- A second base learner $b_{\text{cor}}^{[m]}$ is fitted to the accumulated previous errors ("error-corrected pseudo residuals") to accelerate the fitting into that direction.

Accelerated gradient boosting machine

- Applying Nesterov's momentum in GB was already proposed by Biau et al. (2019) and refined in an algorithm called Accelerated Gradient Boosting Machine (AGBM) by Lu et al. (2020):

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

- A second base learner $b_{\text{cor}}^{[m]}$ is fitted to the accumulated previous errors ("error-corrected pseudo residuals") to accelerate the fitting into that direction.
- In each iteration, two base learners $b^{[m]}$ and $b_{\text{cor}}^{[m]}$ are fitted.

Accelerated component-wise boosting

In Schalk et al. (2022a), we incorporated the updating scheme of AGBM into an accelerated CWB (ACWB) version:

- Both base learners $b^{[m]} \in \{b_1, \ldots, b_K\}$ and $b_{\text{cor}}^{[m]} \in \{b_1, \ldots, b_K\}$ that are added in one iteration are the result of a selection process w.r.t. to the minimal SSE.
- Update the parameter estimations accordingly to allow the estimation of partial feature effects.

Considering these points allows maintaining all advantages of CWB in ACWB.

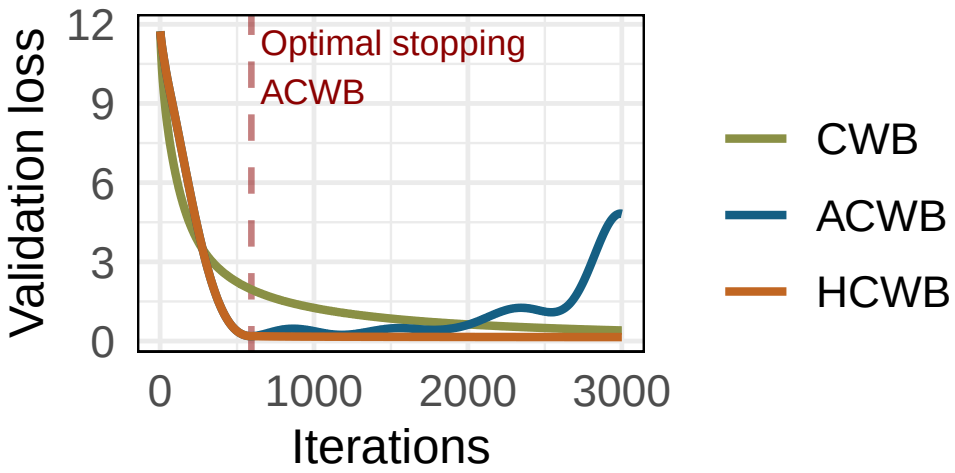
ACWB Algorithm

Input Data $\mathcal{D}$, learning rate ν, momentum parameter γ, number of boosting iterations M, loss function L, set of base learner $\mathcal{B}$
Output Prediction model $\hat{f}^{[M]}$ defined by fitted $\hat{\boldsymbol{\theta}}^{[1]}, \ldots, \hat{\boldsymbol{\theta}}^{[M]}$ and $\hat{\boldsymbol{\theta}}^{[1]}_{\text{cor}}, \ldots, \hat{\boldsymbol{\theta}}^{[M]}_{\text{cor}}$

procedure ACWB($\mathcal{D}, \nu, \gamma, L, \mathcal{B}$)
  Initialize $\hat{f}^{[0]}(\boldsymbol{x}) = \underset{c \in \mathbb{R}}{\arg\min}\, \mathcal{R}_{\text{emp}}(c, \mathcal{D})$; $\hat{h}^{[0]}(\boldsymbol{x}) = \hat{f}^{[0]}(\boldsymbol{x})$; $\hat{g}^{[0]}(\boldsymbol{x}) = \hat{f}^{[0]}(\boldsymbol{x})$;
  for $m \in \{1, \ldots, M\}$ **do**
    $\vartheta_m = \frac{2}{m+1}$
    $\hat{g}^{[m]}(\boldsymbol{x}) = (1 - \vartheta_m)\hat{f}^{[m-1]}(\boldsymbol{x}) + \vartheta_m \hat{h}^{[m-1]}(\boldsymbol{x})$
    $r^{[m](i)} = - \left. \frac{\partial L\left(y^{(i)}, f\left(\boldsymbol{x}^{(i)}\right)\right)}{\partial g(\boldsymbol{x}^{(i)})} \right|_{g = \hat{g}^{[m]}}, \quad \forall i \in \{1, \ldots, n\}$
    $(\hat{\boldsymbol{\theta}}^{[m]}, k^{[m]}) = \text{findBestBaselearner}(\boldsymbol{r}^{[m]}, \mathcal{B})$
    $\hat{f}^{[m]}(\boldsymbol{x}) = \hat{g}^{[m]}(\boldsymbol{x}) + \nu b_{k^{[m]}}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}^{[m]})$
    if $m > 1$ **then**
      $c^{[m](i)} = r^{[m](i)} + \frac{m}{m+1}\left(c^{[m-1](i)} - b_{k^{[m-1]}_{\text{cor}}}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}^{[m]}_{\text{cor}}\right), \quad \forall i \in \{1, \ldots, n\}$
    else
      $\boldsymbol{c}^{[m]} = \boldsymbol{r}^{[m]}$
    end if
    $(\hat{\boldsymbol{\theta}}^{[m]}_{\text{cor}}, k^{[m]}_{\text{cor}}) = \text{findBestBaselearner}(\boldsymbol{c}^{[m]}, \mathcal{B})$
    $\eta_m = \gamma\nu\vartheta_m^{-1}$
    $\hat{h}^{[m]}(\boldsymbol{x}) = \hat{h}^{[m-1]}(\boldsymbol{x}) + \eta_m b_{k^{[m]}_{\text{cor}}}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}^{[m]}_{\text{cor}})$
  end for
  return $\hat{f} = \hat{f}^{[M]}$
end procedure

Hybrid component-wise boosting

- ACWB can overfit or overshoot an optimal solution if not stopped early.
- Therefore, a hybrid CWB (HCWB) approach combines ACWB for an accelerated fitting in the beginning and CWB to fine-tune the model:



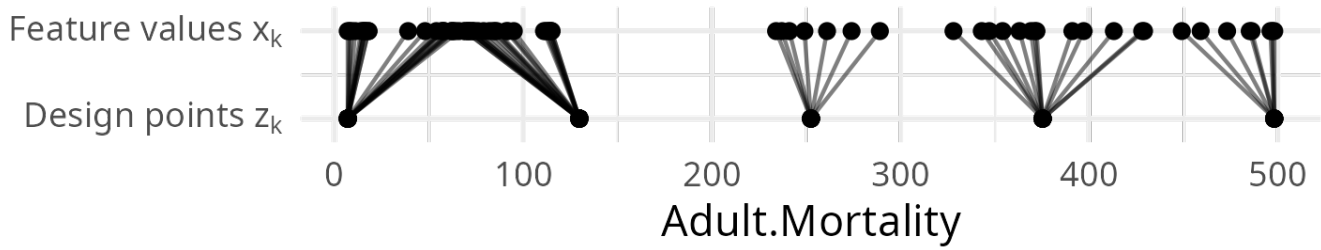
(Figure reference: Schalk et al. (2022a))

Efficiency

Reduced memory consumption for numeric base learner

Binning

- To reduce the memory consumption, we applied binning to operate on a reduced representation of Z_k.
- Binning is a technique that allows to represent the n values $x_k^{(1)}, \ldots, x_k^{(n)}$ of $\mathbf{x}_k$ 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 $\text{ind}_k^{(i)}$: $x_k^{(i)} \approx z_k^{(\text{ind}_k^{(i)})}$

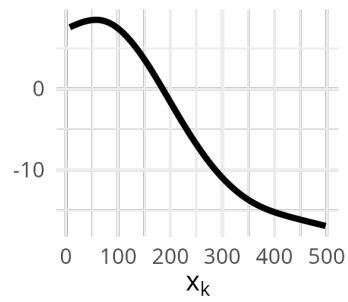


Binning

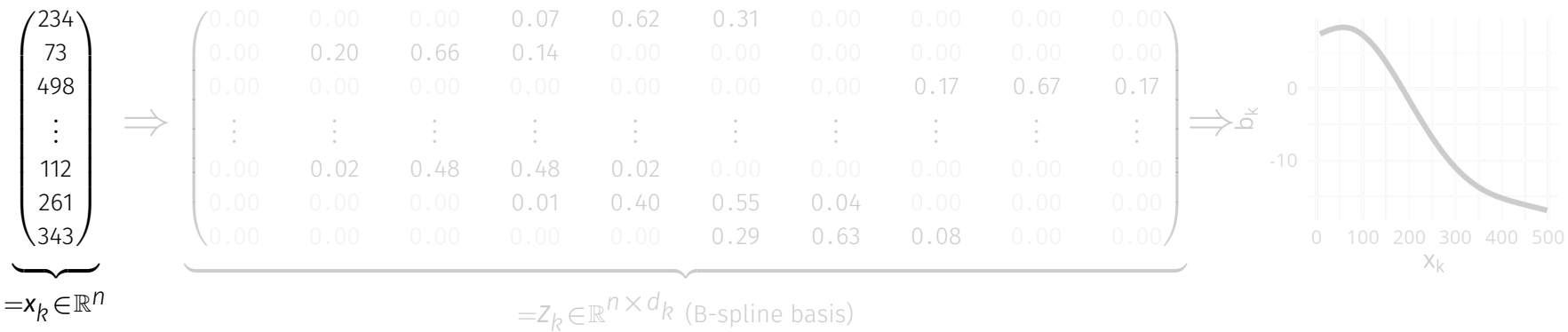
- Lang et al. (2014) used binning to discretize feature vectors to increase the efficiency of multilevel structured additive regression.
- Wood et al. (2017) applied binning to fit GAMs to gigadata and argue that the best approximation is achieved by setting $n^* = \sqrt{n}$.
- Li and Wood (2020) presented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

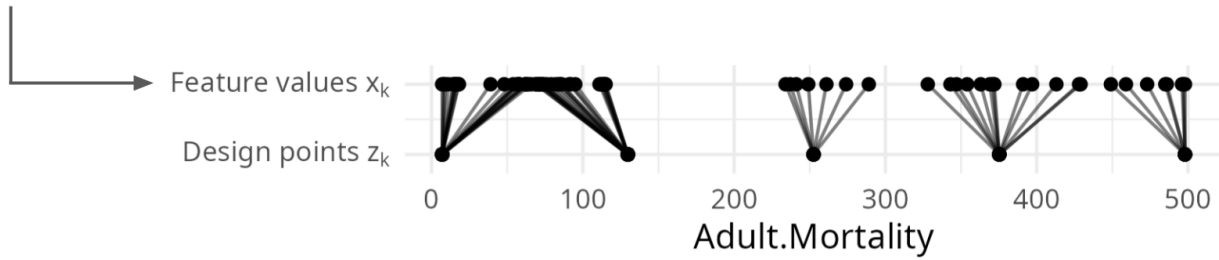
Initializing a base learner with binning

$$\underbrace{\begin{pmatrix} 234 \\ 73 \\ 498 \\ \vdots \\ 112 \\ 261 \\ 343 \end{pmatrix}}_{=x_k \in \mathbb{R}^n} \Rightarrow \underbrace{\begin{pmatrix} 0.00 & 0.00 & 0.00 & 0.07 & 0.62 & 0.31 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.20 & 0.66 & 0.14 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.17 & 0.67 & 0.17 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.00 & 0.02 & 0.48 & 0.48 & 0.02 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.01 & 0.40 & 0.55 & 0.04 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.29 & 0.63 & 0.08 & 0.00 & 0.00 \end{pmatrix}}_{=Z_k \in \mathbb{R}^{n \times d_k} \text{ (B-spline basis)}} \Rightarrow$$

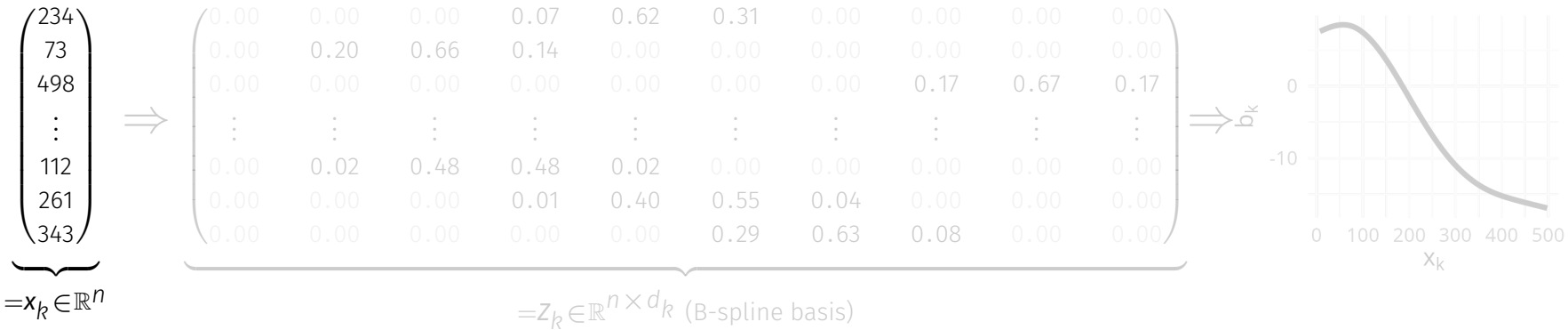


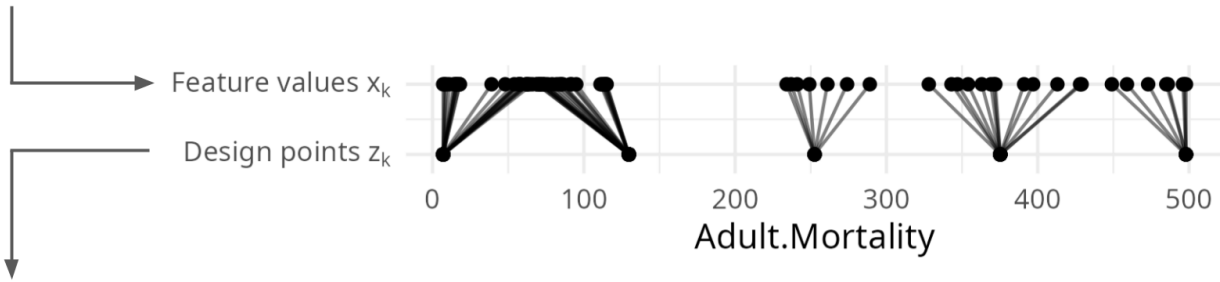
Initializing a base learner with binning

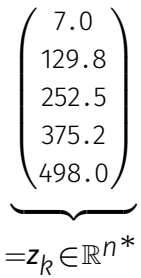




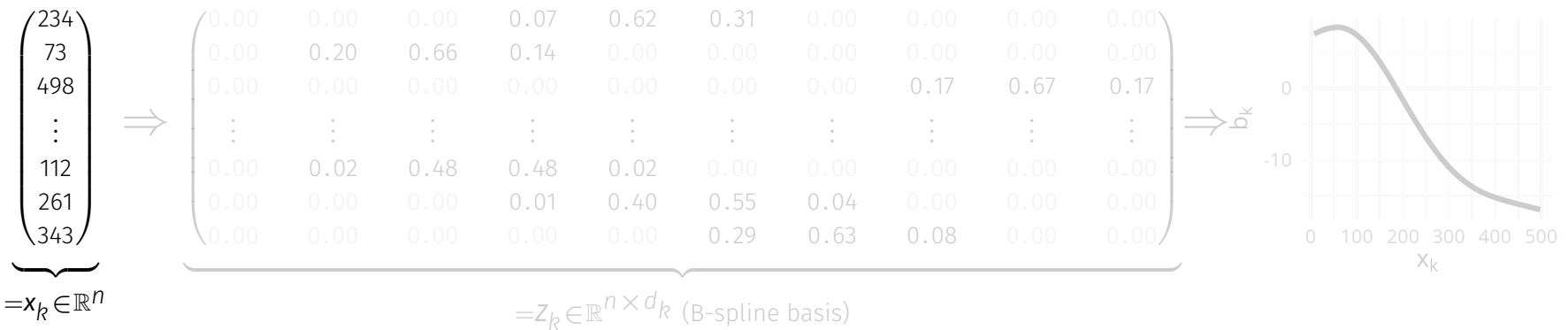
Initializing a base learner with binning

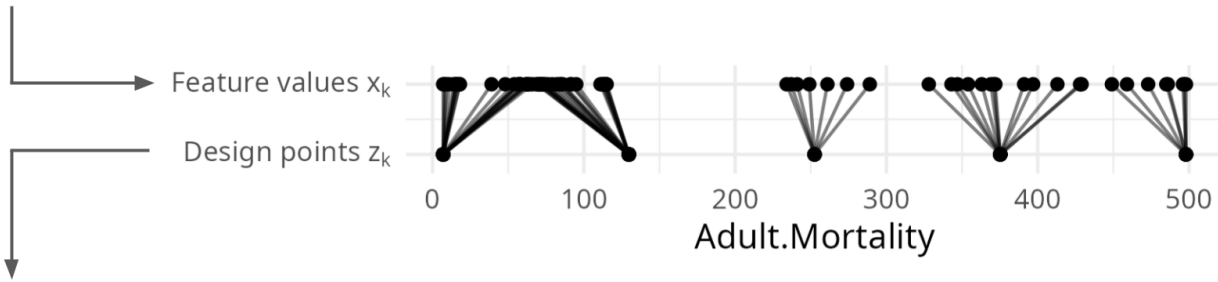


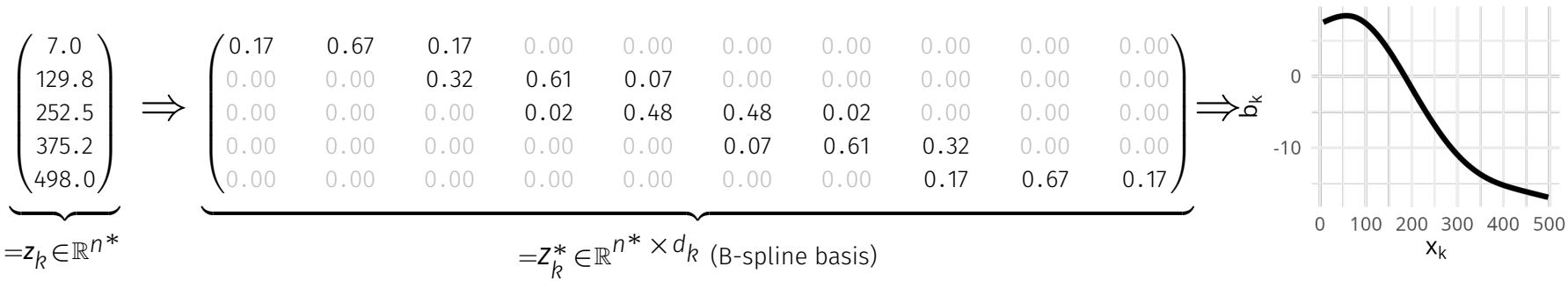




Initializing a base learner with binning







Binning in CWB

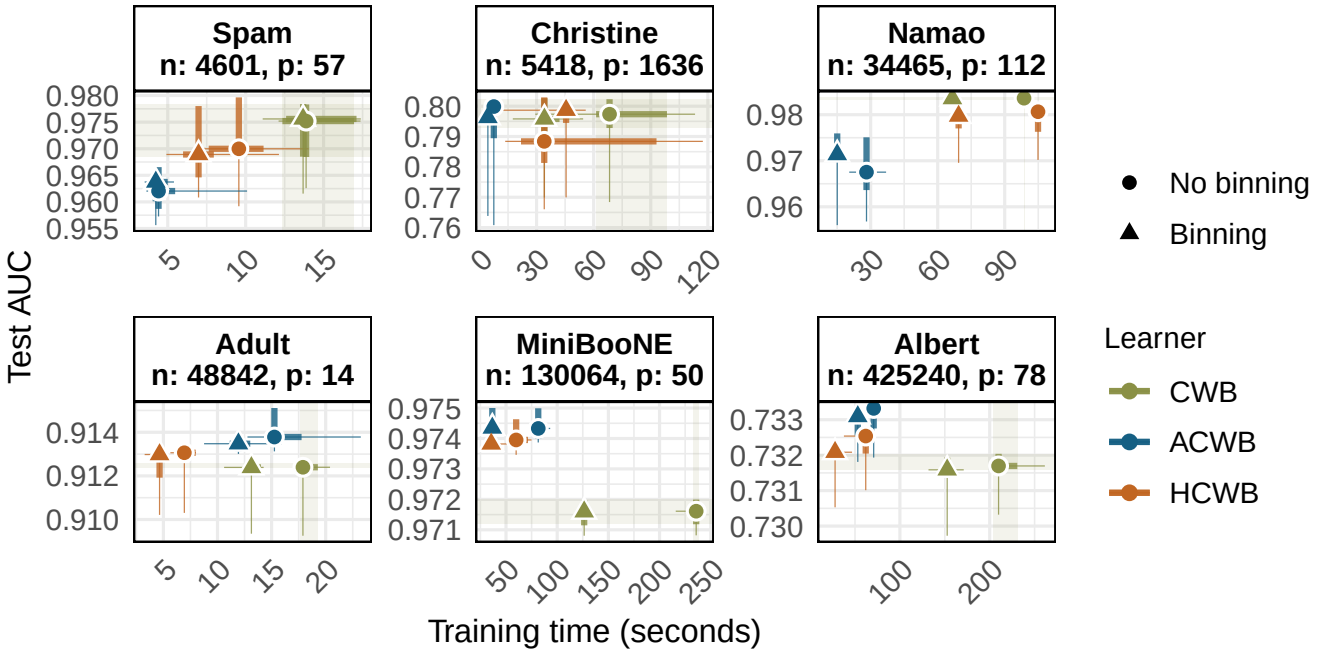
Each (univariate) base learner b_k can apply binning individually to:

- Reduce the memory consumption of $\boldsymbol{Z}_k$.
- Use optimized matrix operations m_M and m_V based on binning:
  - Speed up the **model initialization**: Calculation of information that do not depend on m like $\boldsymbol{Z}_1^*, \ldots, \boldsymbol{Z}_K^*$ and $\boldsymbol{F}_k^{-1} = (m_M(\boldsymbol{Z}_k^*, \boldsymbol{Z}_k^*) + \boldsymbol{K}_k)^{-1}$, that do not depend on m prior to the fitting
  - Speed up the **fitting process**: Estimate parameter with $\hat{\boldsymbol{\theta}}_k^{[m]} = \boldsymbol{F}_k^{-1} m_V(\boldsymbol{Z}_k^*, \boldsymbol{r}^{[m]})$

Efficiency

Benchmark result and big data examples

Benchmark comparisons of CWB variants

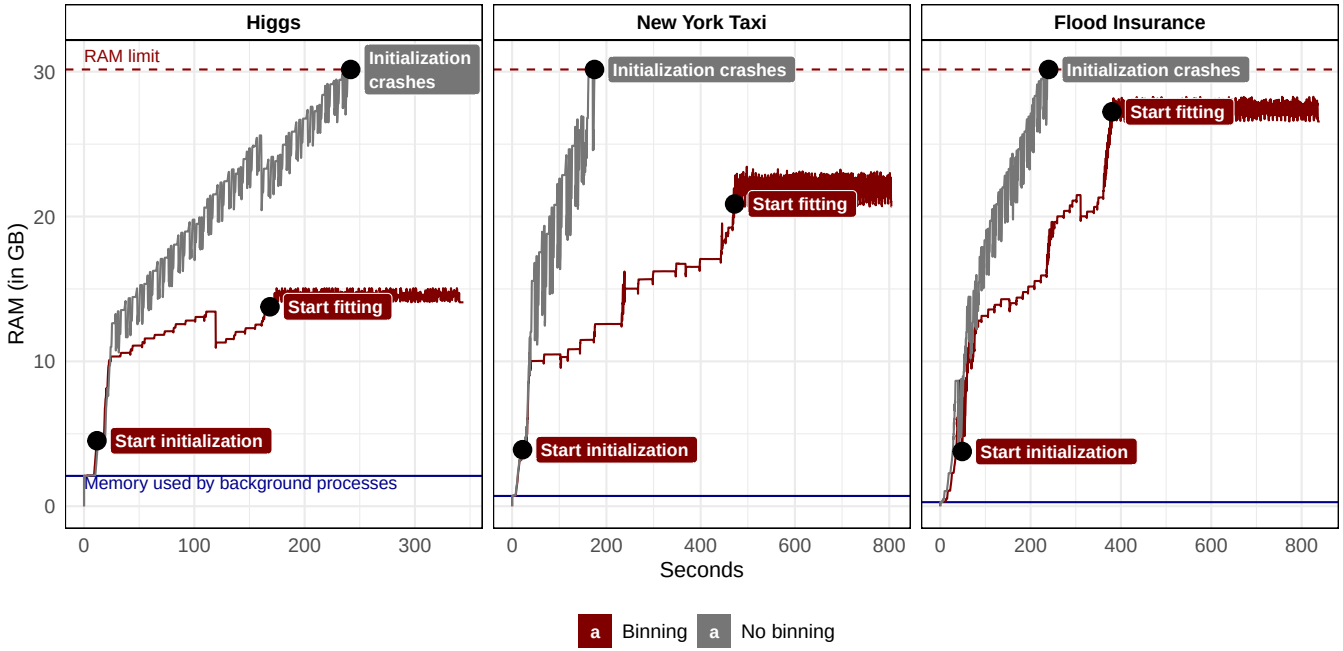


(Figure[1] reference: Schalk et al. (2022a))

- Using binning improves the runtime.
- Accelerating CWB can speed up the training time further without sacrificing predictive performance.

