Adaptive Step Length Selection in Gradient Boosting for Generalized Additive Models for Location, Scale and Shape

Boyao Zhang
Supervisor: Prof. Dr. Sonja Greven
Co-Supervisor: Dr. Elisabeth Waldmann
Tobias Hepp

Master Thesis

Institut für Statistik
Ludwig–Maximilians–Universität München

Munich, May 1, 2019

Abstract

Generalized additive models for location, scale and shape (GAMLSS) are an approach that regresses not only the expected mean value as the conventional generalized additive models but also other distribution parameters. Fitting the GAMLSS with boosting algorithm allows simultaneous estimation of predictor effects and variable selection. The non-cyclical componentwise gradient boosting approach reduces the optimizing procedure from a multi-dimensional to a one-dimensional problem with vastly decreased complexity. Tuning in boosting algorithm relies mainly on the number of iterations of the algorithm. The other flexible component step length in most cases is set to 0.1. When developing complex models like GAMLSS, this setting will lead to unbalanced decisions. This thesis studied the influence of the adaptive step length on this balance and other performance measures.

Based on the simulation study, the adaptive approach usually updates the distribution parameters in a balanced manner. Within the limited number of boosting iterations, the adaptive approach will also lead to better estimations than the fixed step length settings, especially when the coefficients are huge. When fitting the high dimensional data, the adaptive approach is more efficient in computing. This thesis also introduced a semi-analytical adaptive step length (SAASL) algorithm for the Gaussian distribution, which is faster and more stable in balance than the adaptive step length found by doing a line search. Based on the mathematical induction, the optimal step length of the scale parameter in Gaussian distribution converges to 0.5. Applying this step length to the SAASL will result in a much more faster algorithm (SAASL05) at the cost of slightly unbalanced decisions. Because of the aggressive step length in each iterations, the adaptive approaches cannot good estimated the correlated models.

***Keywords*—** GAMLSS, gradient boosting, componentwise gradient boosting, adaptive step length, semi-analytical adaptive step length

Acknowledgement

Firstly, I would like to express my sincere gratitude to my supervisor Prof. Dr. Sonja Greven for her support of my thesis for her patience, motivation and immense knowledge. Her advice gave me a lot of inspiration.

Besides my supervisor, I would like to acknowledge Dr. Elisabeth Waldmann and M.Sc. Tobias Hepp, Faculty of Medicine, Friedrich-Alexander-Universität Erlangen-Nürnberg. They listened to my ideas patiently and directed me with lots of valuable advice.

I would like to thank Dr. David Rügamer, Department of Statistics, Ludwig-Maximilians-Universität München for his encouragement and help.

I would also like to thank my friends Miao Liu and Yingxin Cai for their support and encouragement. They make my life more colourful.

Finally, I must express very profound gratitude to my parents and my grandma for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis.

You are miracles in my life. Thank you.

Contents

List of Algorithms

List of Figures

1 Introduction

The generalized additive models for location, scale and shape (GAMLSS) were introduced by [Rigby and Stasinopoulos, 2005], which regress the univariate response with a set of statistical models. It is a more generalized model of the conventional generalized additive models (GAM) [Hasties and Tibshirani, 1990], as the latter regress only the location parameter. Given a set of covariates, the GAMLSS do not require the conditional distribution of the response variable to be a member of the exponential family. The optional distributions for GAMLSS can be found from the work of [Stasinopoulos and Rigby, 2007]. Another feature of GAMLSS is that every distribution parameter is modelled by its own predictor and an associated link function [Mayr et al., 2012]. The estimation of the coefficients is usually based on the penalized maximum likelihood [Rigby and Stasinopoulos, 2005]. Just like the other common used regression models (e.g. linear model or GAMs), the variable selection is an important procedure, especially for the high dimensional data. The generalized Akaike information criterion (GAIC) can be used as a selection method [Rigby and Stasinopoulos, 2005], but this procedure is infeasible when there are more covariates than observations [Mayr et al., 2012]. Moreover, other shortcomings like the inclusion of a large number of non-informative variables [Ripley, 2004] are also inherited by the GAIC. Some authors [Bühlmann and Yu, 2003] showed that the gradient boosting can be applied to fit the generalized additive models, and they also found that the variable selection procedure is included in the modified algorithm, i.e. the componentwise gradient boosting algorithm, which updates only one predictor in each iteration. This approach was also generalized to the GAMLSS models (denoted as *gamboostLSS*) [Mayr et al., 2012], which performs the estimation and variable selection simultaneously. The original gamboostLSS algorithm is a "cyclical" fitting, i.e. every distribution parameters of the univariate response variable will be updated in each iteration. As the gradient boosting an algorithm that tends to select a relatively high number of false-positive variables, some authors [Thomas et al., 2018] introduced a "non-cyclical" fitting that combines the gamboostLSS with the stability selection [Meinshausen and Bühlmann, 2010], which is a generic method that investigates the importance of the covariates in a statistical model by repeatedly subsampling the data. By this way, not only the variable selection, but also a selection of the best submodel (location, scale, or shape) that leads to the largest improvement in model fit is also performed in the "non-cyclical" approach. Moreover, the maximum number of boosting iterations for each distribution parameter can be replaced with the overall number of iterations. Thus tuning the complete model reduces from a multi-dimensional to a one-dimensional optimization problem. The computing time, hence, reduced drastically.