[1] 5-fold cross-validation with early stopping in each fold.

Memory consumption for bigger data sets



(Figure[1] reference: Schalk et al. (2022a))

- **Higgs**: 2.4 GB, $n = 11 \cdot 10^6$, $p = 29$ (all numeric)
- **NYC Taxi**: 3.3 GB, $n = 24.3 \cdot 10^6$, $p = 22$ (all numeric)
- **Flood Insurance**: 3.4 GB, $n = 14.5 \cdot 10^6$, $p = 50$ (29 are numeric)

[1] Fitting was conducted for 50 iterations.

Distributed computing

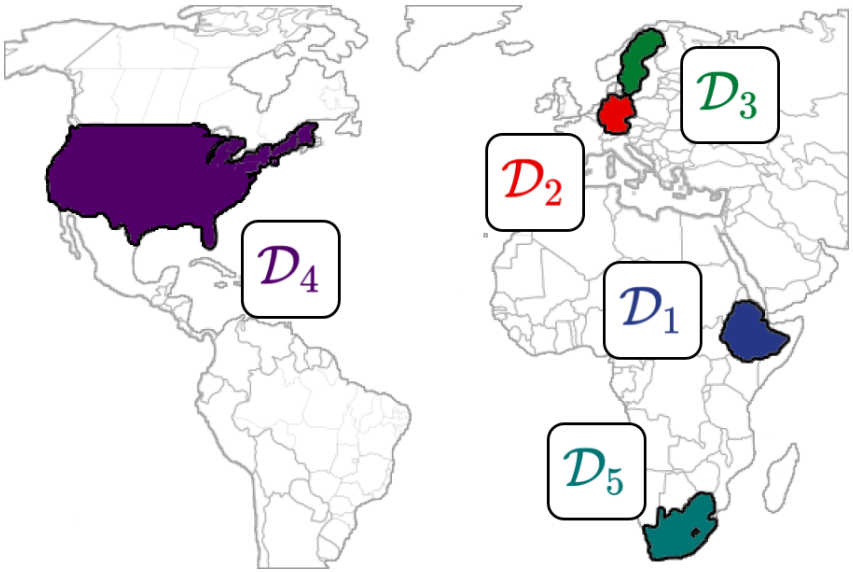
Distributed data set

Assume the `Country` column is not present in the data set and each country holds a its own partition:

Life.expectancy	Country	Year	BMI	Adult.Mortality	Data set
51.2	ETH	2000	12.3	391	$\mathcal{D}_1$
⋮	⋮	⋮	⋮	⋮	
64.8	ETH	2015	17.6	225	
78.0	GER	2000	55.1	95	$\mathcal{D}_2$
⋮	⋮	⋮	⋮	⋮	
81.0	GER	2015	62.3	68	
79.6	SWE	2000	52.8	73	$\mathcal{D}_3$
⋮	⋮	⋮	⋮	⋮	
82.4	SWE	2015	59.5	53	
76.8	USA	2000	6.1	114	$\mathcal{D}_4$
⋮	⋮	⋮	⋮	⋮	
79.3	USA	2015	69.6	13	
57.3	ZAF	2000	4.1	397	$\mathcal{D}_5$
⋮	⋮	⋮	⋮	⋮	
62.9	ZAF	2015	51.1	328	

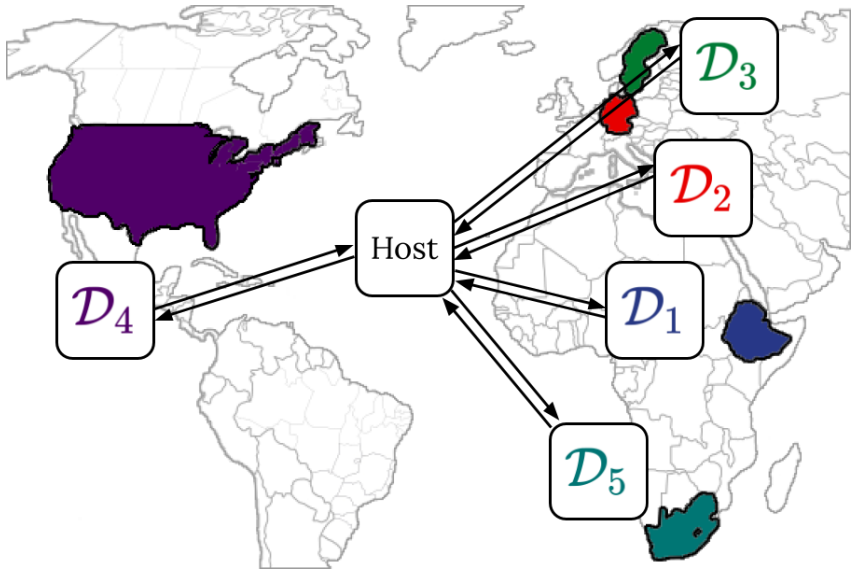
$\Rightarrow$ Due to privacy reasons, it is not allowed to share and merge these data sets to $\mathcal{D} = \cup_{s=1}^{S} \mathcal{D}_s$.

Distributed data setup



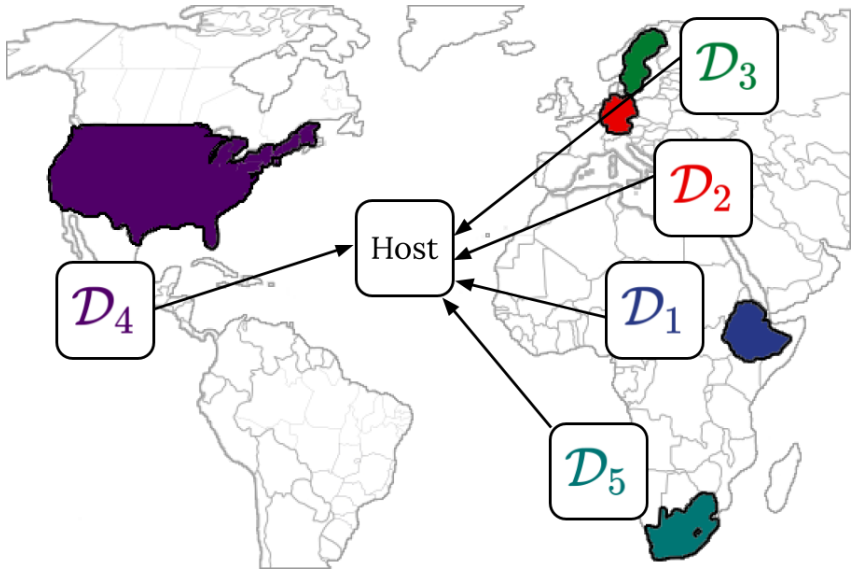
- Data is partitioned **horizontally**: Each of the S sites hold the same features but different observations.

Distributed data setup



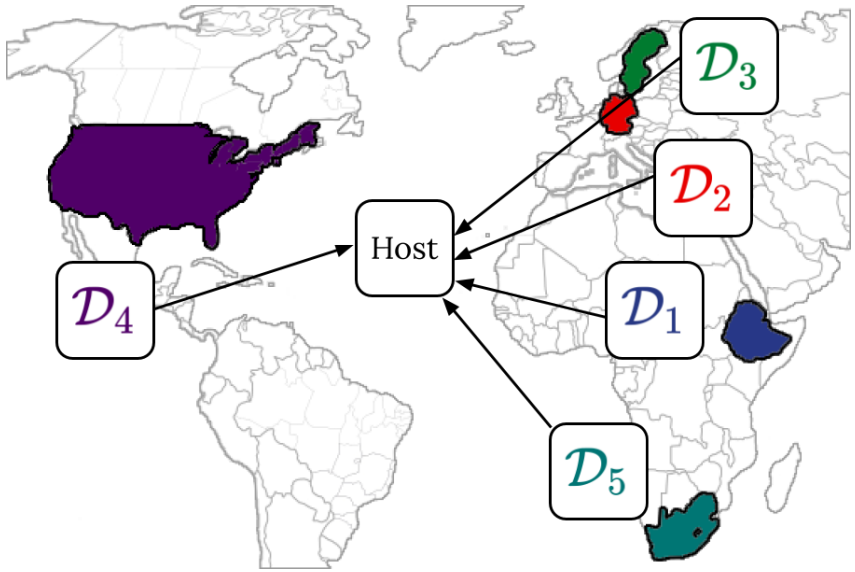
- A host controls the communication with the sites, the sites cannot communicate with each other.
- The host is the vulnerable component.

Distributed data setup



- The communicated data from the sites must ensure privacy of the original data sets. What is allowed to be shared?
  - Aggregated data that does not allow reconstructing parts of the original data set.
  - Encrypted data.

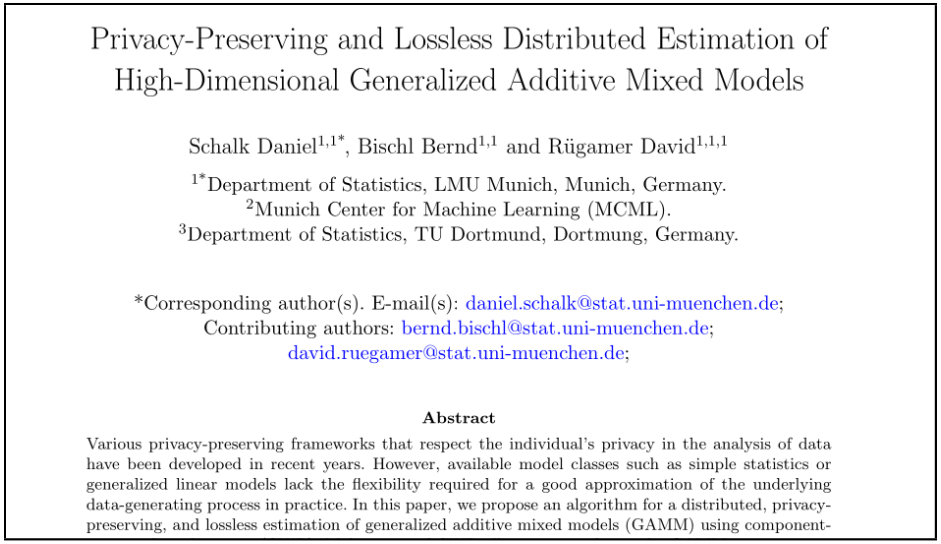
Distributed data setup



- The communicated data from the sites must ensure privacy of the original data sets. What is allowed to be shared?
  - Aggregated data that does not allow reconstructing parts of the original data set.
  - Encrypted data

$\Rightarrow$ Is it still possible to fit a model with CWB?

Publication (Schalk et al., 2023a)



Privacy-Preserving and Lossless Distributed Estimation of High-Dimensional Generalized Additive Mixed Models

Schalk Daniel[1,1*], Bischl Bernd[1,1] and Rügamer David[1,1,1]

[1*]Department of Statistics, LMU Munich, Munich, Germany.
[2]Munich Center for Machine Learning (MCML).
[3]Department of Statistics, TU Dortmund, Dortmung, Germany.

*Corresponding author(s). E-mail(s): daniel.schalk@stat.uni-muenchen.de;
Contributing authors: bernd.bischl@stat.uni-muenchen.de;
david.ruegamer@stat.uni-muenchen.de;

Abstract

Various privacy-preserving frameworks that respect the individual's privacy in the analysis of data have been developed in recent years. However, available model classes such as simple statistics or generalized linear models lack the flexibility required for a good approximation of the underlying data-generating process in practice. In this paper, we propose an algorithm for a distributed, privacy-preserving, and lossless estimation of generalized additive mixed models (GAMM) using component-

Conbributions:

- Provide a **distributed, privacy-preserving, and lossless CWB algorithm**:

$$\text{distCWB}(\mathcal{D}_1, \ldots, \mathcal{D}_S) = \text{CWB}(\mathcal{D})$$

- **Allow for site-specific corrections** to account for heterogeneity in the data.

Site-specific vs. main effects

- In the distributed setup, we denote b_k as **shared** or **main effect** that is equal between all sites.
- Further, a main effects b_k is extended by **site-specific effects** $b_{k,s}$ to allow a site-specific correction.

Site-specific vs. main effects

- In the distributed setup, we denote b_k as **shared** or **main effect** that is equal between all sites.
- Further, a main effects b_k is extended by **site-specific effects** $b_{k,s}$ to allow a site-specific correction.

$$f(\mathbf{x}|s) = f_0 + \sum_{k=1}^{K} \left(b_k(\mathbf{x}|\boldsymbol{\theta}_k) + \sum_{s'=1}^{S} \mathbb{1}_{\{s=s'\}} b_{k,s}(\mathbf{x}|\boldsymbol{\theta}_{k_\times,s}) \right)$$

$$= f_0 + \sum_{k=1}^{K} \underbrace{b_k(\mathbf{x}|\boldsymbol{\theta}_k)}_{\text{main effect}} + \underbrace{(b_0 \odot b_k)(\mathbf{x}|\boldsymbol{\theta}_{k_\times})}_{=b_{k_\times},\ \text{site-specific effects}}$$

- Equal to CWB with base learners $b_1, \ldots, b_K$

Site-specific vs. main effects

- In the distributed setup, we denote b_k as **shared** or **main effect** that is equal between all sites.
- Further, a main effects b_k is extended by **site-specific effects** $b_{k,s}$ to allow a site-specific correction.

$$
\begin{aligned}
f(\mathbf{x}|s) &= f_0 + \sum_{k=1}^{K} \left(b_k(\mathbf{x}|\boldsymbol{\theta}_k) + \sum_{s'=1}^{S} \mathbb{1}_{\{s=s'\}} b_{k,s}(\mathbf{x}|\boldsymbol{\theta}_{k_\times,s}) \right) \\
&= f_0 + \sum_{k=1}^{K} \underbrace{b_k(\mathbf{x}|\boldsymbol{\theta}_k)}_{\text{main effect}} + \underbrace{(b_0 \odot b_k)(\mathbf{x}|\boldsymbol{\theta}_{k_\times})}_{=b_{k_\times},\ \text{site-specific effects}}
\end{aligned}
$$

- Equal to CWB with base learners $b_1, \ldots, b_K$

Site-specific vs. main effects

- In the distributed setup, we denote b_k as **shared** or **main effect** that is equal between all sites.
- Further, a main effects b_k is extended by **site-specific effects** $b_{k,s}$ to allow a site-specific correction.

$$
\begin{aligned}
f(\mathbf{x}|s) &= f_0 + \sum_{k=1}^{K} \left(b_k(\mathbf{x}|\boldsymbol{\theta}_k) + \sum_{s'=1}^{S} \mathbb{1}_{\{s=s'\}} b_{k,s}(\mathbf{x}|\boldsymbol{\theta}_{k_\times,s}) \right) \\
&= f_0 + \sum_{k=1}^{K} \underbrace{b_k(\mathbf{x}|\boldsymbol{\theta}_k)}_{\text{main effect}} + \underbrace{(b_0 \odot b_k)(\mathbf{x}|\boldsymbol{\theta}_{k_\times})}_{=b_{k_\times},\ \text{site-specific effects}}
\end{aligned}
$$

- Equal to CWB with base learners $b_1, \ldots, b_K$ and
- RWTP base learners $b_{k_\times} = b_0 \odot b_k$, $k = 1, \ldots, K$, with b_0 modelling a latent categorical feature site $x_0 \in \{1, \ldots, S\}$.

Distributed CWB

Estimation of main effects

Estimation of main effects

Algorithm 2 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 $\hat{f} = \hat{f}^{[M]}$

procedure CWB($\mathcal{D}, \nu, M, L, b_1, \ldots, b_K$)

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

  while $m \leq M$ **do**

    $r^{[m](i)} = - \left. \frac{\partial L(y^{(i)}, f(\mathbf{x}^{(i)}))}{\partial f(\mathbf{x}^{(i)})} \right|_{f=\hat{f}^{[m-1]}}, \quad \forall i \in \{1, \ldots, n\}$

    for $k \in \{1, \ldots, K\}$ **do**

      $\hat{\boldsymbol{\theta}}_k^{[m]} = \left(Z_k^\top Z_k + K_k \right)^{-1} Z_k^\top \boldsymbol{r}^{[m]}$ ← Distribute

      $\text{SSE}_k = \sum_{i=1}^{n} (r^{[m](i)} - b_k(\mathbf{x}^{(i)}|\hat{\boldsymbol{\theta}}_k^{[m]}))^2$

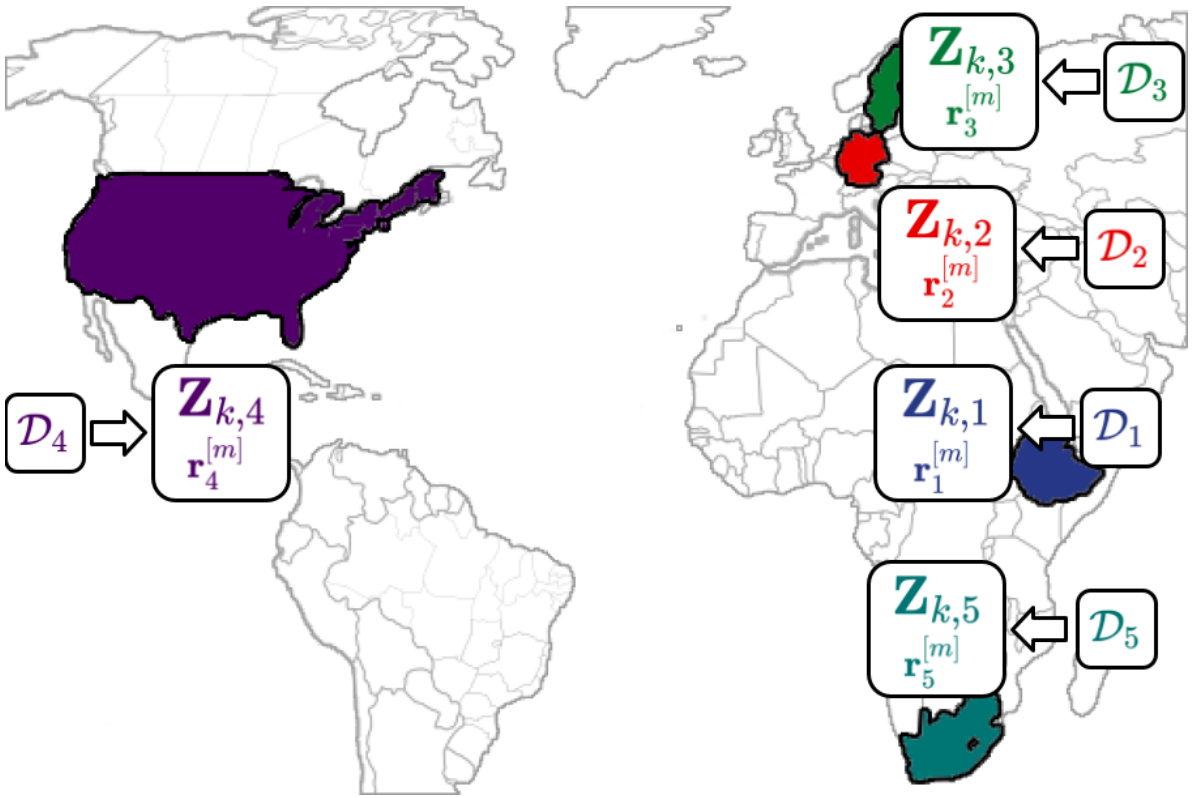
    $k^{[m]} = \arg\min_{k \in \{1, \ldots, K\}} \text{SSE}_k$

    $\hat{f}^{[m]}(\mathbf{x}) = \hat{f}^{[m-1]}(\mathbf{x}) + \nu b_{k^{[m]}}(\mathbf{x}|\hat{\boldsymbol{\theta}}_{k^{[m]}}^{[m]})$

  return $\hat{f} = \hat{f}^{[M]}$

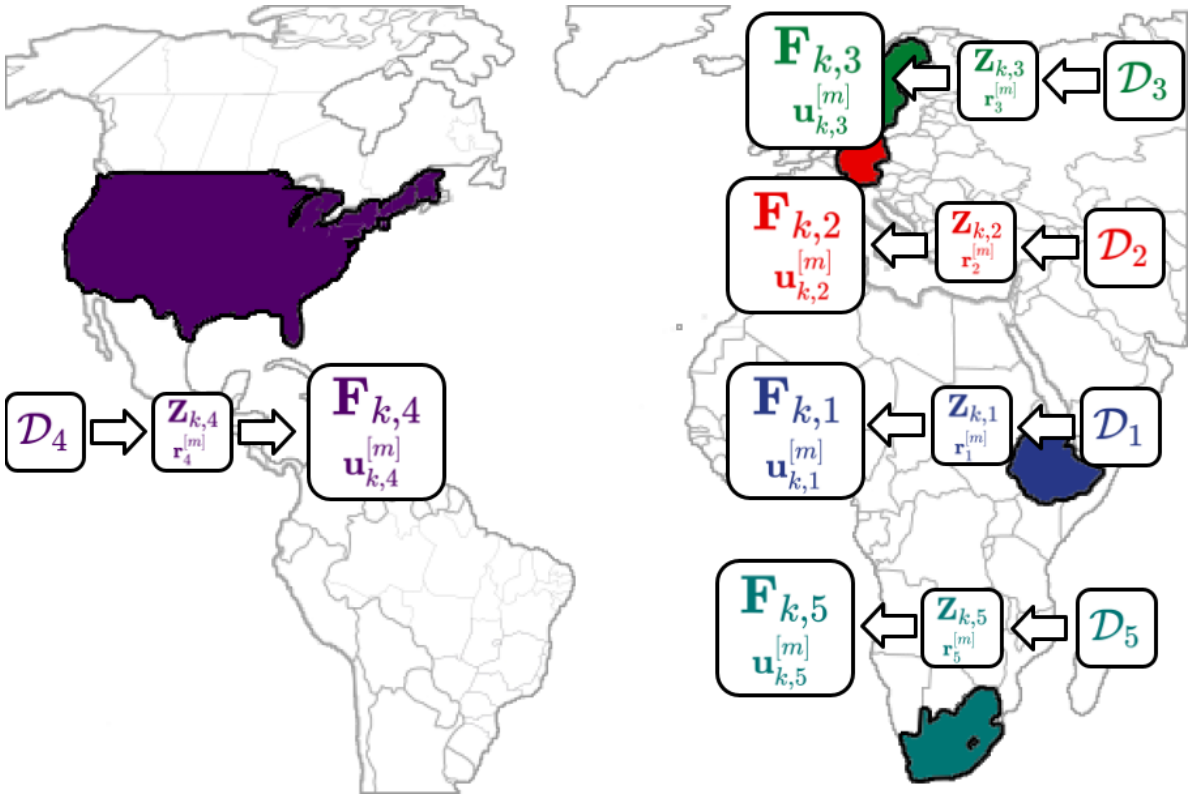
Estimation of main effects

- Each site s holds a design matrix $Z_{k,s}$ for the k^{th} base learner and a slice of the pseudo residuals $r_s^{[m]}$.



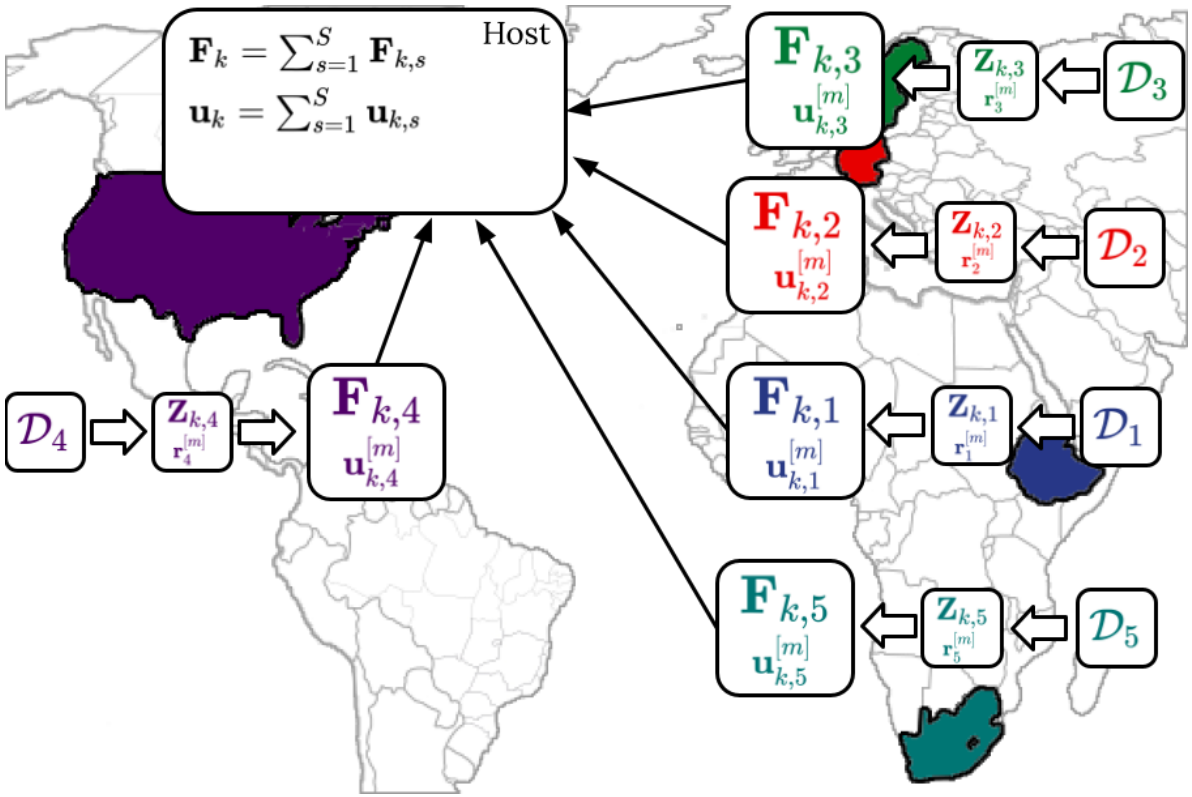
Estimation of main effects

- Each site calculates $F_{k,s} = Z_{k,s}^T Z_{k,s}$ and $u_{k,s}^{[m]} = Z_{k,s}^T r_k^{[m]}$.



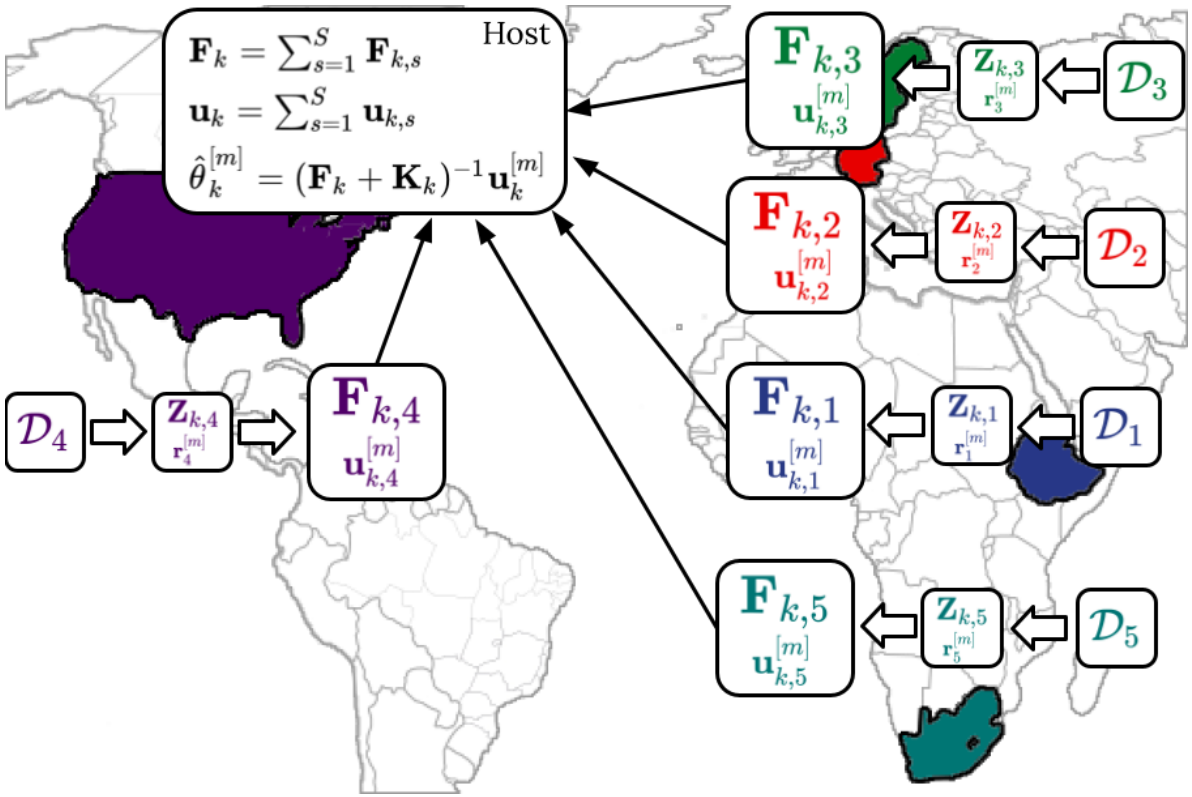
Estimation of main effects

- The sites are allowed to communicate $F_{k,s}$ and $u_{k,s}^{[m]}$ as long as "enough observations" are used. The host calculates $F_k = \sum_{s=1}^{S} F_{k,s}$ and $u_k^{[m]} = \sum_{s=1}^{S} u_{k,s}^{[m]}$.



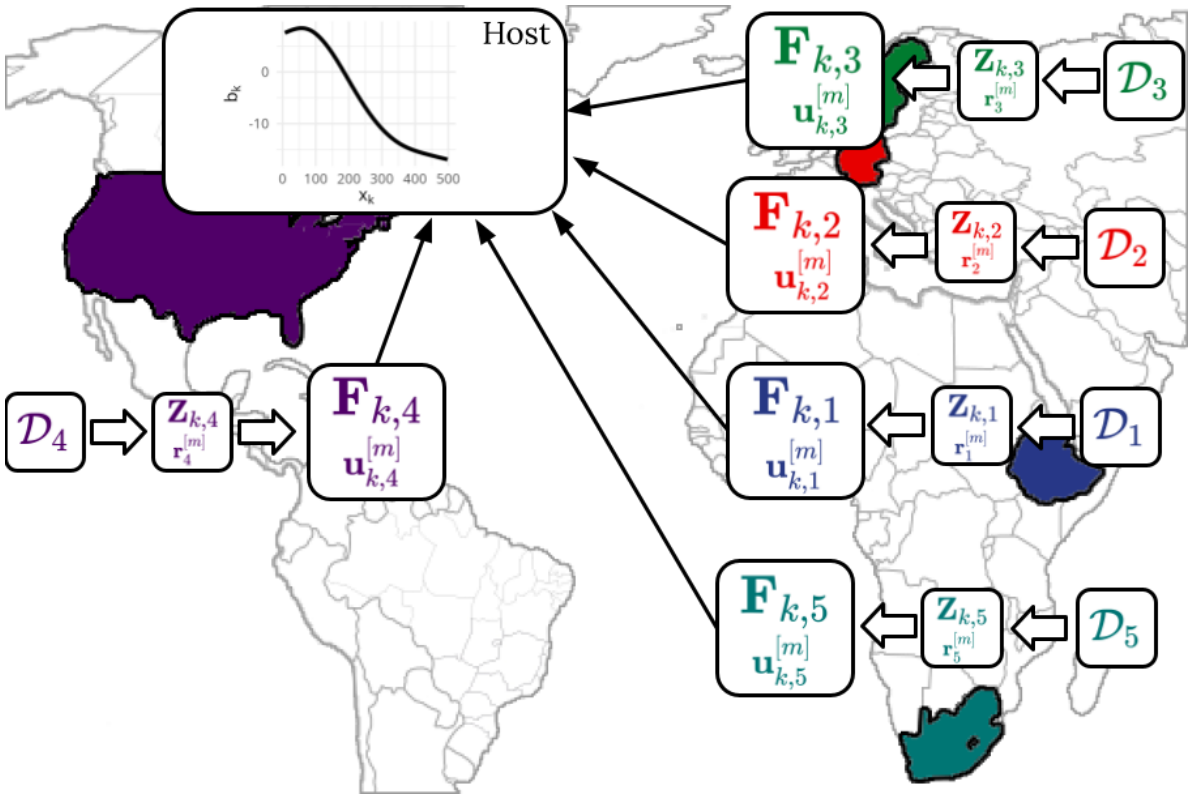
Estimation of main effects

- Finally, the host can estimate $\hat{\boldsymbol{\theta}}_k^{[m]} = (F_k + K_k)^{-1} u_k$.



Estimation of main effects

- Finally, the host can estimate $\hat{\boldsymbol{\theta}}_k^{[m]} = (F_k + K_k)^{-1} \boldsymbol{u}_k$. (The estimated main effects must be communicated back to sites to update the pseudo residuals $\boldsymbol{r}_k^{[m]}$.)



Estimation of main effects

Algorithm 3 Distributed Effect Estimation (Karr et al., 2005).
The line prefixes `[S]` and `[H]` indicate whether the operation is conducted at the sites (`[S]`) or at the host (`[H]`).

Input Sites design matrices $\boldsymbol{Z}_{k,1}, \ldots, \boldsymbol{Z}_{k,S}$, response vectors $\boldsymbol{r}_1^{[m]}, \ldots, \boldsymbol{r}_S^{[m]}$ and an optional penalty matrix $\boldsymbol{K}_k$.
Output Estimated parameter vector $\hat{\boldsymbol{\theta}}_k$.

procedure distFit($\boldsymbol{Z}_{k,1}, \ldots, \boldsymbol{Z}_{k,S}, \boldsymbol{r}_1^{[m]}, \ldots, \boldsymbol{r}_S^{[m]}, \boldsymbol{K}_k$)
  for $s \in \{1, \ldots, S\}$ do
    [S] $\boldsymbol{F}_{k,s} = \boldsymbol{Z}_{k,s}^\top \boldsymbol{Z}_{k,s}$
    [S] $\boldsymbol{u}_{k,s} = \boldsymbol{Z}_{k,s}^\top \boldsymbol{r}_s^{[m]}$
    [S] Communicate $\boldsymbol{F}_{k,s}$ and $\boldsymbol{u}_{k,s}$ to the host
  [H] $\boldsymbol{F}_k = \sum_{s=1}^{S} \boldsymbol{F}_{k,s} + \boldsymbol{K}_k$
  [H] $\boldsymbol{u}_k = \sum_{s=1}^{S} \boldsymbol{u}_{k,s}$
  [H] return $\hat{\boldsymbol{\theta}}_k = \boldsymbol{F}_k^{-1} \boldsymbol{u}_k$

Estimation of main effects

Substituting $\hat{\boldsymbol{\theta}}_k^{[m]} = \left(Z_k^\top Z_k + K_k\right)^{-1} Z_k^\top \boldsymbol{r}^{[m]}$ with **distFit** in each iteration m gives a first lossless distributed CWB algorithm.

Algorithm 2 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 $\hat{f} = \hat{f}^{[M]}$

procedure CWB($\mathcal{D}, \nu, M, L, b_1, \ldots, b_K$)

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

  while $m \leq M$ **do**

    $r^{[m](i)} = - \left. \frac{\partial L(y^{(i)}, f(\mathbf{x}^{(i)}))}{\partial f(\mathbf{x}^{(i)})} \right|_{f = \hat{f}^{[m-1]}}, \quad \forall i \in \{1, \ldots, n\}$

    for $k \in \{1, \ldots, K\}$ **do**

      $\hat{\boldsymbol{\theta}}_k^{[m]} = \left(Z_k^\top Z_k + K_k\right)^{-1} Z_k^\top \boldsymbol{r}^{[m]}$ ← Distribute

      $\text{SSE}_k = \sum_{i=1}^{n} (r^{[m](i)} - b_k(\mathbf{x}^{(i)}|\hat{\boldsymbol{\theta}}_k^{[m]}))^2$

    $k^{[m]} = \arg\min_{k \in \{1, \ldots, K\}} \text{SSE}_k$

    $\hat{f}^{[m]}(\mathbf{x}) = \hat{f}^{[m-1]}(\mathbf{x}) + \nu b_{k^{[m]}}(\mathbf{x}|\hat{\boldsymbol{\theta}}_{k^{[m]}}^{[m]})$

  return $\hat{f} = \hat{f}^{[M]}$

Estimation of main effects

Substituting $\hat{\boldsymbol{\theta}}_k^{[m]} = \left(Z_k^\top Z_k + K_k\right)^{-1} Z_k^\top \boldsymbol{r}^{[m]}$ with **distFit** in each iteration m gives a first lossless distributed CWB algorithm.

Algorithm 4 Distributed CWB algorithm

Input Site data $\mathcal{D}_1, \ldots, \mathcal{D}_K$, learning rate ν, number of boosting iterations M, loss function L, base learners $b_1, \ldots, b_K$

Output Model $\hat{f} = \hat{f}^{[M]}$

procedure distCWB($\mathcal{D}_1, \ldots, \mathcal{D}_K, \nu, M, L, b_1, \ldots, b_K$)
  Initialize: $f_0 = \hat{f}^{[0]}(\mathrm{x}) = \arg\min_{c \in \mathcal{Y}} \mathcal{R}_{\text{emp}}(c|\mathcal{D})$
  while $m \leq M$ **do**
    $r^{[m](i)} = - \left. \frac{\partial L(y^{(i)}, f(\mathrm{x}^{(i)}))}{\partial f(\mathrm{x}^{(i)})} \right|_{f=\hat{f}^{[m-1]}}, \quad \forall i \in \{1, \ldots, n\}$
    for $k \in \{1, \ldots, K\}$ **do**
      $\hat{\boldsymbol{\theta}}_k^{[m]} = \text{distFit}(\mathrm{Z}_{k,1}, \ldots, \mathrm{Z}_{k,S}, \mathrm{r}_1^{[m]}, \ldots, \mathrm{r}_S^{[m]}, \mathrm{K}_k)$
      $\text{SSE}_k = \text{aggregate}(\text{get}(\text{SSE}_{k,1}, \ldots, \text{SSE}_{k,S}))$
    $k^{[m]} = \arg\min_{k \in \{1,\ldots,K\}} \text{SSE}_k$
    $\hat{f}^{[m]}(\mathrm{x}) = \hat{f}^{[m-1]}(\mathrm{x}) + \nu b_{k^{[m]}}(\mathrm{x}|\hat{\boldsymbol{\theta}}_{k^{[m]}}^{[m]})$
  return $\hat{f} = \hat{f}^{[M]}$

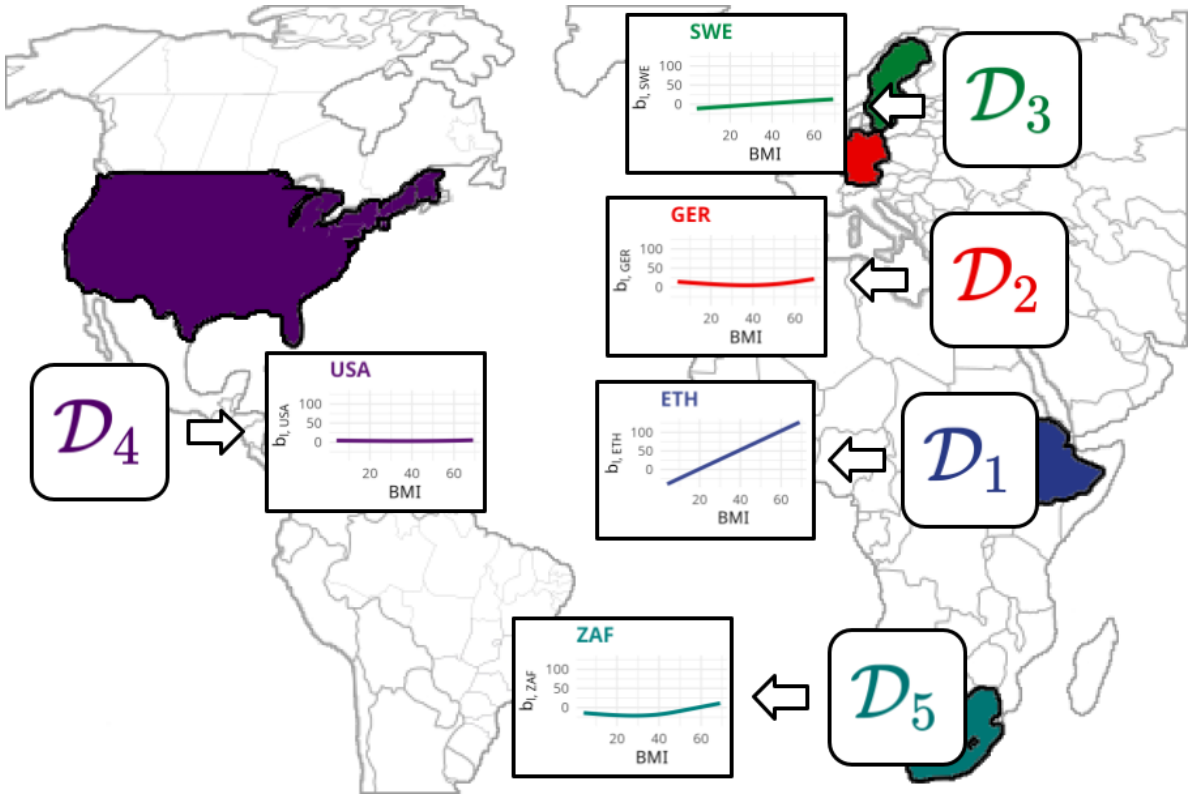
Note: The communication costs are a third resource that affects the efficiency.