In contrast with the "cyclical" algorithm, the "non-cyclical" fitting, however, destroyed the internal balance between the distribution parameters. In other words, some parameters will be updated more frequently than the others. If the cost of large numbers of iteration can be ignored, this unbalanced decisions will not affect the final estimations, as all distribution parameters can be fitted sufficiently. But if the maximum of the number of boosting iteration is limited, those parameters whose potential improvement are intrinsically small will get little chance to be updated.

A possible solution to the unbalanced decision is using the adaptive step length to update the predictor in each iteration, i.e. finding the optimal step length based on the reduction of the empirical risk, and use it to update the base-learners. By this way, the predictors of distribution parameters with a vast improvement in each iteration can be updated rapidly. Thus, the balance of decision is kept, because the remaining improvement of these predictors and the potential improvement of the other parameters, whose values are intrinsically small, are on the same level.

The idea of using the adaptive step length in gradient boosting was introduced by [Friedman, 2001], whose value in each iteration is estimated by performing a line search. However, the step length in most cases is set to 0.1, because some authors [Bühlmann and Hothorn, 2007] argued that the use of this adaptive step length is unnecessary, as the procedure of doing a line search costs additional computing time. Nearly all publications accepted this setting and study mainly the number of iterations of the algorithm or its practical applications. Nothing has been published that study the influence of the step length on the balance of decisions in GAMLSS models.

This thesis studied the effect of the step length on this balance and compared the performance of the estimation between the adaptive step length and the fixed step length (set with 0.1). Accounting for the additional runtime of the line search, we also introduced a semi-analytical method (SAASL) to determining the adaptive step length in Gaussian distribution, which computes the adaptive step length analytically instead of an optimizing procedure. As the analytical solution to the scale parameter in Gaussian distribution does not exits, we replace its optimal step length in each iteration with a constant asymptotic value (0.05), which is even though not an adaptive value but a more reasonable and appropriate value. So we get a new algorithm (SAASL05), which quits the optimizing procedure and result in a more faster algorithm with almost little costs.

This thesis is organized as follows: In section 2 we describe briefly the generalized additive models for location, scale and thape (GAMLSS). The theory of generalized additive models for location, scale and shape (GAMLSS) is introduced in section 3. Section 4 demonstrated the boosted GAMLSS and listed the "cyclical" and "non-cyclical" componentwise gradient boosting algorithms for GAMLSS. Section 5 describes the step length in gradient boosting, including the effectiveness of fixed step length, the line search method used in R program when finding the adaptive step length, and the induction of the analytical adaptive step length. The results of simulation experiments will be demonstrated in section 6. The final section 7 summarises advantages and shortcomings of each step length approach and concludes this thesis.

2 Generalized Additive Models for Location, Scale and Shape

Generalized additive models for location, scale and shape (GAMLSSs) were introduced by Ridgby and Stasinopoulos (2005) as a general class of statistical models for the univariate response variable. The model assumes independent observations of the response variable given the explanatory variables, the model parameters as well as the random effects. Given a set of explanatory variables, the conditional distribution of response variable in GAMLSS can be selected from a very general family of distributions instead of the exponential family that generalized additive models (GAM) required.