Distributed CWB

Estimation of site-specific effects

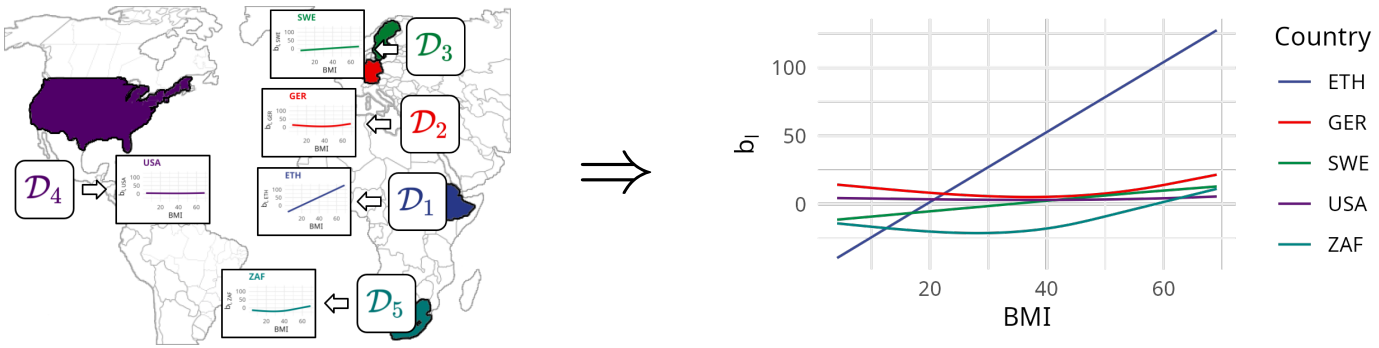
Estimation of site-specific effects

Example: Without sharing sensitive data, we want to estimate site-specific effects (e.g. for `BMI`):



Distributed estimation of site-specific effects

Site-specific effects $b_{k_\times} = b_0 \odot b_k$ are equal to a RWTP base learner with latent categorical base learner b_0 encoding the sites and the main effect b_k:

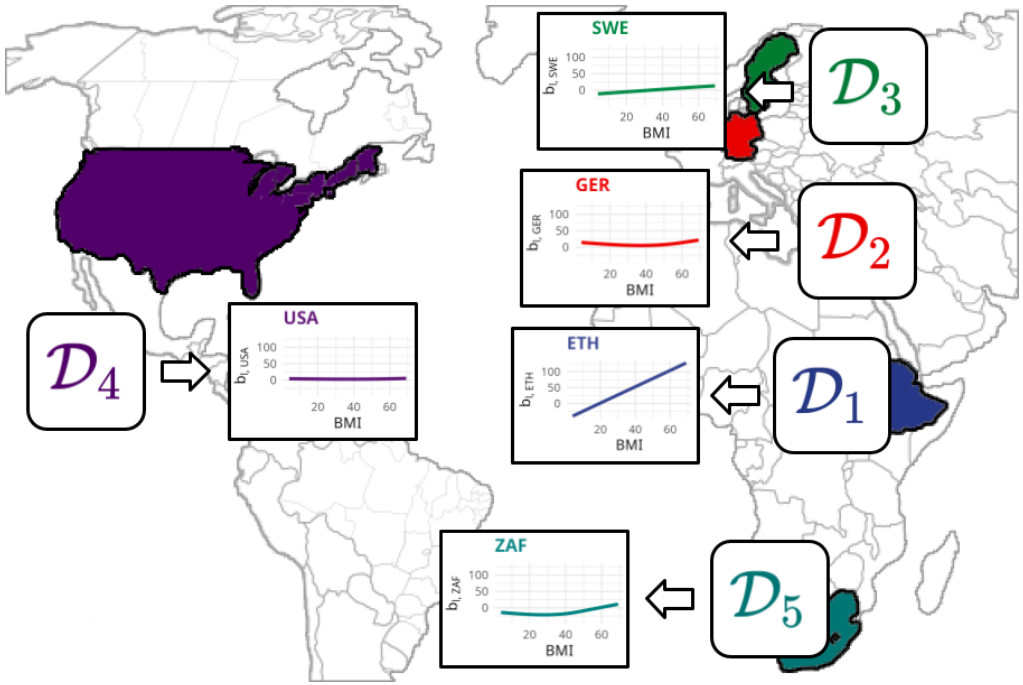


$$\hat{\boldsymbol{\theta}}_{k_\times}^{[m]} = \left(\boldsymbol{Z}_{k_\times}^\top \boldsymbol{Z}_{k_\times} + \boldsymbol{K}_{k_\times}\right)^{-1} \boldsymbol{Z}_{k_\times}^\top \boldsymbol{r}^{[m]}$$

$$= \begin{pmatrix} (\boldsymbol{Z}_{k,1}^\top \boldsymbol{Z}_{k,1} + \lambda_0 \boldsymbol{I}_{d_k} + \boldsymbol{K}_k)^{-1} \boldsymbol{Z}_{k,1}^\top \boldsymbol{r}_1^{[m]} \\ \vdots \\ (\boldsymbol{Z}_{k,S}^\top \boldsymbol{Z}_{k,S} + \lambda_0 \boldsymbol{I}_{d_k} + \boldsymbol{K}_k)^{-1} \boldsymbol{Z}_{l,S}^\top \boldsymbol{r}_S^{[m]} \end{pmatrix} = \begin{pmatrix} \hat{\boldsymbol{\theta}}_{k_\times,1}^{[m]} \\ \vdots \\ \hat{\boldsymbol{\theta}}_{k_\times,S}^{[m]} \end{pmatrix} \in \mathbb{R}^{Sd_k}$$

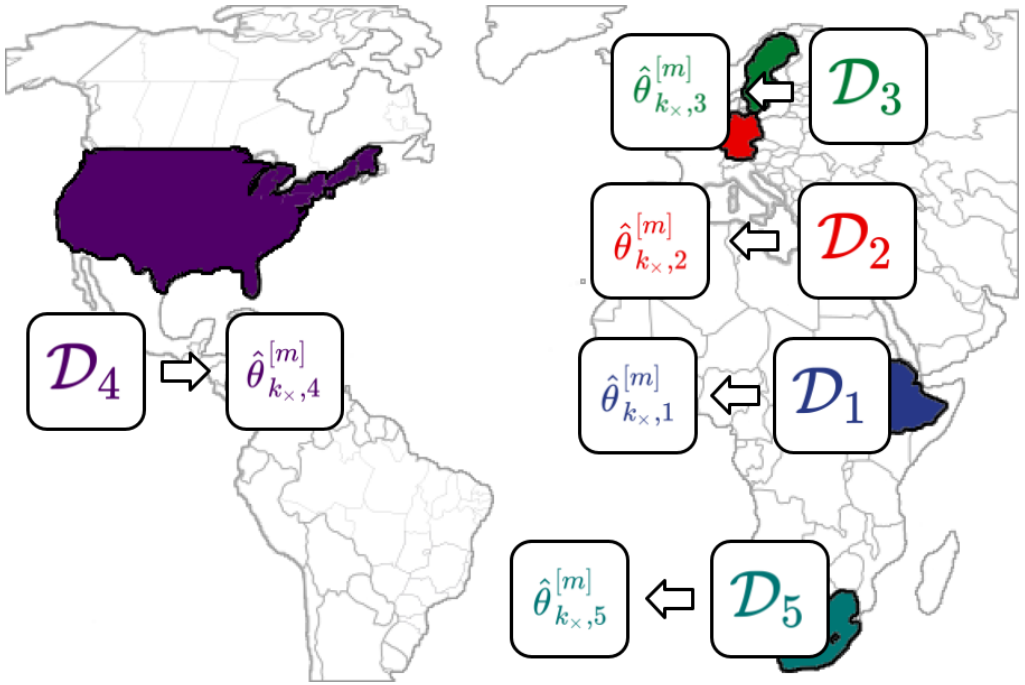
Estimation of site-specific effects

- Each site calculates and communicates the site-specific correction $\hat{\boldsymbol{\theta}}^{[m]}_{k_\times,s} = (\boldsymbol{Z}^\top_{k,s}\boldsymbol{Z}_{k,s} + \lambda_0 \boldsymbol{I}_{d_k} + \boldsymbol{K}_k)^{-1}\boldsymbol{Z}^\top_{k,s}\boldsymbol{r}^{[m]}_s$.



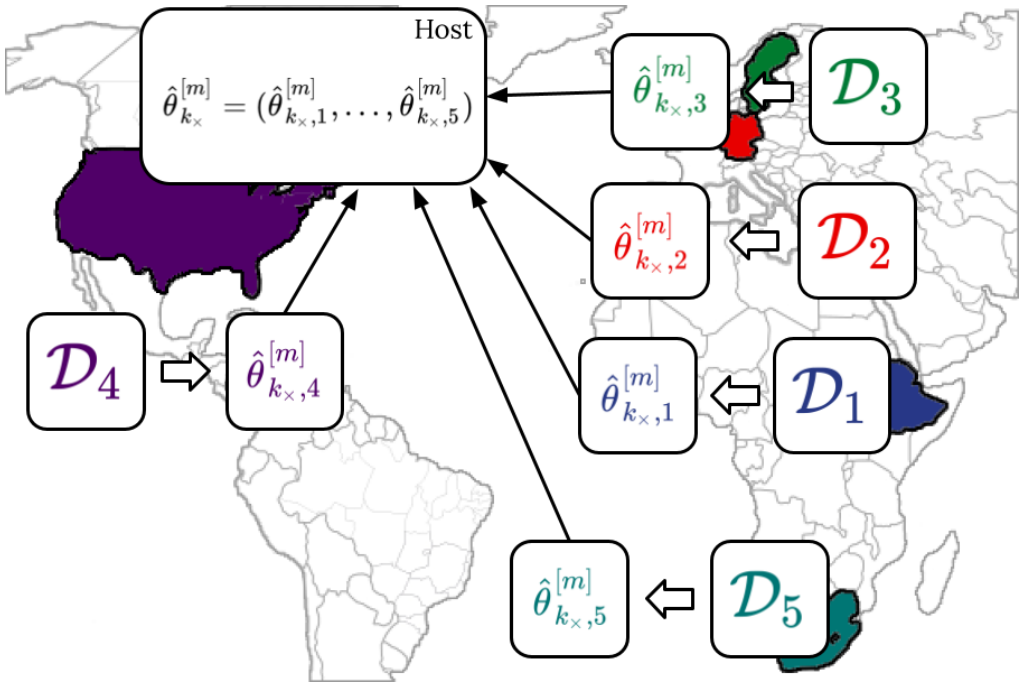
Estimation of site-specific effects

- Each site calculates and communicates the site-specific correction $\hat{\boldsymbol{\theta}}^{[m]}_{k_\times,s} = (\boldsymbol{Z}^\top_{k,s}\boldsymbol{Z}_{k,s} + \lambda_0 \boldsymbol{I}_{d_k} + \boldsymbol{K}_k)^{-1}\boldsymbol{Z}^\top_{k,s}\boldsymbol{r}^{[m]}_s$.



Estimation of site-specific effects

- The host collects all site parameters $\hat{\boldsymbol{\theta}}^{[m]}_{k_\times,1}, \ldots, \hat{\boldsymbol{\theta}}^{[m]}_{k_\times,S}$ to reconstruct the parameter vector $\hat{\boldsymbol{\theta}}^{[m]}_{k_\times}$ of the RWTP base learner $b_{k_\times}$.



Estimation of site-specific effects

Algorithm 5 Distributed CWB algorithm

Input Site data $\mathcal{D}_1, \ldots, \mathcal{D}_K$, learning rate ν, number of boosting iterations M, loss function L, base learners $b_1, \ldots, b_K$

Output Model $\hat{f} = \hat{f}^{[M]}$

procedure distCWB($\mathcal{D}_1, \ldots, \mathcal{D}_K, \nu, M, L, b_1, \ldots, b_K$)

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

  while $m \leq M$ **do**

    $r^{[m](i)} = - \left. \frac{\partial L(y^{(i)}, f(\mathrm{x}^{(i)}))}{\partial f(\mathrm{x}^{(i)})} \right|_{f=\hat{f}^{[m-1]}}, \quad \forall i \in \{1, \ldots, n\}$

    for $k \in \{1, \ldots, K\}$ **do**

      $\hat{\boldsymbol{\theta}}_k^{[m]} = \text{distFit}(\mathrm{Z}_{k,1}, \ldots, \mathrm{Z}_{k,S}, \mathrm{r}_1^{[m]}, \ldots, \mathrm{r}_S^{[m]}, \mathrm{K}_k)$

      $\hat{\boldsymbol{\theta}}_{k_\times}^{[m]} = \text{getSiteEffects}(\hat{\boldsymbol{\theta}}_{k_\times,1}^{[m]}, \ldots, \hat{\boldsymbol{\theta}}_{k_\times,S}^{[m]})$

      $\text{SSE}_k = \text{aggregate}(\text{get}(\text{SSE}_{k,1}, \ldots, \text{SSE}_{k,S}))$

      $\text{SSE}_{k_\times} = \text{aggregate}(\text{get}(\text{SSE}_{k_\times,1}, \ldots, \text{SSE}_{k_\times,S}))$

    $k^{[m]} = \arg\min_{k \in \{1, \ldots, K, 1_\times, \ldots, K_\times\}} \text{SSE}_k$

    $\hat{f}^{[m]}(\mathrm{x}) = \hat{f}^{[m-1]}(\mathrm{x}) + \nu b_{k^{[m]}}(\mathrm{x}|\hat{\boldsymbol{\theta}}_{k^{[m]}}^{[m]})$

  return $\hat{f} = \hat{f}^{[m]}$

Note: The algorithm requires further refinements like sharing of main effects from host to sites or sharing of the SSE values from the sites to the host.

Distributed CWB algorithm

Algorithm 3 Distributed CWB Algorithm.
The line prefixes `[S]` and `[H]` indicate whether the operation is conducted at the sites (`[S]`) or at the host (`[H]`).

Input Sites with site data $\mathcal{D}_k$, learning rate ν, number of boosting iterations M, loss function L, set of shared effects $\mathcal{B}$ and respective site-specific effects $\mathcal{B}_\times$
Output Prediction model $\hat{f}$

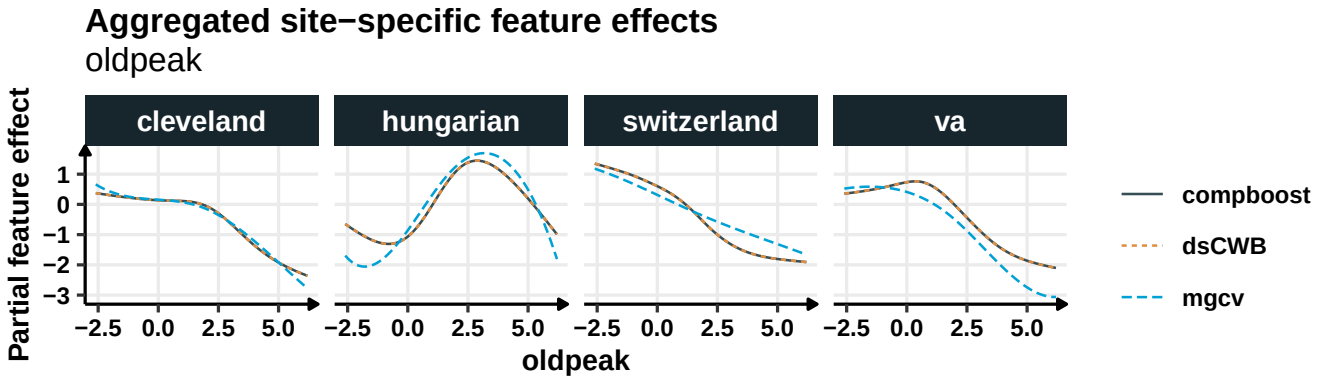
procedure distrCWB($\nu, L, \mathcal{B}, \mathcal{B}_\times$)
  Initialization:
  `[H]` Initialize shared model $\hat{f}^{[0]}_{\text{shared}}(\boldsymbol{x}) = \arg\min_{c\in\mathbb{R}} \mathcal{R}_{\text{emp}}(c)$
  `[S]` Calculate $\boldsymbol{Z}_{l,s}$ and $\boldsymbol{F}_{l,s} = \boldsymbol{Z}_{l,s}^\top \boldsymbol{Z}_{l,s}$, $\forall l \in \{1,\ldots,|\mathcal{B}|\},\ s \in \{1,\ldots,S\}$
  `[S]` Set $\hat{f}_s^{[0]} = \hat{f}^{[0]}_{\text{shared}}$
  for $m \in \{1,\ldots,M\}$ or **while** an early stopping criterion is not met **do**
    `[S]` Update pseudo residuals:
    `[S]` $\tilde{r}_s^{[m](i)} = -\nabla_f L(y^{(i)}, f(\boldsymbol{x}^{(i)}))|_{f=\hat{f}_s^{[m-1]}}, \quad \forall i \in \{1,\ldots,n_s\}$
    for $l \in \{1,\ldots,|\mathcal{B}|\}$ **do**
      `[H]` Calculate shared effect: $\hat{\boldsymbol{\theta}}_l^{[m]} = \mathsf{distFit}(\boldsymbol{Z}_{l,1},\ldots,\boldsymbol{Z}_{l,S},\boldsymbol{y}_1,\ldots,\boldsymbol{y}_S,\boldsymbol{K}_l)$
      `[H]` Communicate $\hat{\boldsymbol{\theta}}_l^{[m]}$ to the sites
      for $k \in \{1,\ldots,S\}$ **do**
        `[S]` Fit l^{th} site-specific effect: $\hat{\boldsymbol{\theta}}^{[m]}_{l_\times,s} = (\boldsymbol{F}_{l,s} + \lambda_0 \boldsymbol{I}_{d_l} + \boldsymbol{K}_l)^{-1} \boldsymbol{Z}_{l,s} \tilde{\boldsymbol{r}}_s^{[m]}$
        `[S]` Calculate the SSE for the l^{th} shared and site-specific effect:
        `[S]` $\mathsf{SSE}_{l,s} = \sum_{i=1}^{n_s} (\tilde{r}^{[m](i)} - g_l(\boldsymbol{x}^{(i)})^\top \hat{\boldsymbol{\theta}}_l^{[m]})^2$
        `[S]` $\mathsf{SSE}_{l_\times,s} = \sum_{i=1}^{n_s} (\tilde{r}_s^{[m](i)} - g_l(\boldsymbol{x}^{(i)})^\top \hat{\boldsymbol{\theta}}^{[m]}_{l_\times,s})^2$
        `[S]` Send $\mathsf{SSE}_{l,s}$ and $\mathsf{SSE}_{l_\times,s}$ to the host
      end for
      `[H]` Aggregate SSE values: $\mathsf{SSE}_l = \sum_{s=1}^S \mathsf{SSE}_{l,s}$ and $\mathsf{SSE}_{l_\times} = \sum_{s=1}^S \mathsf{SSE}_{l_\times,s}$
    end for
    `[H]` Select best base learner: $l^{[m]} = \arg\min_{l\in\{1,\ldots,|\mathcal{B}|,1_\times,\ldots,|\mathcal{B}|_\times\}} \mathsf{SSE}_l$
    if $b_{l^{[m]}}$ is a shared effect **then**
      `[H]` Update model: $\hat{f}^{[m]}_{\text{shared}}(\boldsymbol{x}) = \hat{f}^{[m-1]}_{\text{shared}}(\boldsymbol{x}) + \nu b_{l^{[m]}}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}^{[m]}_{l^{[m]}})$
      `[H]` Upload model update $\hat{\boldsymbol{\theta}}^{[m]}_{l^{[m]}}$ to the sites.
    end if
    `[S]` Update site model $\hat{f}_s^{[m]}$ via parameter updates $\hat{\boldsymbol{\theta}}_{l^{[m]}} = \hat{\boldsymbol{\theta}}_{l^{[m]}} + \nu \hat{\boldsymbol{\theta}}^{[m]}_{l^{[m]}}$
  end for
  `[S]` Communicate site-specific effects $\hat{\boldsymbol{\theta}}_{1_\times},\ldots,\hat{\boldsymbol{\theta}}_{|\mathcal{B}|_\times}$ to the host
  `[H]` Add site-specific effects to the model of shared effects $\hat{f}^{[M]}_{\text{shared}}$ to obtain the full model $\hat{f}^{[M]}$
  `[H]` **return** $\hat{f} = \hat{f}^{[M]}$
end procedure

Distributed CWB

Comparison with pooled approaches

Comparison with pooled approaches

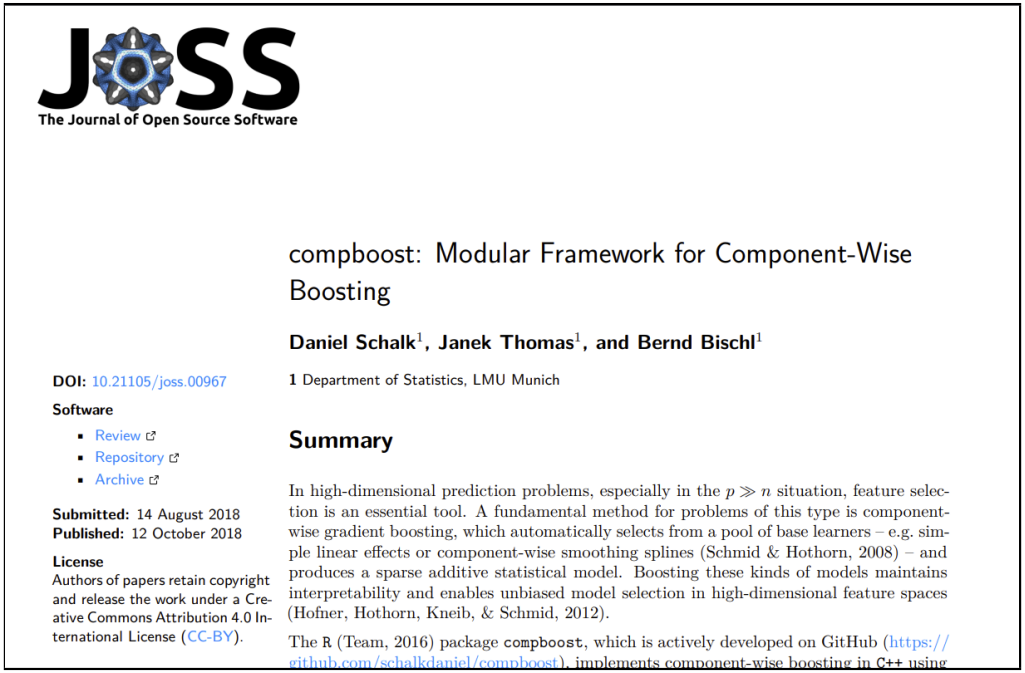
- **Reminder:** The proposed algorithm is a lossless and distributed pendant to CWB on the merged data.
- Instead of benchmarking the algorithm, we compared it with a GAMM fitted with `mgcv`.



(Figure reference: Schalk et al. (2023a))

Further contributions

Publication (Schalk et al., 2018)



JOSS
The Journal of Open Source Software

compboost: Modular Framework for Component-Wise Boosting

Daniel Schalk[1], **Janek Thomas**[1], **and Bernd Bischl**[1]

1 Department of Statistics, LMU Munich

DOI: 10.21105/joss.00967

Software

- Review
- Repository
- Archive

Submitted: 14 August 2018
Published: 12 October 2018

Summary

In high-dimensional prediction problems, especially in the $p \gg n$ situation, feature selection is an essential tool. A fundamental method for problems of this type is component-wise gradient boosting, which automatically selects from a pool of base learners – e.g. simple linear effects or component-wise smoothing splines (Schmid & Hothorn, 2008) – and produces a sparse additive statistical model. Boosting these kinds of models maintains interpretability and enables unbiased model selection in high-dimensional feature spaces (Hofner, Hothorn, Kneib, & Schmid, 2012).

The R (Team, 2016) package `compboost`, which is actively developed on GitHub (https://github.com/schalkdaniel/compboost), implements component-wise boosting in C++ using

Contributions:

- Efficient and object-oriented CWB implementation.

About

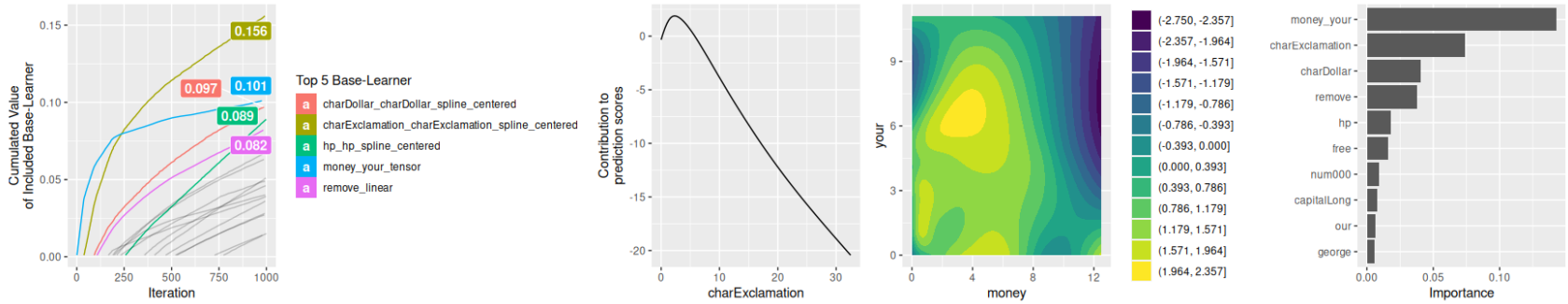
- `compboost` is an `R` package that implements CWB.
- The core is implemented in `C++` for faster runtime and exported via `Rcpp` to `R`.
- `mlr3` learners to, e.g., evaluate and tune the model.
- Parallelized model fitting with `OpenMP` and model export as `JSON`.