2.1 Model Definition

The p distribution parameters $\boldsymbol{\theta}^T = (\theta_1, \theta_2, \cdots, \theta_p)$ of a density function $f(y|\boldsymbol{\theta})$ are modelled by using a set of additive models. The model class assumes that the observations y_i for $i \in 1, \cdots, n$ are conditionally independent given a set of explanatory variables and random effects.

Let $\mathbf{y}^T = (y_1, y_2, \cdots, y_n)$ be the vector of response variable, and let $g_k(\cdot), k = 1, \cdots, p$ be a known monotonic link function that relates the explanatory variables and random effects through an additive model given by

$$g_k(\boldsymbol{\theta}_k) = \boldsymbol{\eta}_k = \mathbf{X}_k \boldsymbol{\beta}_k + \sum_{j=1}^{J_k} \mathbf{Z}_{jk} \boldsymbol{\gamma}_{jk}, \tag{2.1}$$

where $\boldsymbol{\theta}_k$ and $\boldsymbol{\eta}_k$ are vectors of length n, and $\boldsymbol{\eta}_k$ is also called *predictors*, $\boldsymbol{\beta}_k^T = (\beta_{1k}, \beta_{2k}, \cdots, \beta_{J'_k k})$ is a parameter vector of length J'_k, $\mathbf{X}_k$ is a known design matrix of order $n \times J'_k$, $\mathbf{Z}_{jk}$ is a fixed known $n \times q_{jk}$ design matrix and $\boldsymbol{\gamma}_{jk}$ is a q_{jk}-dimensional random variable. This model (2.1) is called GAMLSS.

The model as given in Eq. (2.1) allows combinations of different types of additive random-effects terms to be incorporated by specifying $\mathbf{Z}_{jk}$ and $\boldsymbol{\gamma}_{jk}$, For example $J_k = 0$, the model then reduces to a fully parametric model:

$$g_k(\boldsymbol{\theta}_k) = \boldsymbol{\eta}_k = \mathbf{X}_k \boldsymbol{\beta}_k, \tag{2.2}$$

for other types of effect, see [Rigby and Stasinopoulos, 2005].

The GAMLSS in Eq. (2.2) provided a classical linear effect of the explanatory variables $\mathbf{X_k}$ on the response, i.e. $f_{\text{linear}}(\mathbf{X}_k) = \mathbf{X}_k \boldsymbol{\beta}_k$. However, for a smooth non-linear effect $f(\mathbf{X}_k) = f_{\text{smooth}}(\mathbf{X}_k)$ represented by regression splines, as well as spatial effects or random effects, a more general form of the effect is required. So, in practise and also in this thesis, we use the following GAMLSS:

$$g_k(\boldsymbol{\theta}_k) = \boldsymbol{\eta}_k = f(\mathbf{X}_k \boldsymbol{\beta}_k). \tag{2.3}$$

where the intercept β_0 is included in $\boldsymbol{\beta}_k$. Obviously, if the location parameter ($\boldsymbol{\theta}_1 = \boldsymbol{\mu}$) is the only distribution parameter to be regressed on the explanatory variables and the response variable is from the exponential family, a GAMLSS reduces to the conventional GAM.

The two important distribution parameters, that are usually characterized in GAMLSS, are the location $\boldsymbol{\theta}_1 = \boldsymbol{\mu}$ and scale $\boldsymbol{\theta}_2 = \boldsymbol{\sigma}$ parameter. For other families of distributions, the two shape parameters, skewness $\boldsymbol{\theta}_3 = \boldsymbol{\nu}$ and kurtosis $\boldsymbol{\theta}_4 = \boldsymbol{\tau}$, are also simultaneously modelled in GAMLSS. Thus, GAMLSS usually model these four parameters, but theoretically, any distribution with any number of parameters can be applied to GAMLSS.

2.2 Model Estimation

The unknown parameters in GAMLSS can be estimated by maximizing the log-likelihood

$$\ell = \sum_{i=1}^{n} \log\{f(y_i|\boldsymbol{\theta}_i)\} = \sum_{i=1}^{n} \log\{f(y_i|\mu_i, \sigma_i, \nu_i, \tau_i)\} \tag{2.4}$$

The estimates of each components of $\boldsymbol{\theta}_i$ are then obtained from back-transforming the estimates of the prediction functions, which are denoted by $\hat{\eta}_{\theta_{ik}}, k \in \{1, \cdots, 4\}$, via the inverse link:

$$\begin{aligned} \hat{\mu}_i &= g_1^{-1}(\hat{\eta}_{\theta_{i1}}) \\ \hat{\sigma}_i &= g_2^{-1}(\hat{\eta}_{\theta_{i2}}) \\ \hat{\nu}_i &= g_3^{-1}(\hat{\eta}_{\theta_{i3}}) \\ \hat{\tau}_i &= g_4^{-1}(\hat{\eta}_{\theta_{i4}}) \end{aligned} \tag{2.5}$$

A penalized likelihood approach based on the modified versions of the back-fitting algorithm for general GAM estimation [Mayr et al., 2012] is used to estimate the predictor functions η_{θ_k}. Two algorithms were developed based on the principle: in each iteration, back-fitting steps are successively applied to the distribution parameters, with the sub-model fits of the previous iteration used as offset values for those parameters that are not involved in the current back-fitting step [Mayr et al., 2012], for more details, see [Rigby and Stasinopoulos, 2005].

3 Gradient Boosting

In machine learning theory, boosting is considered to be one of the most potent ideas. It is mainly used as a technique for solving regression and classification problems, which fits the prediction model as an ensemble of weak learners. The weak learners are defined as a prediction rule with a correct classification rate that is at least slightly better than random guessing, i.e. more than 50% accuracy, as a comparison, strong learners should be able to be trained to have a nearly perfect classification, e.g. 99% accuracy. It is typically easy to construct a weak learner in practice, whereas very difficult to get a strong one.

Any weak learner can be iteratively boosted to become a strong learner [Schapire, 1990]. AdaBoost (Adaptive Boosting) [Freund and Schapire, 1996] was the first generated boosting algorithm based on this idea. In AdaBoost, the base-learner is sequentially applied to weighted training observations. Before the next iteration, the misclassified observations receive a higher weight, repeat this process until the adequate number of misclassified observations is met [Freund and Schapire, 1997].

A more commonly used boosting method is the gradient boosting [Friedman, 2001]. Unlike the AdaBoost, which can only solve the binary classification problems, the gradient boosting provides a more general framework, and the paradigm is developed for additive expansions based on any fitting criterion. Based on the framework of gradient boosting, many algorithms have been developed: Gradient boosting of regression trees produces highly robust and interpretable procedures for both regression and classification [Friedman, 2001]. Componentwise gradient boosting [Bühlmann and Yu, 2003] incorporates the variable selection procedure into the learning process. XGBoost [Chen and Guestrin, 2016] provides a scalable tree boosting system and is able to solve problems using a minimal amount of resources. The LightGBM [Ke et al., 2017] developed a leaf-wise tree growth strategy and that have great performance in terms of computational speed and memory consumption.

In this section, we describe only the mechanism of the gradient boosting and the componentwise gradient boosting.

3.1 Gradient Boosting

Gradient boosting [Friedman, 2001] is probably the most widely used boosting technique, which builds a connection between stagewise additive expansions and steepest descent minimization.

3.1.1 Gradient Descent

Let $f(\mathbf{x})$ be an arbitrary, differentiable objective function, which we want to minimize. The gradient $\nabla f(\mathbf{x}) = \left(\frac{df}{dx_1}, \cdots, \frac{df}{dx_k}\right)$ is the direction of the steepest ascent, where k is the dimensions of vector $\mathbf{x}$, correspondingly, the steepest descent is $-\nabla f(\mathbf{x})$. Given the current point $\mathbf{x}^{[m]}, m = 1, \cdots, M$, the updated $\mathbf{x}^{[m+1]}$, which result in a lower value of the objective function (i.e. $f(\mathbf{x}^{[m]}) > f(\mathbf{x}^{[m+1]})$), is calculated by

$$\mathbf{x}^{[m+1]} = \mathbf{x}^{[m]} - \nu \nabla f(\mathbf{x}^{[m]}), \tag{3.1}$$

where ν controls the *step length* towards steepest descent.