Demo

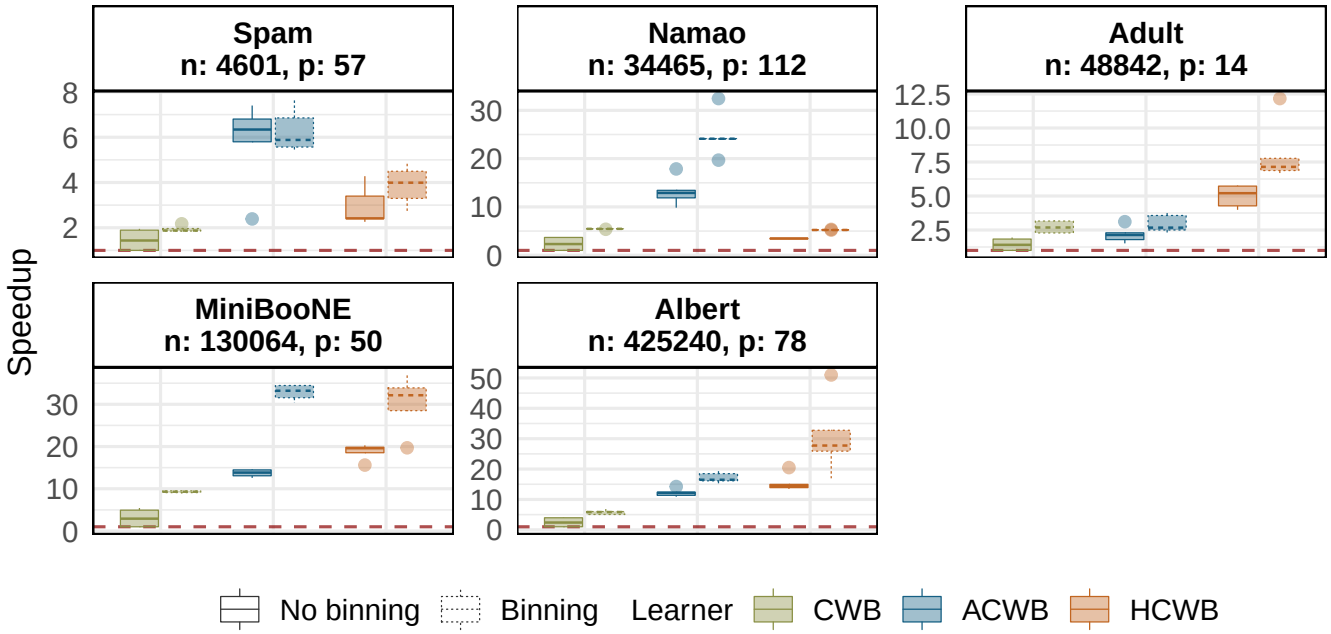
```
library(compboost)

cb = boostComponents(spam, "type", iterations = 0, df = 5)
cb$addTensor("money", "your")
cb$train(1000)

plotBaselearnerTraces(cb) | plotPEUni(cb, "charExclamation") |
  plotTensor(cb, "money_your_tensor") | plotFeatureImportance(cb, 10)
```



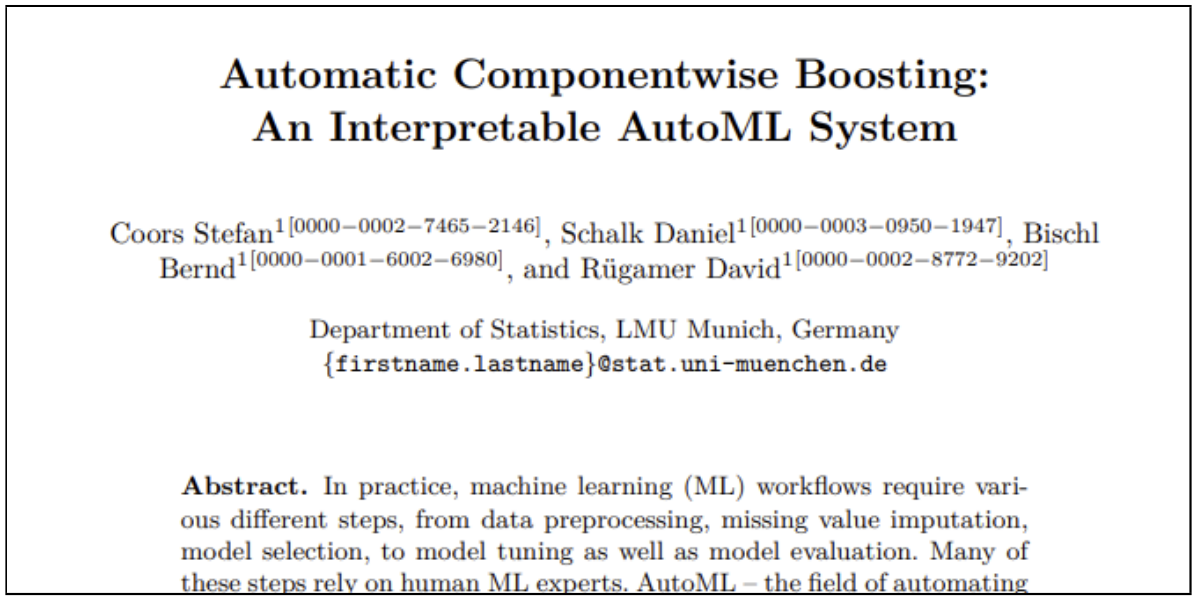
Speedup compared to mboost



(Figure reference: Schalk et al. (2022a))

- The pure implementation (CWB, green) is up to 5 times and with binning up to 10 times faster than `mboost`.
- ACWB and HCWB elevate the speedup even more.

Publication (Coors et al., 2021)



Automatic Componentwise Boosting: An Interpretable AutoML System

Coors Stefan[1 [0000−0002−7465−2146]], Schalk Daniel[1 [0000−0003−0950−1947]], Bischl Bernd[1 [0000−0001−6002−6980]], and Rügamer David[1 [0000−0002−8772−9202]]

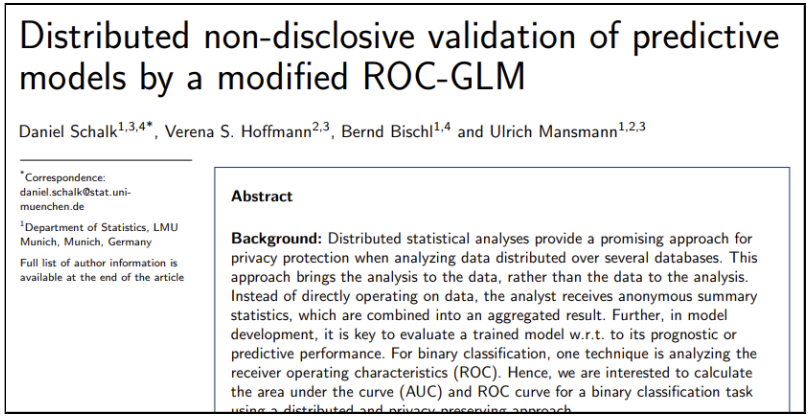
Department of Statistics, LMU Munich, Germany
{firstname.lastname}@stat.uni-muenchen.de

Abstract. In practice, machine learning (ML) workflows require various different steps, from data preprocessing, missing value imputation, model selection, to model tuning as well as model evaluation. Many of these steps rely on human ML experts. AutoML – the field of automating

Contributions:

- Interpretable automated ML (AutoML) system based on three stages with increasing complexity and CWB as the fitting engine.
- Tools to assess the required model complexity and the decision-making process.

Publications (Schalk et al., 2022b, 2023b)



Distributed non-disclosive validation of predictive models by a modified ROC-GLM

Daniel Schalk[1,3,4*], Verena S. Hoffmann[2,3], Bernd Bischl[1,4] and Ulrich Mansmann[1,2,3]

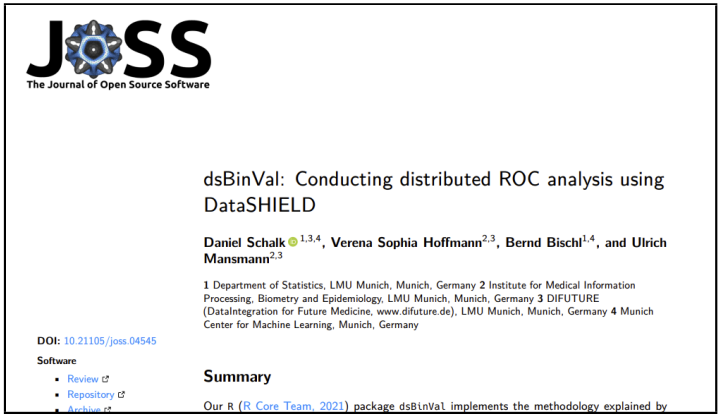
*Correspondence: daniel.schalk@stat.uni-muenchen.de

[1]Department of Statistics, LMU Munich, Munich, Germany

Full list of author information is available at the end of the article

Abstract

Background: Distributed statistical analyses provide a promising approach for privacy protection when analyzing data distributed over several databases. This approach brings the analysis to the data, rather than the data to the analysis. Instead of directly operating on data, the analyst receives anonymous summary statistics, which are combined into an aggregated result. Further, in model development, it is key to evaluate a trained model w.r.t. to its prognostic or predictive performance. For binary classification, one technique is analyzing the receiver operating characteristics (ROC). Hence, we are interested to calculate the area under the curve (AUC) and ROC curve for a binary classification task using a distributed and privacy-preserving approach



JOSS The Journal of Open Source Software

dsBinVal: Conducting distributed ROC analysis using DataSHIELD

Daniel Schalk[1,3,4], Verena Sophia Hoffmann[2,3], Bernd Bischl[1,4], and Ulrich Mansmann[2,3]

1 Department of Statistics, LMU Munich, Munich, Germany 2 Institute for Medical Information Processing, Biometry and Epidemiology, LMU Munich, Munich, Germany 3 DIFUTURE (DataIntegration for Future Medicine, www.difuture.de), LMU Munich, Munich, Germany 4 Munich Center for Machine Learning, Munich, Germany

DOI: 10.21105/joss.04545

Software

- Review
- Repository
- Archive

Summary

Our R (R Core Team, 2021) package dsBinVal implements the methodology explained by

Contributions:

- Privacy-preserving and distributed evaluation based on the AUC.
- Implementation in `DataSHIELD` (Gaye et al., 2014) to validate binary classification models.

Conclusion and outlook

Conclusion

The presented adaptions to CWB improve the algorithm in several aspects:

- **Efficiency** (Schalk et al., 2022a)
  - ⇒ CWB for big data (Software: `compboost` (Schalk et al., 2018)).
- **Distributed computing** (Schalk et al., 2023a)
  - ⇒ Fit CWB to (horizontally) distributed data sets by preserving privacy (Software: `dsCWB`).
- **Automation** (Coors et al., 2021)
  - ⇒ Easy access to CWB also for non-experts by a multi-stage approach (Software: `Autocompboost`).

Outlook

Efficiency:

- `compboost`: Better binning support and array arithmetic to accelerate the fitting for RWTP base learners.

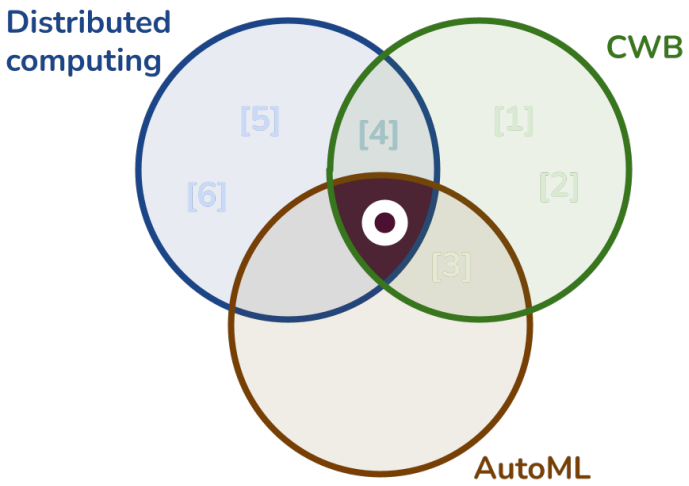
Distributed computing:

- distCWB: Account for vertically and horizontally distributed data and reduce communication costs.
- A general framework for distributed model evaluation.

Automation:

- Focus on the third stage:
  - Investigate how problematic the switch in the model class is.
  - This relates to detecting the relevant interactions (based on a random forest) and the base learner (tress) used in the third stage.

Outlook



⇒ Framework that combines all the presented aspects:

- Distributed `Autocompboost` that fits a privacy-preserving CWB variant to horizontally and vertically distributed data sets.
- Provide practitioners with insights about the required complexity, feature importance, main effects and site-specific corrections, as well as transparent decision-making.

Thank you for your attention!

Backup

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318.

Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., et al. (2020). Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115.

Audet, C. and Hare, W. (2017). *Derivative-free and blackbox optimization*, volume 2. Springer.

Barrett, R., Berry, M., Chan, T. F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., and Van der Vorst, H. (1994). *Templates for the solution of linear systems: building blocks for iterative methods*. SIAM.

Bates, D., Maechler, M., and Jagan, M. (2022). *Matrix: Sparse and Dense Matrix Classes and Methods*. R package version 1.5-1.

Bekkerman, R., Bilenko, M., and Langford, J. (2011). *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press.

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

References ii

Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., Thomas, J., Ullmann, T., Becker, M., Boulesteix, A.-L., et al. (2021). Hyperparameter optimization: Foundations, algorithms, best practices and open challenges. *arXiv preprint arXiv:2107.05847*.

Bischl, B., Mersmann, O., Trautmann, H., and Weihs, C. (2012). Resampling methods for meta-model validation with recommendations for evolutionary computation. *Evolutionary computation*, 20:249–75.

Bost, R., Popa, R. A., Tu, S., and Goldwasser, S. (2014). Machine learning classification over encrypted data. Cryptology ePrint Archive, Paper 2014/331. `https://eprint.iacr.org/2014/331`.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Brier, G. W. et al. (1950). Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3.

Brockhaus, S., Rügamer, D., and Greven, S. (2020). Boosting functional regression models with fdboost. *Journal of Statistical Software*, 94(10):1–50.

Browne, M. W. (2000). Cross-validation methods. *Journal of Mathematical Psychology*, 44(1):108–132.

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.

Buluc, A. and Gilbert, J. R. (2008). Challenges and advances in parallel sparse matrix-matrix multiplication. In *2008 37th International Conference on Parallel Processing*, pages 503–510.

References iii

Casalicchio, G. (2019). *On benchmark experiments and visualization methods for the evaluation and interpretation of machine learning models*. PhD dissertation, LMU Munich.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., et al. (2015). Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4.

Chen, Y.-R., Rezapour, A., and Tzeng, W.-G. (2018). Privacy-preserving ridge regression on distributed data. *Information Sciences*, 451:34–49.

Choi, J., Walker, D. W., and Dongarra, J. J. (1994). Pumma: Parallel universal matrix multiplication algorithms on distributed memory concurrent computers. *Concurrency: Practice and Experience*, 6(7):543–570.

Coors, S., Schalk, D., Bischl, B., and Rügamer, D. (2021). Automatic componentwise boosting: An interpretable automl system. *ECML-PKDD Workshop on Automating Data Science*.

Cunha, M., Mendes, R., and Vilela, J. P. (2021). A survey of privacy-preserving mechanisms for heterogeneous data types. *Computer Science Review*, 41:100403.

Dagum, L. and Menon, R. (1998). Openmp: an industry standard api for shared-memory programming. *Computational Science & Engineering, IEEE*, 5(1):46–55.

Davis, T. A. (2006). *Direct methods for sparse linear systems*. SIAM.

DeLong, E. R., DeLong, D. M., and Clarke-Pearson, D. L. (1988). Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. *Biometrics*, pages 837–845.

References iv

Drozdal, J., Weisz, J., Wang, D., Dass, G., Yao, B., Zhao, C., Muller, M., Ju, L., and Su, H. (2020). Trust in automl: Exploring information needs for establishing trust in automated machine learning systems. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, IUI '20, page 297–307, New York, NY, USA. Association for Computing Machinery.

Duff, I. S., Grimes, R. G., and Lewis, J. G. (1989). Sparse matrix test problems. *ACM Transactions on Mathematical Software (TOMS)*, 15(1):1–14.

Dwork, C. (2006). Differential privacy. In *International Colloquium on Automata, Languages, and Programming*, pages 1–12. Springer.

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. (2006a). Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–503. Springer.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006b). Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer.

Dwork, C., Roth, A., et al. (2014). The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407.

Eilers, P. H. and Marx, B. D. (1996). Flexible smoothing with B-splines and penalties. *Statistical science*, pages 89–102.

Fang, H. and Qian, Q. (2021). Privacy preserving machine learning with homomorphic encryption and federated learning. *Future Internet*, 13(4).

Fawcett, T. (2006). An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874.

References v

Feurer, M. and Hutter, F. (2019). Hyperparameter optimization. In *Automated machine learning*, pages 3–33. Springer, Cham.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F. (2015). Efficient and robust automated machine learning. *Advances in neural information processing systems*, 28.

Flach, P. (2012). *Machine learning: the art and science of algorithms that make sense of data*. Cambridge university press.

Freitas, A. A. (2019). Automated machine learning for studying the trade-off between predictive accuracy and interpretability. In Holzinger, A., Kieseberg, P., Tjoa, A. M., and Weippl, E., editors, *Machine Learning and Knowledge Extraction*, pages 48–66, Cham. Springer International Publishing.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

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

Gambs, S., Kégl, B., and Aïmeur, E. (2007). Privacy-preserving boosting. *Data Mining and Knowledge Discovery*, 14(1):131–170.

Gaye, A., Marcon, Y., Isaeva, J., LaFlamme, P., Turner, A., Jones, E. M., Minion, J., Boyd, A. W., Newby, C. J., Nuotio, M.-L., et al. (2014). Datashield: taking the analysis to the data, not the data to the analysis. *International journal of epidemiology*, 43(6):1929–1944.

References vi

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178.

Gong, M., Xie, Y., Pan, K., Feng, K., and Qin, A. (2020). A survey on differentially private machine learning [review article]. *IEEE Computational Intelligence Magazine*, 15(2):49–64.

Gordon, D. F. and Desjardins, M. (1995). Evaluation and selection of biases in machine learning. *Machine learning*, 20(1):5–22.

Hastie, T. J. (2017). Generalized additive models. In *Statistical models in S*, pages 249–307. Routledge.

Hofner, B., Hothorn, T., Kneib, T., and Schmid, M. (2011). A framework for unbiased model selection based on boosting. *Journal of Computational and Graphical Statistics*, 20(4):956–971.

Hofner, B., Mayr, A., and Schmid, M. (2016). gamboostLSS: An R package for model building and variable selection in the GAMLSS framework. *Journal of Statistical Software*, 74(1).

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

Hothorn, T., Bühlmann, P., Kneib, T., Schmid, M., and Hofner, B. (2020). *mboost: Model-based boosting*. R package version 2.9-7.

Hutter, F., Kotthoff, L., and Vanschoren, J. (2019). *Automated machine learning: methods, systems, challenges*. Springer Nature.

Jayaraman, B. and Evans, D. (2019). Evaluating differentially private machine learning in practice. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1895–1912.

References vii

John, G. H. (1995). Robust decision trees: Removing outliers from databases. In *KDD*, volume 95, pages 174–179.

Karr, A. F., Lin, X., Sanil, A. P., and Reiter, J. P. (2005). Secure regression on distributed databases. *Journal of Computational and Graphical Statistics*, 14(2):263–279.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F., and Leyton-Brown, K. (2017). Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka. *Journal of Machine Learning Research*, 18(25):1–5.

Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F., and Leyton-Brown, K. (2019). Auto-weka: Automatic model selection and hyperparameter optimization in weka. In *Automated machine learning*, pages 81–95. Springer, Cham.

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

Lazarevic, A. and Obradovic, Z. (2001). The distributed boosting algorithm. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 311–316.

Li, J., Kuang, X., Lin, S., Ma, X., and Tang, Y. (2020). Privacy preservation for machine learning training and classification based on homomorphic encryption schemes. *Information Sciences*, 526:166–179.

References viii

Li, Y., Jiang, X., Wang, S., Xiong, H., and Ohno-Machado, L. (2016). Vertical grid logistic regression (vertigo). *Journal of the American Medical Informatics Association*, 23(3):570–579.

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.

Liew, B. X., Rügamer, D., Abichandani, D., and De Nunzio, A. M. (2020a). Classifying individuals with and without patellofemoral pain syndrome using ground force profiles – Development of a method using functional data boosting. *Gait & Posture*, 80:90–95.

Liew, B. X., Rügamer, D., Stocker, A., and De Nunzio, A. M. (2020b). Classifying neck pain status using scalar and functional biomechanical variables – Development of a method using functional data boosting. *Gait & posture*, 76:146–150.

Liu, W. and Vinter, B. (2014). An efficient gpu general sparse matrix-matrix multiplication for irregular data. In *2014 IEEE 28th International Parallel and Distributed Processing Symposium*, pages 370–381.

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.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

References ix

Luo, C., Islam, M., Sheils, N. E., Buresh, J., Reps, J., Schuemie, M. J., Ryan, P. B., Edmondson, M., Duan, R., Tong, J., et al. (2022). Dlmm as a lossless one-shot algorithm for collaborative multi-site distributed linear mixed models. *Nature Communications*, 13(1):1–10.

Machanavajjhala, A., Kifer, D., Gehrke, J., and Venkitasubramaniam, M. (2007). l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3–es.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. y. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. In Singh, A. and Zhu, J., editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR.

Mohassel, P. and Zhang, Y. (2017). Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE symposium on security and privacy (SP)*, pages 19–38. IEEE.

Molnar, C. (2020). *Interpretable machine learning*. Lulu. com.

Nesterov, Y. (1983). A method for solving the convex programming problem with convergence rate $O(1/k^2)$.

Nori, H., Jenkins, S., Koch, P., and Caruana, R. (2019). Interpretml: A unified framework for machine learning interpretability. *arXiv preprint arXiv:1909.09223*.

Pepe, M. S. (2000). An interpretation for the roc curve and inference using glm procedures. *Biometrics*, 56(2):352–359.

Pepe, M. S. (2003). The statistical evaluation of medical tests for classification and prediction. *Journal of the American Statistical Association*.

References x

Pfisterer, F. (2022). *Democratizing Machine Learning – Contributions in AutoML and Fairness*. PhD thesis, LMU Munich.