The process described in Eq. (3.1) is called *gradient descent.* Gradient descent is a greedy algorithm, i.e. it moves toward the local minimum in every iteration. If $f(\mathbf{x})$ is a convex function, this algorithm can find the global minimum, on the other hand, if $f(\mathbf{x})$ is a non-convex function, it can only find a local minimum, and just might find a global one, which depends on the initial value.

The step length ν is an essential parameter of the gradient descent algorithm, as it influences the learning speed and also affects whether the minimum can be found or not. If ν is very small, the learning process will converge very slowly, however, if it is enormous, the process may not converge, because $\mathbf{x}$ jumps around the "valley".

The step length ν can either be set manually with a constant value or be estimated in each iteration with some methods. Here, we call the former *fixed step length*, and the latter *adaptive step length.* Usually, the estimation of the adaptive step length is carried by doing a line search. In this thesis, we induced an analytical solution for Gaussian distribution, which can also be used as an adaptive step length. Details will be discussed in Section 5.

3.1.2 Stagewise Additive Expansions

Another part of gradient boosting is the forward stagewise additive expansions, which estimate the prediction function as an additive model in a forward stagewise way.

Assume a space of base learners H and $h \in H$, the additive model can be displayed as:

$$\eta(x) = \sum_{i=1}^{M} \nu^{[m]} h(x, \theta^{[m]}), \tag{3.2}$$

where ν and θ are the weights/step length and the parameter in the base learner $h(\cdot, \cdot)$ correspondingly. Given the training data $(y^{(i)}, x^{(i)}), i = 1, \cdots, n$, the regression model is fitted by minimizing the *empirical risk* $\mathcal{R}$, which is defined as:

$$\mathcal{R} = \frac{1}{n} \sum_{i=1}^{n} \rho\left(y^{(i)}, \eta(x^{(i)})\right) \tag{3.3}$$

$$= \frac{1}{n} \sum_{i=1}^{n} \rho\left(y^{(i)}, \sum_{m=1}^{M} \nu^{[m]} h(x^{(i)}, \theta^{[m]})\right), \tag{3.4}$$

where ρ is the *loss function.* The desired $\nu^{[m]}$ and $\theta^{[m]}$ are found by minimizing the empirical risk $\mathcal{R}$:

$$\min_{\nu^{[m]}, \theta^{[m]}} \sum_{i=1}^{n} \rho\left(y^{(i)}, \eta^{[m]}(x^{(i)})\right) = \min_{\nu^{[m]}, \theta^{[m]}} \sum_{i=1}^{n} \rho\left(y^{(i)}, \sum_{i=1}^{M} \nu^{[m]} h(x^{(i)}, \theta^{[m]})\right) \tag{3.5}$$

However, this problem requires computationally intensive numerical optimization techniques [Hastie et al., 2009], an alternative problem, which minimizes the risk only with respect to the next component, is often used in practice:

$$\min_{\nu, \theta} \sum_{i=1}^{n} \rho\left(y^{(i)}, \eta^{[m-1]} + \nu h(x^{(i)}, \theta)\right). \tag{3.6}$$

The loss function measures the discrepancy between the true value of $y^{(i)}$ and the additive learner $\eta(x^{(i)})$. The most widely used loss function is squared-error or L2 loss $(y^{(i)} - \eta(x^{(i)}))^2$ for regression problem, and binomial loss $-y^{(i)}\eta(x^{(i)}) + \log(1 + \exp(\eta(x^{(i)})))$ for binary classification problem, i.e. $y^{(i)} \in \{0, 1\}$. Usually, depending on the desired model, the loss is derived from the negative log likelihood of the distribution of Y.

Algorithm 1 described the process of the forward stagewise additive modeling.