Pfisterer, F., Thomas, J., and Bischl, B. (2019). Towards human centered automl. *arXiv preprint arXiv:1911.02391*.

Prasser, F., Kohlbacher, O., Mansmann, U., Bauer, B., and Kuhn, K. A. (2018). Data integration for future medicine (difuture). *Methods Inf Med*, 57(S01):e57–e65.

R Core Team (2022). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215.

Rügamer, D., Brockhaus, S., Gentsch, K., Scherer, K., and Greven, S. (2018). Boosting factor-specific functional historical models for the detection of synchronization in bioelectrical signals. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 67(3):621–642.

References xi

Saintigny, P., Zhang, L., Fan, Y.-H., El-Naggar, A. K., Papadimitrakopoulou, V. A., Feng, L., Lee, J. J., Kim, E. S., Hong, W. K., and Mao, L. (2011). Gene expression profiling predicts the development of oral cancer. *Cancer Prevention Research*, 4(2):218–229.

Samarati, P. and Sweeney, L. (1998). Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression.

Sanderson, C. and Curtin, R. (2016). Armadillo: a template-based c++ library for linear algebra. *Journal of Open Source Software*, 1(2):26.

Sanderson, C. and Curtin, R. (2018). A user-friendly hybrid sparse matrix class in c++. In *International Congress on Mathematical Software*, pages 422–430. Springer.

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

Schalk, D., Bischl, B., and Rügamer, D. (2023a). Privacy-preserving and lossless distributed estimation of high-dimensional generalized additive mixed models. *arXiv preprint arXiv:2210.07723*.

Schalk, D., Hoffmann, V. S., Bischl, B., and Mansmann, U. (2022b). Distributed non-disclosive validation of predictive models by a modified roc-glm. *arXiv preprint arXiv:2203.10828*.

Schalk, D., Hoffmann, V. S., Bischl, B., and Mansmann, U. (2023b). dsBinVal: Conducting distributed roc analysis using datashield. *Journal of Open Source Software*, 8(82):4545.

References xii

Schalk, D., Thomas, J., and Bischl, B. (2018). compboost: Modular framework for component-wise boosting. *Journal of Open Source Software*, 3(30):967.

Schmid, M. and Hothorn, T. (2008). Boosting additive models using component-wise p-splines. *Computational Statistics & Data Analysis*, 53(2):298–311.

Shahnaz, R., Usman, A., and Chughtai, I. R. (2005). Review of storage techniques for sparse matrices. In *2005 Pakistan Section Multitopic Conference*, pages 1–7.

Stehman, S. V. (1997). Selecting and interpreting measures of thematic classification accuracy. *Remote sensing of Environment*, 62(1):77–89.

Sun, X., Zhang, P., Liu, J. K., Yu, J., and Xie, W. (2020). Private machine learning classification based on fully homomorphic encryption. *IEEE Transactions on Emerging Topics in Computing*, 8(2):352–364.

Sweeney, L. (2002). k-anonymity: A model for protecting privacy. *International journal of uncertainty, fuzziness and knowledge-based systems*, 10(05):557–570.

Thomas, J., Coors, S., and Bischl, B. (2018). Automatic gradient boosting. *ICML AutoML Workshop*.

Thomas, J., Hepp, T., Mayr, A., and Bischl, B. (2017). Probing for sparse and fast variable selection with model-based boosting. *Computational and mathematical methods in medicine*, 2017.

Thornton, C., Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2013). Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855.

References xiii

Tutz, G. and Gertheiss, J. (2016). Regularized regression for categorical data. *Statistical Modelling*, 16(3):161–200.

Van Buuren, S. (2018). *Flexible imputation of missing data*. CRC press.

Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Verbelen, T., and Rellermeyer, J. S. (2020). A survey on distributed machine learning. *Acm computing surveys (csur)*, 53(2):1–33.

Vuk, M. and Curk, T. (2006). Roc curve, lift chart and calibration plot. *Metodoloski zvezki*, 3(1):89.

Wang, Q. and Kurz, D. (2022). Reconstructing training data from diverse ml models by ensemble inversion. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2909–2917.

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

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.

Xanthopoulos, I., Tsamardinos, I., Christophides, V., Simon, E., and Salinger, A. (2020). Putting the human back in the automl loop. In *EDBT/ICDT Workshops*.

Yan, Z., Zachrison, K. S., Schwamm, L. H., Estrada, J. J., and Duan, R. (2022). Fed-glmm: A privacy-preserving and computation-efficient federated algorithm for generalized linear mixed models to analyze correlated electronic health records data. *medRxiv*.

Zhu, R., Jiang, C., Wang, X., Wang, S., Zheng, H., and Tang, H. (2020). Privacy-preserving construction of generalized linear mixed model for biomedical computation. *Bioinformatics*, 36(Supplement_1):i128–i135.

Backup

Terminology

- p-dimensional covariate or feature vector $\mathbf{x} = (x_1, \ldots, x_p) \in \mathcal{X} = \mathcal{X}_1 \times \cdots \times$ and target variable $y \in \mathcal{Y}$.
- Data set $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)}) \mid i = 1, \ldots, n\}$ with $(\mathbf{x}^{(i)}, y^{(i)})$ sampled from an unknown probability distribution $\mathbb{P}_{xy}$.
- True underlying relationship $f : \mathcal{X}^p \to \mathbb{R}$, $\mathbf{x} \mapsto f(\mathbf{x})$.
- Goal of Machine Learning (ML) is to estimate a model $\hat{f} = \arg\min_f \mathcal{R}_{\text{emp}}(f|\mathcal{D})$ with
  - Empirical risk $\mathcal{R}_{\text{emp}}(f|\mathcal{D}) = n^{-1} \sum_{(\mathbf{x},y)\in\mathcal{D}} L(y, \hat{f}(\mathbf{x}))$ and
  - Loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$, $(y, \hat{y}) \mapsto L(y, \hat{y})$.
- The inducer $\mathcal{I} : \mathbb{D} \times \Lambda \to \mathcal{F}$, $(\mathcal{D}, \boldsymbol{\lambda}) \mapsto \hat{f} = \mathcal{I}_{\boldsymbol{\lambda}}(\mathcal{D})$ gets a data set $\mathcal{D} \in \mathbb{D}$ with hyperparameters (HPs) $\boldsymbol{\lambda} \in \Lambda$.

Gradient boosting

- Gradient boosting (GB) aims to estimate f based on assembling weak base learners $b : \mathcal{X} \to \mathcal{Y}, \mathbf{x} \mapsto b(\mathbf{x}|\boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$.
- The model estimate $\hat{f}$ is fitted by conducting functional gradient descent $\hat{f}^{[m-1]} = \hat{f}^{[m]} + \nu \hat{b}^{[m]}$ for M steps. The estimated model is then $\hat{f} = \hat{f}^{[M]}$.
- To obtain the model update $\hat{b}^{[m]}$ in iteration m, the weak base learner b is fit to pseudo residuals $\boldsymbol{r}^{[m]}$ by minimizing the SSE: $\hat{\boldsymbol{\theta}}^{[m]} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} (r^{[m](i)} - b(\mathbf{x}^{(i)}|\boldsymbol{\theta}))^2$
- The pseudo residuals $r^{[m](i)} = - \left. \frac{\partial L(y^{(i)}, f(\mathbf{x}^{(i)}))}{\partial f(\mathbf{x}^{(i)})} \right|_{f = \hat{f}^{[m-1]}}$, $i \in \{1, \ldots, n\}$, ($\boldsymbol{r}^{[m]}$ is the vector of pseudo residuals) contain the information in which direction to move $\hat{f}^{[m]}$ for a better fit to the training data $\mathcal{D}$.
- The fitting is initialized with $\hat{f}^{[0]}(\mathbf{x}) = \arg\min_{c \in \mathcal{Y}} \mathcal{R}_{\text{emp}}(c|\mathcal{D})$ and repeated M times or until an early stopping criterion is met.

Gradient boosting – Algorithm

Algorithm 6 GB algorithm

Input Train data $\mathcal{D}$, number of boosting iterations M, loss function L, base learner b
Output Model $\hat{f} = \hat{f}^{[M]}$

procedure GB($\mathcal{D}, M, L, b$)
  Initialize: $f_0 = \hat{f}^{[0]}(\mathbf{x}) = \textbf{arg min}_{c \in \mathcal{Y}} \mathcal{R}_{\text{emp}}(c|\mathcal{D})$
  while $m \leq M$ **do**
    $r^{[m](i)} = - \left. \frac{\partial L(y^{(i)}, f(\mathbf{x}^{(i)}))}{\partial f(\mathbf{x}^{(i)})} \right|_{f = \hat{f}^{[m-1]}}, \quad \forall i \in \{1, \ldots, n\}$
    $\hat{\boldsymbol{\theta}}^{[m]} = \textbf{arg min}_{\boldsymbol{\theta}} \sum_{i=1}^{n} (r^{[m](i)} - b(\mathbf{x}^{(i)}|\boldsymbol{\theta}))^2$
    $\nu_m = \textbf{arg min}_{\nu \in \mathbb{R}} \sum_{i=1}^{n} L(r^{[m](i)}, \hat{f}^{[m]} + \nu \hat{b}^{[m]}(\mathbf{x}^{(i)}|\hat{\boldsymbol{\theta}}^{[m]}))$
    $\hat{f}^{[m]}(\mathbf{x}) = \hat{f}^{[m-1]}(\mathbf{x}) + \nu_m \hat{b}^{[m]}(\mathbf{x}|\hat{\boldsymbol{\theta}}^{[m]})$
  return $\hat{f} = \hat{f}^{[M]}$

A common choice for the base learner in GB is, e.g., to use trees (Friedman, 2001). Based on the base learner, further adaptions to the algorithm are made to, e.g., increase speed or predictive power (Chen et al., 2015).

Basics

- Compared to GB, CWB can choose from a set of K base learners $b \in \{b_1, \ldots, b_K\}$.
- Often, $b_1, \ldots, b_K$ are chosen to be (interpretable) statistical models and hence f corresponds to a generalized additive model (GAM; Hastie, 2017):

$$f(\mathbf{x}) = f_0 + \sum_{k=1}^{K} b_k(\mathbf{x}|\boldsymbol{\theta}), \quad \text{intercept } f_0$$

- Advantages of CWB:
  - Feasible to get fit in high-dimensional feature spaces ($p \gg n$).
  - An inherent (unbiased) feature selection.
  - Interpretable/explainable partial feature effects (depending on the choice of base learners).

Base learner

- From now on, each base learner b_k is defined by a basis transformation $g_k : \mathcal{X} \to \mathbb{R}^{d_k}$ with $g_k(\mathbf{x}) = (g_{k,1}(\mathbf{x}), \ldots, g_{k,d_k}(\mathbf{x}))^\top$.
- The base learners are also restricted to be linear in the parameters: $b_k(\mathbf{x}|\boldsymbol{\theta}_k) = g_k(\mathbf{x})^\top \boldsymbol{\theta}_k$
- Due to the linearity, the sum of two base learners $b_k(\mathbf{x}|\boldsymbol{\theta}_l) + b_k(\mathbf{x}|\boldsymbol{\theta}_m)$ equals $b_k(\mathbf{x}|\boldsymbol{\theta}_l + \boldsymbol{\theta}_m)$.
- For n data points $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$, each base learner defines a design matrix $Z_k = (g_k(\mathbf{x}^{(1)})^\top, \ldots, g_k(\mathbf{x}^{(n)})^\top)^\top \in \mathbb{R}^{n \times d_k}$.
- Based on the linearity and the design matrix, each base learner can be fitted by calculating the least squares estimator $\hat{\boldsymbol{\theta}}_k = (Z_k^\top Z_k) Z_k^\top \mathbf{y}$.
- Further, a base learner is allowed to include a penalization defined by a matrix K_k which extends the estimation to $\hat{\boldsymbol{\theta}}_k = (Z_k^\top Z_k + K_k) Z_k^\top \mathbf{y}$.

ACWB

Accelerated gradient boosting machine

- Applying Nesterov's momentum in GB was already proposed by Biau et al. (2019) and refined in an algorithm called Accelerated Gradient Boosting Machine (AGBM) by Lu et al. (2020):

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

$$
\Rightarrow \; f^{[m+1]} = f^{[m]} + \nu b^{[m]} + \theta_m (h^{[m]} - f^{[m]})
$$

- Momentum in the direction of $h^{[m]} - f^{[m]}$.
- In each iteration, two base learners $b^{[m]}$ and $b_{\text{cor}}^{[m]}$.
- ($\theta_m = 2/(m+2),\ m = 0, \ldots, M-1$, momentum $\gamma \in (0, 1]$)

Accelerated gradient boosting machine

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

- A second base learner $b_{\text{cor}}^{[m]}$ is fitted to "error-corrected pseudo residuals" $\boldsymbol{c}^{[m]}$:

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

  This accelerates the fitting into the direction of $\boldsymbol{c}^{[m]}$.

Base learners in AGBM

- $b^{[m]}$ is fitted to pseudo residuals $\boldsymbol{r}^{[m]}$ w.r.t. $\hat{g}^{[m-1]}$ instead of $\hat{f}^{[m]}$.
- AGBM introduces a second base learner $b_{\text{cor}}^{[m]}$ that is fitted to error-corrected pseudo residuals:

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

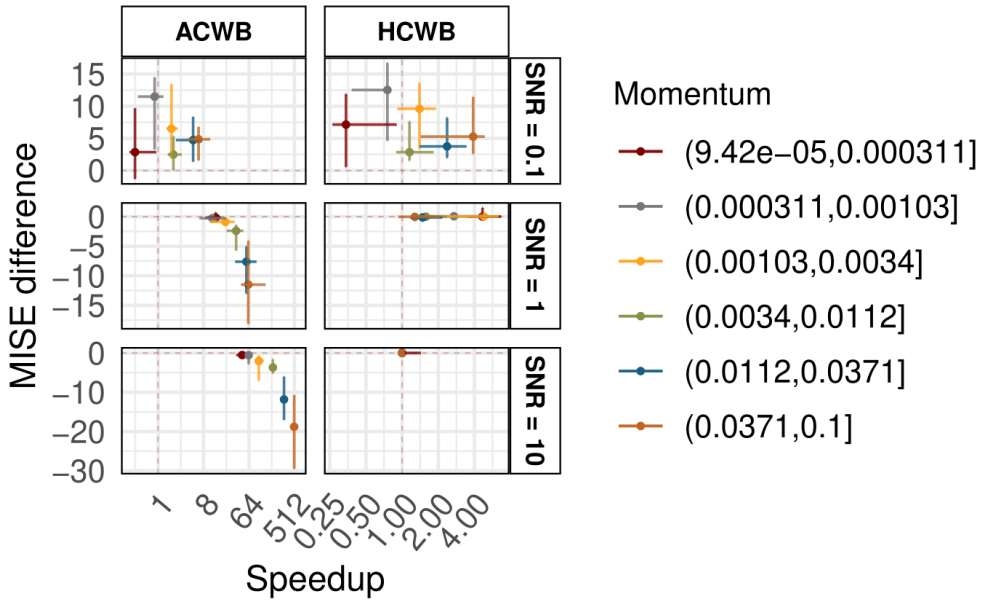
with $i = 1, \ldots, n$, if $m > 1$ and $\boldsymbol{c}^{[m]} = \boldsymbol{r}^{[m]}$ if $m = 0$.

$\Rightarrow$ Each iteration adds but two base learners $b^{[m]}$ and $b_{\text{cor}}^{[m]}$:

- $b_{\text{cor}}^{[m]}$ defines the momentum sequence to accelerate the fitting into the direction of the error-corrected pseudo residuals
- Computing a second base learner also means two double the runtime for the same number of iterations.

Simulation study to assess the estimation quality

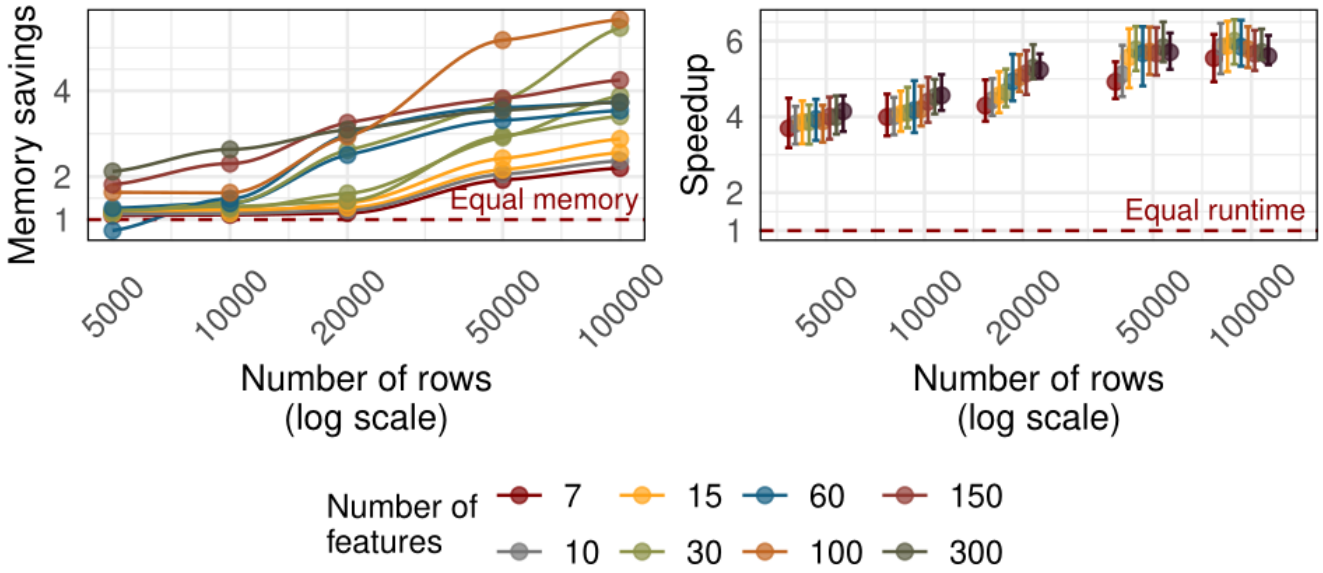
To assess the estimation quality, we measured the mean integrated squared error (MISE) between a true simulated and the estimated effects from CWB, ACWB, and HCWB:



(Figure reference: Schalk et al. (2022a)) The left column shows MISE(*CWB*) − MISE(*ACWB*) and the right column MISE(*CWB*) − MISE(*HCWB*)

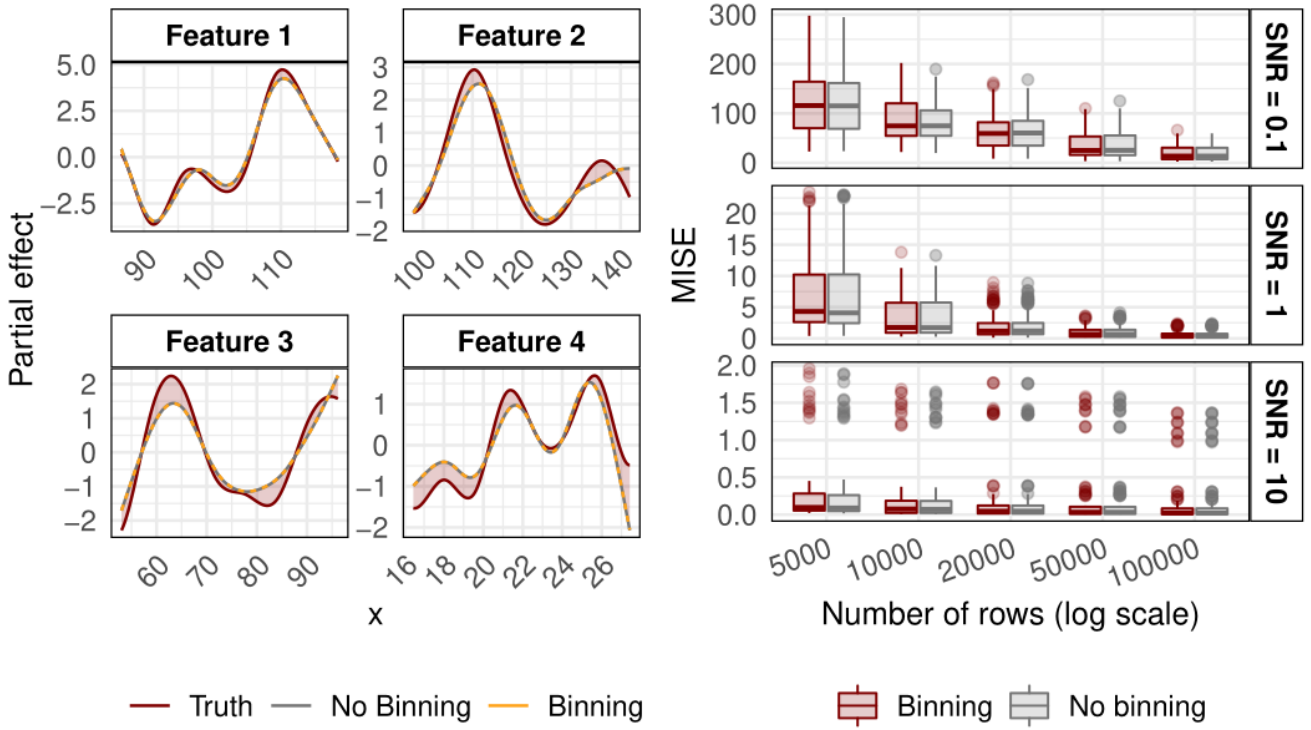
Binning

Efficiency of binning



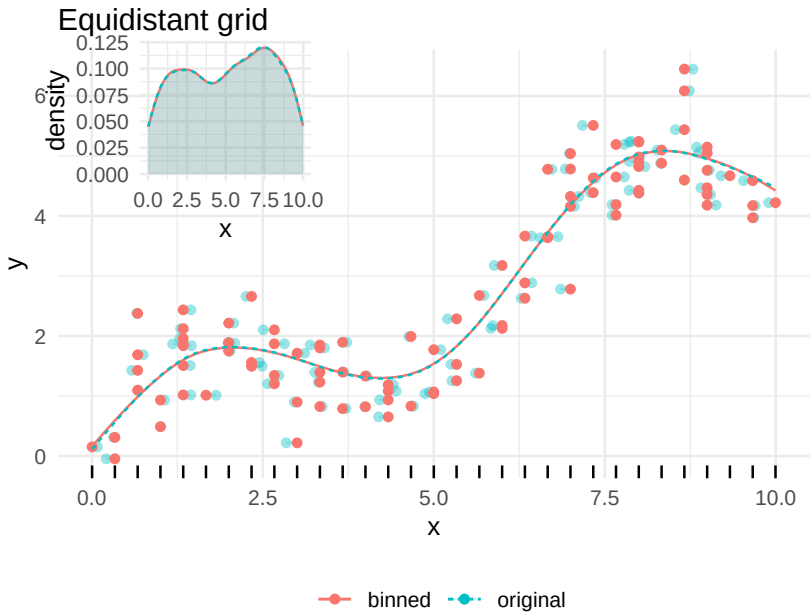
(Figure reference: Schalk et al. (2022a))

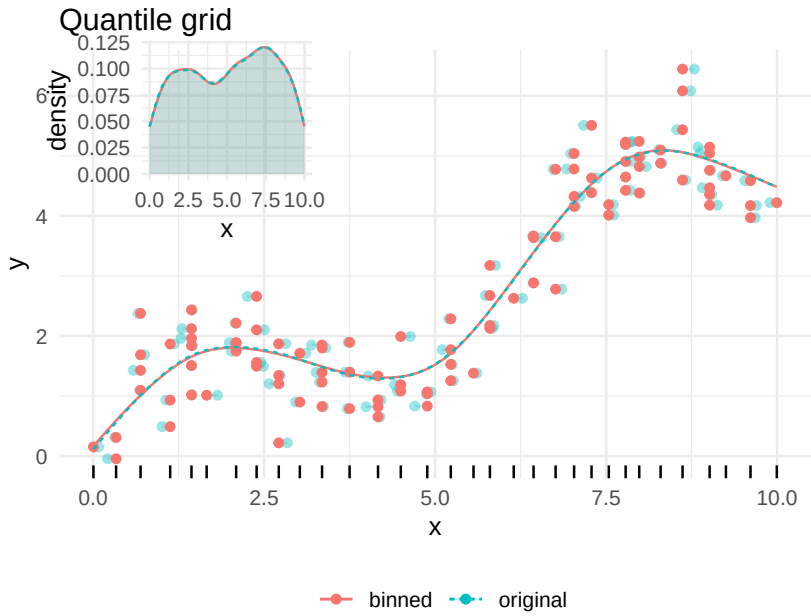
MISE for binning



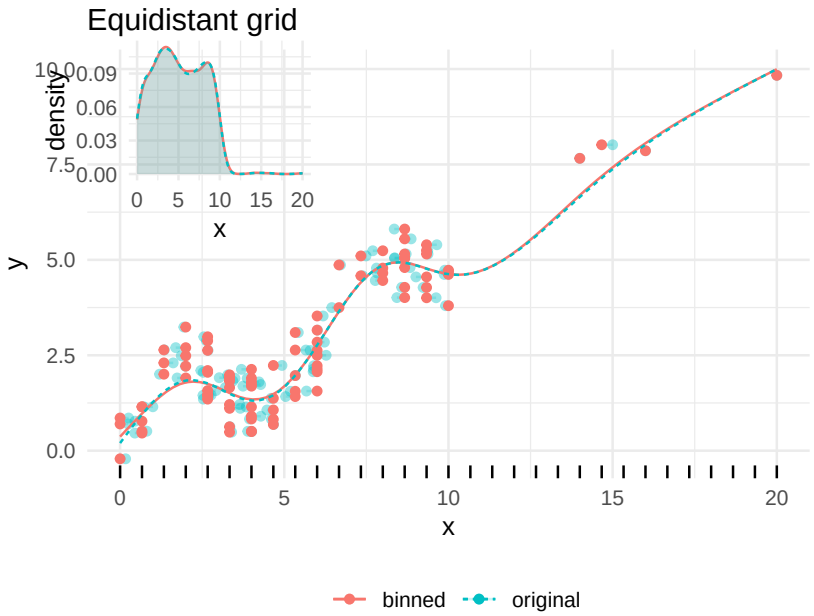
(Figure reference: Schalk et al. (2022a))

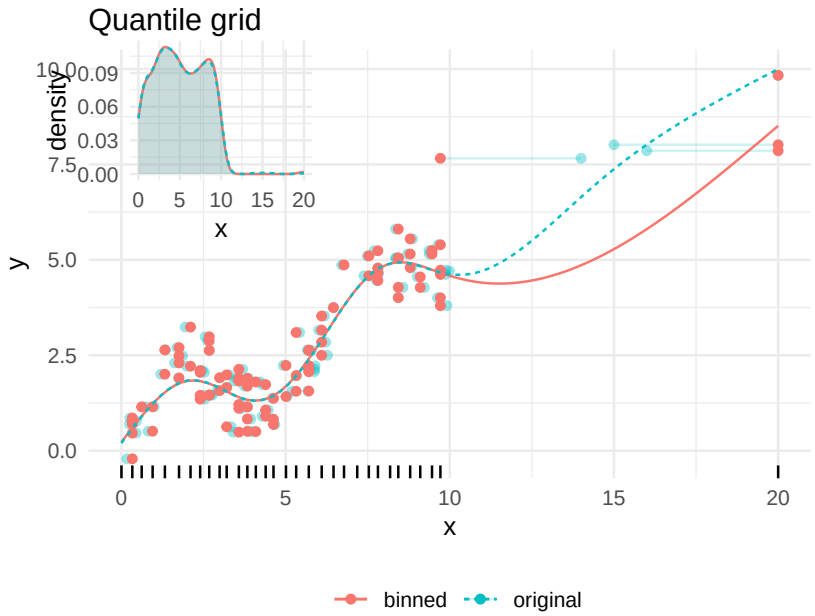
Binning: Grid comparison with uniformly distributed data



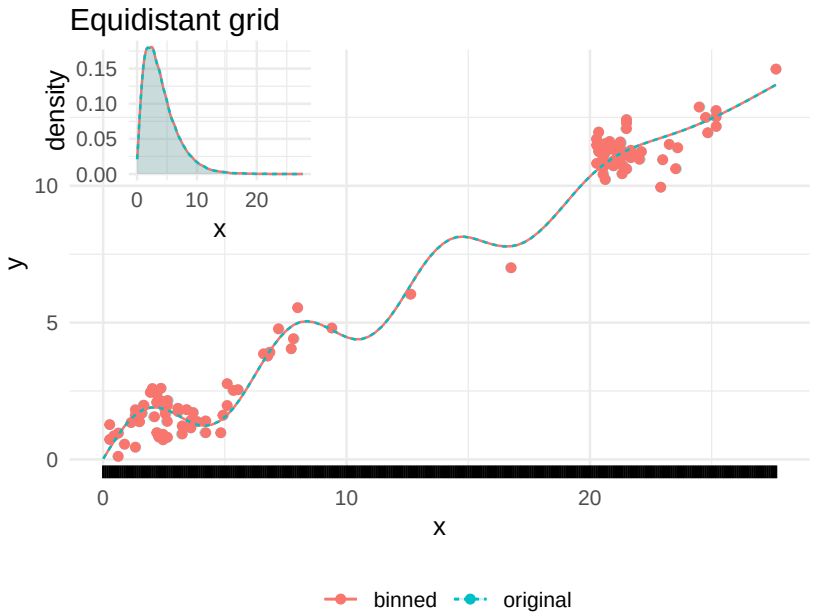


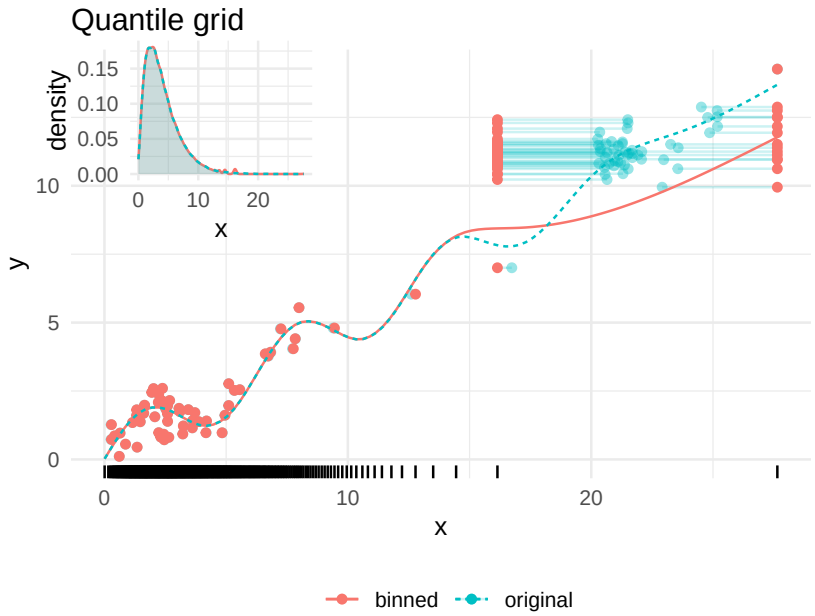
Binning: Grid comparison with outlier



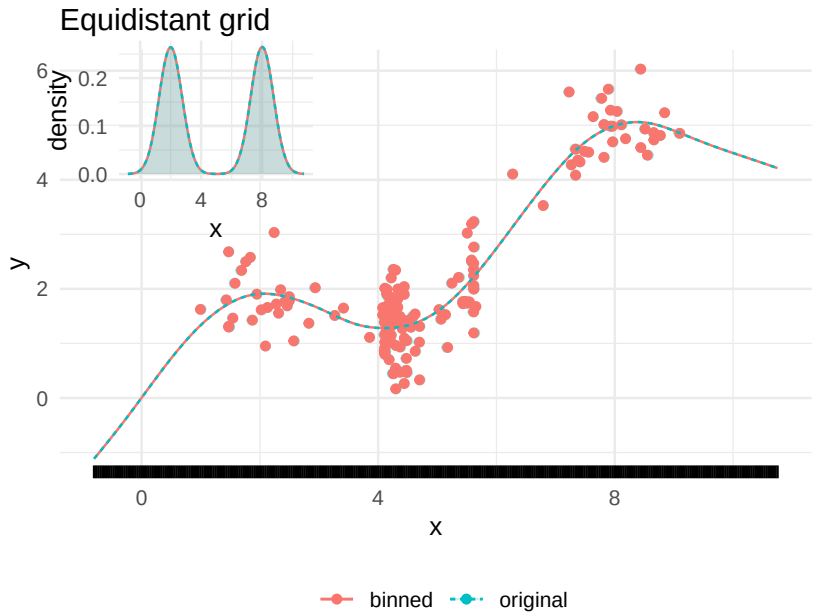


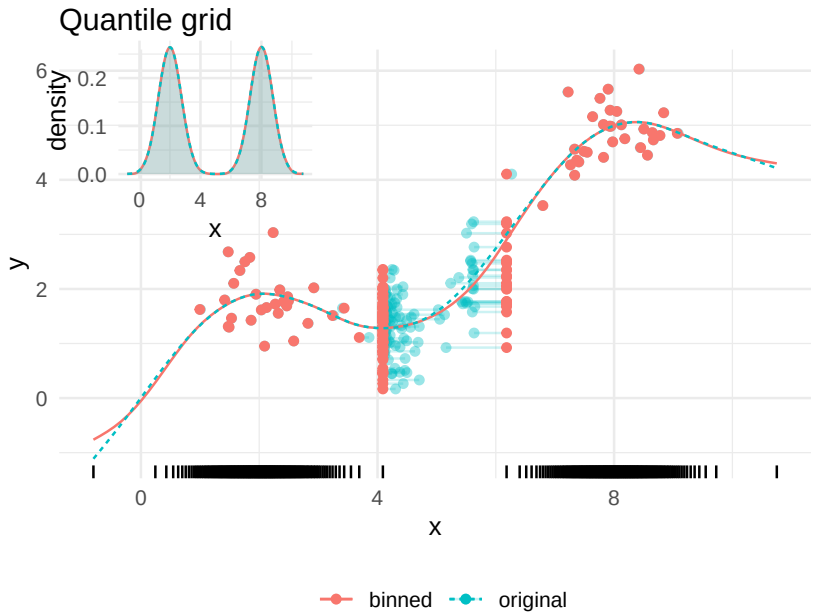
Binning: Grid comparison with skewed distribution (χ^2)





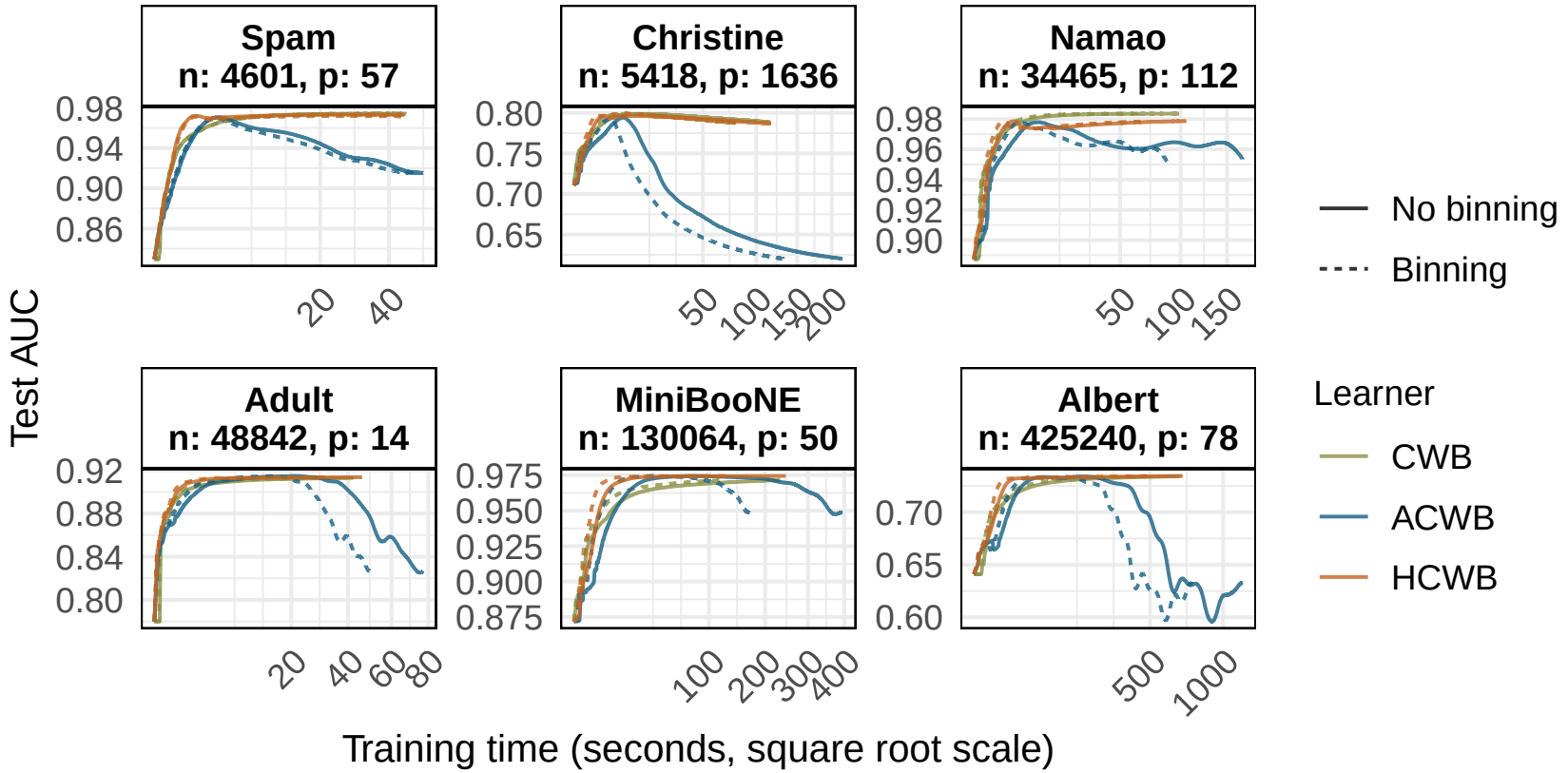
Binning: Grid comparison with multimodal distribution





Efficiency: Further comparison

Runtime comparisons of CWB variants

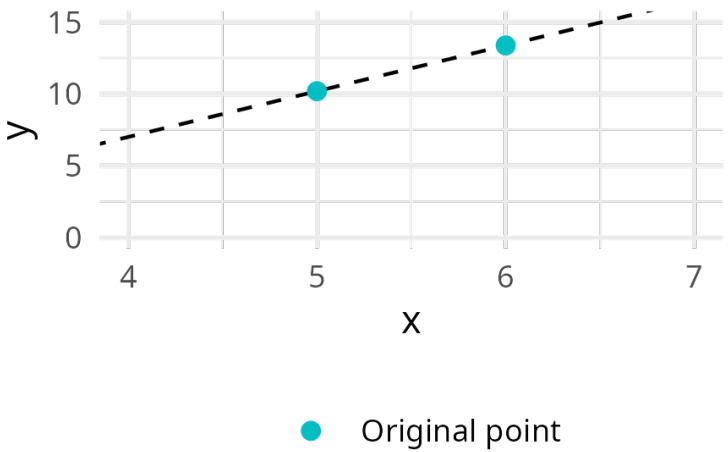


(Figure reference: Schalk et al. (2022a))

5000 boosting iterations without early stopping.

Reconstruct original data

Reconstruct data

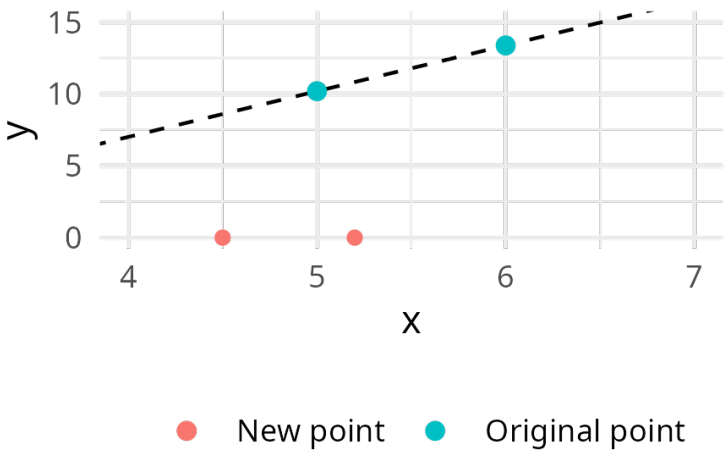


Given: Basis transformation g and

$$\left.\begin{matrix} Z^\top Z \\ Z^\top y \end{matrix}\right\} \Rightarrow \hat{\boldsymbol{\theta}}$$

Also: $n = d$

Reconstruct data



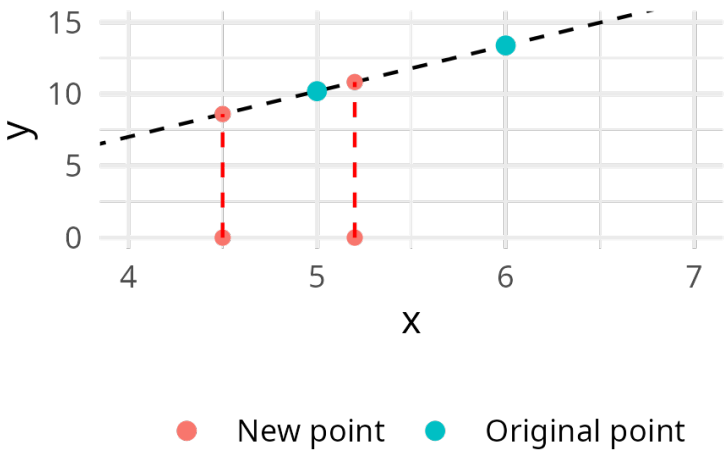
Given: Basis transformation g and

$$\left.\begin{matrix} Z^\top Z \\ Z^\top y \end{matrix}\right\} \Rightarrow \hat{\boldsymbol{\theta}}$$

Also: $n = d$

1. Guess new $\mathbf{x}_0 \in \mathbb{R}^n$

Reconstruct data



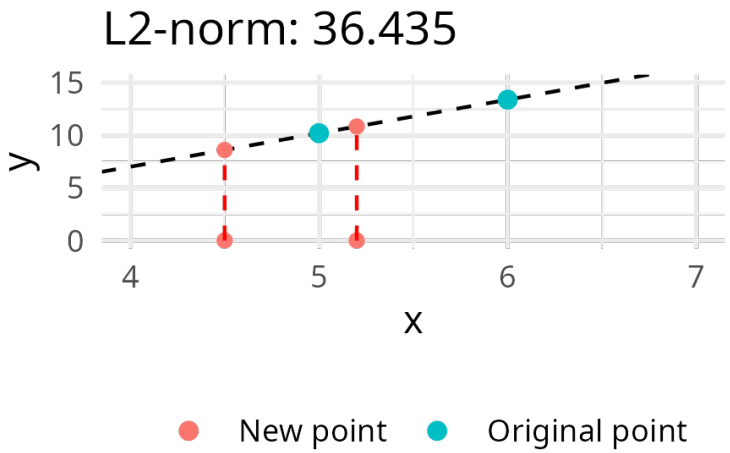
Given: Basis transformation g and

$$\left.\begin{matrix} Z^\top Z \\ Z^\top y \end{matrix}\right\} \Rightarrow \hat{\boldsymbol{\theta}}$$

Also: $n = d$

1. Guess new $\mathbf{x}_0 \in \mathbb{R}^n$
2. Calculate $Z_0 = (g(\mathbf{x}_0^{(1)})^\top, \ldots, g(\mathbf{x}_0^{(n)})^\top)^\top$.
3. Predict $\hat{\mathbf{y}}_0 = Z_0 \hat{\boldsymbol{\theta}}$.

Reconstruct data



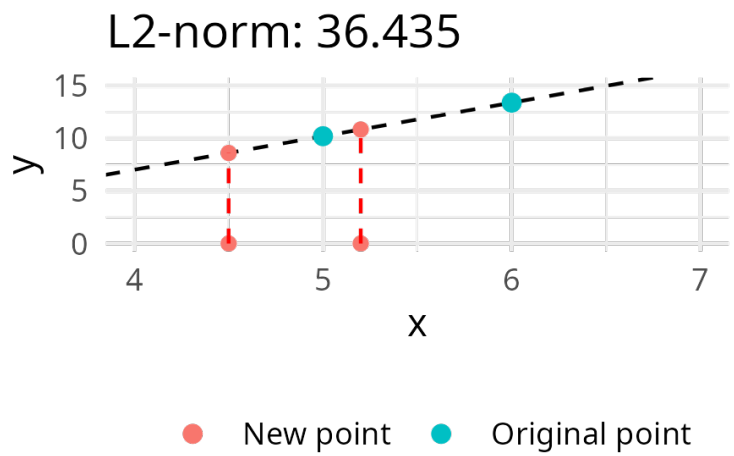
Given: Basis transformation g and

$$\left.\begin{matrix} Z^\top Z \\ Z^\top y \end{matrix}\right\} \Rightarrow \hat{\boldsymbol{\theta}}$$

Also: $n = d$

1. Guess new $\mathbf{x}_0 \in \mathbb{R}^n$
2. Calculate $Z_0 = (g(\mathbf{x}_0^{(1)})^\top, \ldots, g(\mathbf{x}_0^{(n)})^\top)^\top$.
3. Predict $\hat{\mathbf{y}}_0 = Z_0 \hat{\boldsymbol{\theta}}$.
4. Measure the distance $m(\mathbf{x}_0) = \|Z^\top y - Z_0^\top \hat{\mathbf{y}}_0\|$.

Reconstruct data



Given: Basis transformation g and

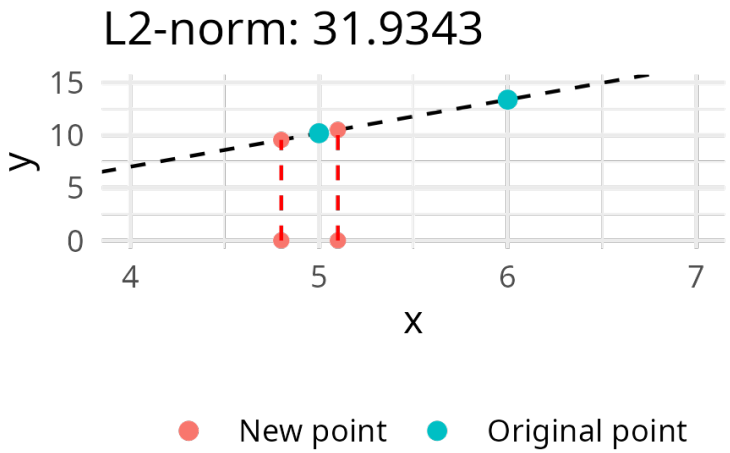
$$\left.\begin{matrix} Z^\top Z \\ Z^\top y \end{matrix}\right\} \Rightarrow \hat{\boldsymbol{\theta}}$$

Also: $n = d$

1. Guess new $\mathbf{x}_0 \in \mathbb{R}^n$
2. Calculate $Z_0 = (g(\mathbf{x}_0^{(1)})^\top, \ldots, g(\mathbf{x}_0^{(n)})^\top)^\top$.
3. Predict $\hat{\mathbf{y}}_0 = Z_0 \hat{\boldsymbol{\theta}}$.
4. Measure the distance $m(\mathbf{x}_0) = \|Z^\top y - Z_0^\top \hat{\mathbf{y}}_0\|$.

Optimize $\mathbf{x}^* = \arg\min_{\mathbf{x}_0 \in \mathbb{R}^n}(m(\mathbf{x}_0))$

Reconstruct data



Given: Basis transformation g and

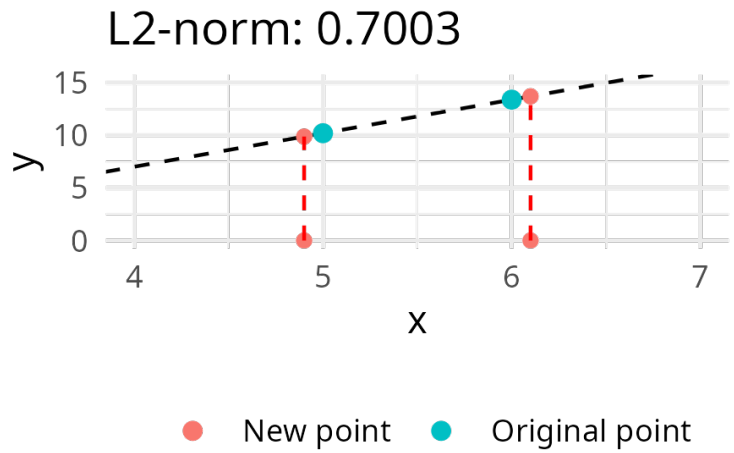
$$\left.\begin{matrix} Z^\top Z \\ Z^\top y \end{matrix}\right\} \Rightarrow \hat{\boldsymbol{\theta}}$$

Also: $n = d$

1. Guess new $\mathbf{x}_0 \in \mathbb{R}^n$
2. Calculate $Z_0 = (g(\mathbf{x}_0^{(1)})^\top, \ldots, g(\mathbf{x}_0^{(n)})^\top)^\top$.
3. Predict $\hat{\mathbf{y}}_0 = Z_0 \hat{\boldsymbol{\theta}}$.
4. Measure the distance $m(\mathbf{x}_0) = \|Z^\top y - Z_0^\top \hat{\mathbf{y}}_0\|$.

Optimize $\mathbf{x}^* = \arg\min_{\mathbf{x}_0 \in \mathbb{R}^n}(m(\mathbf{x}_0))$

Reconstruct data



Given: Basis transformation g and

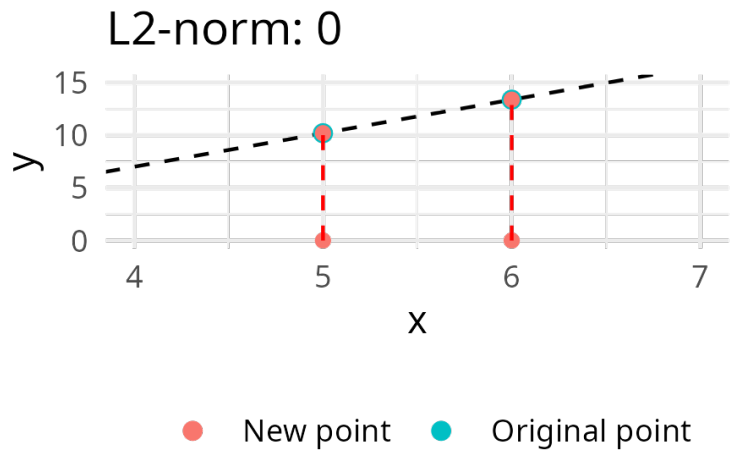
$$\left.\begin{matrix} Z^\top Z \\ Z^\top y \end{matrix}\right\} \Rightarrow \hat{\boldsymbol{\theta}}$$

Also: $n = d$

1. Guess new $\mathbf{x}_0 \in \mathbb{R}^n$
2. Calculate $Z_0 = (g(\mathbf{x}_0^{(1)})^\top, \ldots, g(\mathbf{x}_0^{(n)})^\top)^\top$.
3. Predict $\hat{\mathbf{y}}_0 = Z_0 \hat{\boldsymbol{\theta}}$.
4. Measure the distance $m(\mathbf{x}_0) = \|Z^\top y - Z_0^\top \hat{\mathbf{y}}_0\|$.

Optimize $\mathbf{x}^* = \arg\min_{\mathbf{x}_0 \in \mathbb{R}^n}(m(\mathbf{x}_0))$

Reconstruct data



Given: Basis transformation g and

$$\left.\begin{matrix} Z^\mathsf{T}Z \\ Z^\mathsf{T}y \end{matrix}\right\} \Rightarrow \hat{\boldsymbol{\theta}}$$

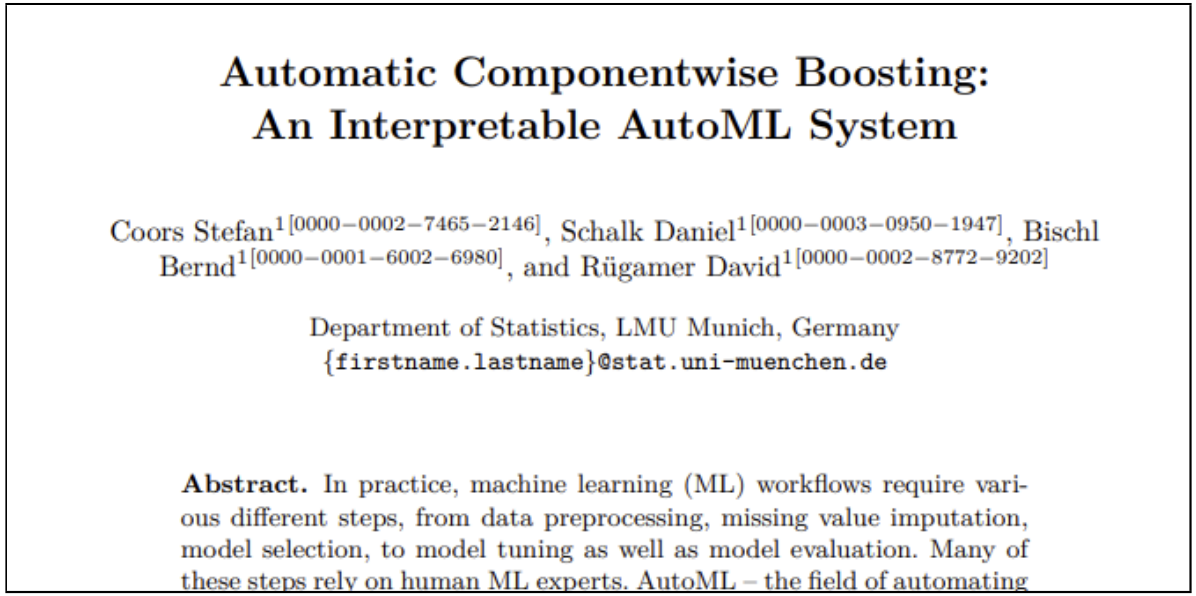
Also: $n = d$

1. Guess new $\mathbf{x}_0 \in \mathbb{R}^n$
2. Calculate $Z_0 = (g(\mathbf{x}_0^{(1)})^\mathsf{T}, \ldots, g(\mathbf{x}_0^{(n)})^\mathsf{T})^\mathsf{T}$.
3. Predict $\hat{\mathbf{y}}_0 = Z_0\hat{\boldsymbol{\theta}}$.
4. Measure the distance $m(\mathbf{x}_0) = \|Z^\mathsf{T}y - Z_0^\mathsf{T}\hat{\mathbf{y}}_0\|$.

Optimize $\mathbf{x}^* = \arg\min_{\mathbf{x}_0 \in \mathbb{R}^n}(m(\mathbf{x}_0)) \Rightarrow$ Calculate $y = (Z_0^\mathsf{T})^{-1}Z^\mathsf{T}y$

Automation

Publication [3]



Automatic Componentwise Boosting: An Interpretable AutoML System

Coors Stefan[1 [0000−0002−7465−2146]], Schalk Daniel[1 [0000−0003−0950−1947]], Bischl Bernd[1 [0000−0001−6002−6980]], and Rügamer David[1 [0000−0002−8772−9202]]

Department of Statistics, LMU Munich, Germany
{firstname.lastname}@stat.uni-muenchen.de

Abstract. In practice, machine learning (ML) workflows require various different steps, from data preprocessing, missing value imputation, model selection, to model tuning as well as model evaluation. Many of these steps rely on human ML experts. AutoML – the field of automating

Aims:

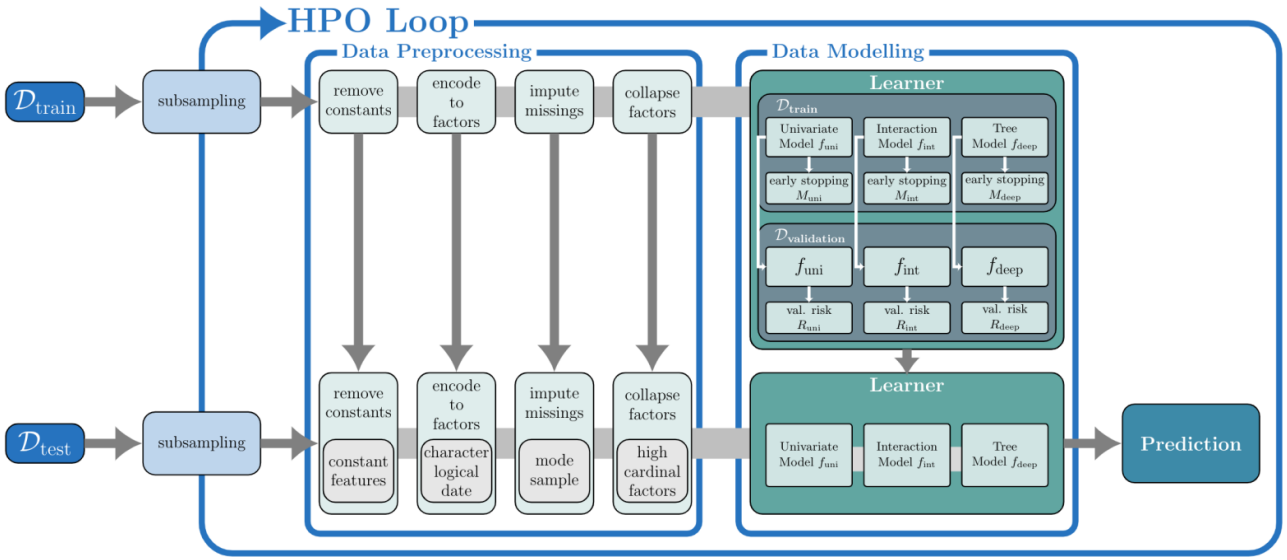
- Interpretable automated ML (AutoML) system with CWB as the fitting engine.
- Assessment of the required model complexity and the decision-making process.

About

- Build a model $f = f_{\text{uni}} + f_{\text{pint}} + f_{\text{deep}}$ based on a multi-stage approach with:
  - Univariate effects f_{uni} decomposed into $f_{\text{uni,linear}}$ and $f_{\text{uni,non-linear}}$
  - Pairwise interactions f_{pint} with a RWTP base learner
  - Deeper interactions f_{deep} with trees

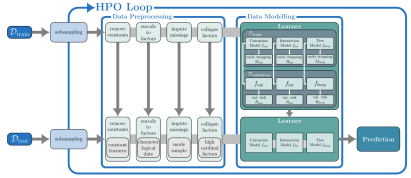
About

- Build a model $f = f_{\text{uni}} + f_{\text{pint}} + f_{\text{deep}}$ based on a multi-stage approach with:
  - Univariate effects f_{uni} decomposed into $f_{\text{uni,linear}}$ and $f_{\text{uni,non-linear}}$
  - Pairwise interactions f_{pint} with a RWTP base learner
  - Deeper interactions f_{deep} with trees
- `Autocompboost` wraps `compboost` with an AutoML pipeline:

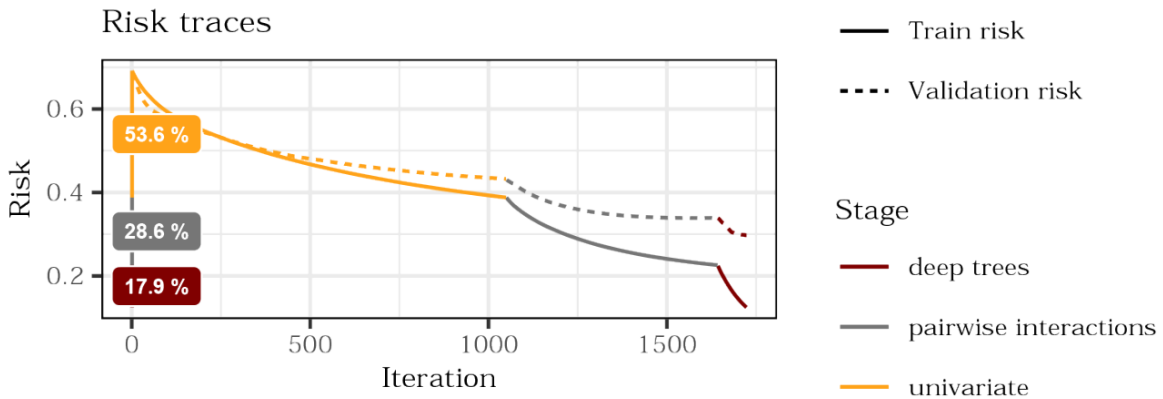


About

- Build a model $f = f_{\text{uni}} + f_{\text{pint}} + f_{\text{deep}}$ based on a multi-stage approach with:
  - Univariate effects f_{uni} decomposed into $f_{\text{uni,linear}}$ and $f_{\text{uni,non-linear}}$
  - Pairwise interactions f_{pint} with a RWTP base learner
  - Deeper interactions f_{deep} with trees
- `Autocompboost` wraps `compboost` with an AutoML pipeline:

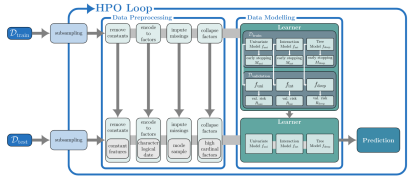


- Using boosting allows to gain a fine grid of the risk improvement for each stage:

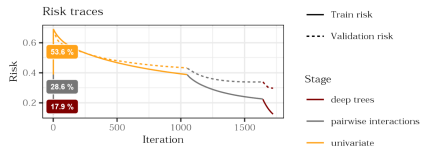


About

- Build a model $f = f_{\text{uni}} + f_{\text{pint}} + f_{\text{deep}}$ based on a multi-stage approach with:
  - Univariate effects f_{uni} decomposed into $f_{\text{uni,linear}}$ and $f_{\text{uni,non-linear}}$
  - Pairwise interactions f_{pint} with a RWTP base learner
  - Deeper interactions f_{deep} with trees
- `Autocompboost` wraps `compboost` with an AutoML pipeline:



- Using boosting allows to gain a fine grid of the risk improvement for each stage:



- Explaining the framework is done by using the CWBs interpretation functionality.

Limitation and Outlook

Limitations:

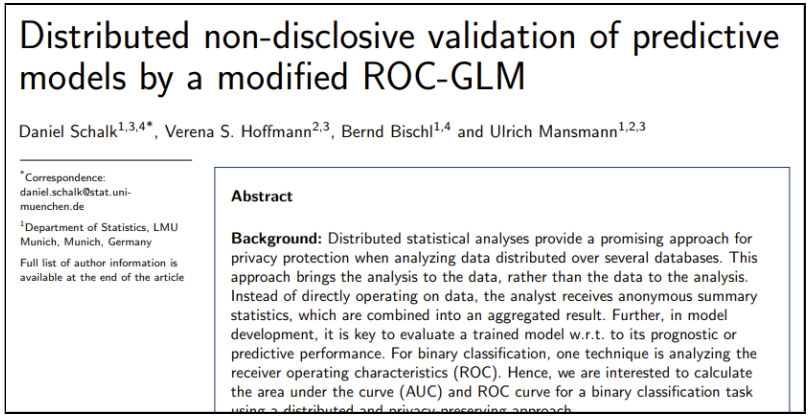
- Detecting interactions is done heuristically by a surrogate random forest and hence on a different model class than stage one and two.
- It is not clear how switching the model class in stage can revert the effects of stage one and two.

Outlook:

- Other techniques to detect interactions and simulations to show their effectiveness.
- Focus on the third stage. An idea is to orthogonalize f_{deep} by $f_{\text{uni}} + f_{\text{pint}}$ to ensure that their effects remain untouched.

Distributed model evaluation

Publications [5,6]



Distributed non-disclosive validation of predictive models by a modified ROC-GLM

Daniel Schalk[1,3,4*], Verena S. Hoffmann[2,3], Bernd Bischl[1,4] and Ulrich Mansmann[1,2,3]

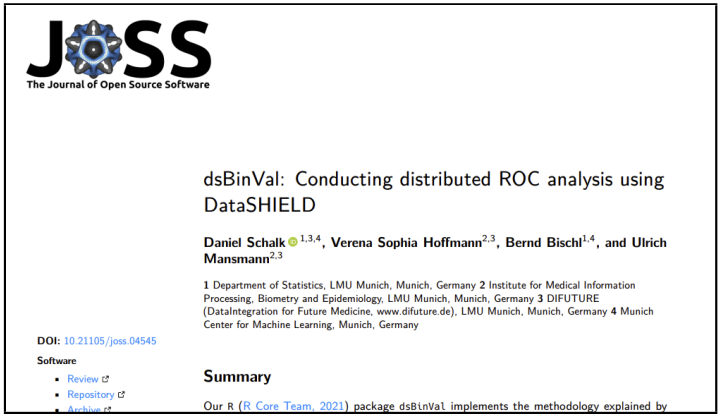
*Correspondence: daniel.schalk@stat.uni-muenchen.de

[1]Department of Statistics, LMU Munich, Munich, Germany

Full list of author information is available at the end of the article

Abstract

Background: Distributed statistical analyses provide a promising approach for privacy protection when analyzing data distributed over several databases. This approach brings the analysis to the data, rather than the data to the analysis. Instead of directly operating on data, the analyst receives anonymous summary statistics, which are combined into an aggregated result. Further, in model development, it is key to evaluate a trained model w.r.t. to its prognostic or predictive performance. For binary classification, one technique is analyzing the receiver operating characteristics (ROC). Hence, we are interested to calculate the area under the curve (AUC) and ROC curve for a binary classification task



JOSS The Journal of Open Source Software

dsBinVal: Conducting distributed ROC analysis using DataSHIELD

Daniel Schalk [1,3,4], Verena Sophia Hoffmann[2,3], Bernd Bischl[1,4], and Ulrich Mansmann[2,3]

1 Department of Statistics, LMU Munich, Munich, Germany 2 Institute for Medical Information Processing, Biometry and Epidemiology, LMU Munich, Munich, Germany 3 DIFUTURE (DataIntegration for Future Medicine, www.difuture.de), LMU Munich, Munich, Germany 4 Munich Center for Machine Learning, Munich, Germany

DOI: 10.21105/joss.04545

Software

- Review
- Repository
- Archive

Summary

Our R (R Core Team, 2021) package dsBinVal implements the methodology explained by

Aims:

- Privacy-preserving and distributed evaluation based on the AUC.
- Implementation in **DataSHIELD** (Gaye et al., 2014) to validate binary classification models.

Challenge

Many performance measures $\rho(\mathbf{y}, \hat{\mathbf{y}})$ that are based on a point-wise loss $L_\rho(y, \hat{f}(\mathbf{x}))$ can be calculated securely by:

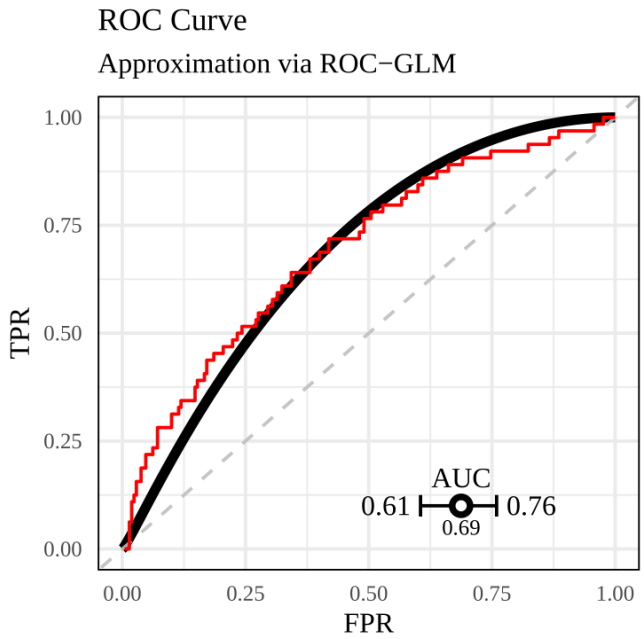
- Sharing $l_s = \sum_{(\mathbf{x},y)\in\mathcal{D}_s} L_\rho(y, \hat{f}(\mathbf{x}))$
- Calculating $\rho(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{s=1}^{S} w_s l_s$ (e.g. $L_\rho(y, \hat{f}(\mathbf{x})) = (y - \hat{f}(\mathbf{x})^2)$ and $w_s = 1$ for $\rho = \mathsf{SSE}$)

But:

- The AUC requires global information about the predictions scores (the order) for calculation.
- Merging these objects is not allowed without security concerns.

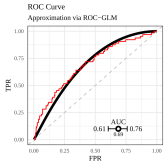
About

- Initialized by **DIFUTURE** (Prasser et al., 2018) to validate a treatment decision score for multiple sclerosis patients.
- Privacy-preserving and distributed calculation of the ROC-GLM as parametric approximation of the empirical ROC curve.



About

- Initialized by **DIFUTURE** (Prasser et al., 2018) to validate a treatment decision score for multiple sclerosis patients.
- Privacy-preserving and distributed calculation of the ROC-GLM as parametric approximation of the empirical ROC curve.



- Privacy is ensured by relying on aggregations as well as incorporating differential privacy (Dwork, 2006).