Algorithm 1 Forward Stagewise Additive Modeling

```
1: Initialize $\hat{f}^{[0]} = 0$
2: for $m = 1 \to M$ do
3:     Compute $(\hat{\nu}^{[m]}, \hat{\theta}^{[m]}) = \arg\min_{\nu,\theta} \sum_{i=1}^{n} \rho\left(y^{(i)}, \hat{f}^{[m-1]} + \nu h(x^{(i)}, \theta)\right)$
4:     Set $\hat{f}^{[m]} = \hat{f}^{[m-1]} + \nu^{[m]} h(x, \hat{\theta}^{[m]})$
5: end for
```

3.1.3 Gradient Boosting

Gradient boosting incorporates the ideas of gradient descent and forward stagewise additive modeling. The required parameters in the base learner can be estimated by minimizing the empirical risk. The gradient descent is a numerical optimizations method that helps to estimate these unknown parameters, and the stagewise additive modeling established a way that combines all individual base-learners as an ensemble model.

The gradient of the empirical risk $\mathcal{R}$ at one observation point $x^{(j)}, j \in \{1, \cdots, n\}$ is

$$\frac{\partial \mathcal{R}}{\partial \eta(x^{(j)})} = \frac{\partial \sum_{i=1}^{n} \rho\left(y^{(i)}, \eta(x^{(i)})\right)}{\partial \eta(x^{(j)})} \tag{3.7}$$

$$= \frac{\partial \rho\left(y^{(j)}, \eta(x^{(j)})\right)}{\eta(x^{(i)})} \tag{3.8}$$

The gradient descent update at this observation can be calculated by:

$$\eta(x^{(j)}) \leftarrow \eta(x^{(j)}) - \nu \frac{\partial \rho\left(y^{(j)}, \eta(x^{(j)})\right)}{\partial \eta(x^{(j)})}, \tag{3.9}$$

and correspondingly, the gradient descent for all observations is then:

$$\eta(x) \leftarrow \eta(x) - \nu \frac{\partial \rho\left(y, \eta(x)\right)}{\partial \eta(x)} \tag{3.10}$$

Eq.(3.10) described the gradient descent procedure and tells direction, where the function $\eta(x)$ should be updated or moved. As in stagewise additive modeling, the real η used for risk minimization is $\eta^{[m-1]}$, finding the optimal value of the unknown parameter $\theta^{[m]}$ results in a regression problem between

$$u^{[m]} = -\left[\frac{\partial \rho\left(y, \eta(x)\right)}{\partial \eta(x)}\right]_{\eta=\eta^{[m-1]}} \tag{3.11}$$

and additive component or base learner $h(x, \theta^{[m]}) \in H$. We call $u^{[m]}$ the *pseudo residuals*, as for squared loss they match the normal residuals, i.e.

$$-\frac{\partial\rho\left(y,\eta(x)\right)}{\partial\eta(x)} = -\frac{\partial(y-\eta(x))^2}{\partial\eta(x)} = 2\underbrace{(y-\eta(x))}_{\text{normal residuals}} . \tag{3.12}$$

For the regression problem, the unknown parameter $\theta^{[m]}$ in base learner $h(x, \theta^{[m]})$ can be simply estimated by minimizing the sum of squared error:

$$\hat{\theta}^{[m]} = \arg\min_{\theta} \sum_{i=1}^{n} \left(u^{[m](i)} - h(x^{(i)}, \theta)\right)^2 . \tag{3.13}$$

Back to Eq. (3.10), the step length $\nu^{[m]}$ can be then found by minimizing the empirical risk, that is,

$$\hat{\nu}^{[m]} = \arg\min_{\nu} \sum_{i=1}^{n} \rho\left(y^{(i)}, \eta^{[m-1]}(x^{(i)}) + \nu h(x^{(i)}, \theta^{[m]})\right) . \tag{3.14}$$

We formally present the procedure of gradient boosting in Algorithm 2.

Algorithm 2 Gradient Boosting Algorithm

```
1: Initialize $\hat{\eta}^{[0]}(x) = \arg\min_{\theta} \sum_{i=1}^{n} L(y^{(i)}, \theta)$
2: for $m = 1 \to M$ do
3:     For all $i \in \{1, \cdots, n\}$ calculate the pseudo-residuals:
           $u^{[m](i)} = -\left[\frac{\partial\rho\left(y^{(j)}, \eta(x^{(j)})\right)}{\partial\eta(x^{(j)})}\right]_{\eta=\hat{\eta}^{[m-1]}}$
4:     Fit a regression base learner $h(x^{(i)}, \theta)$ to the pseudo-residuals $u^{[m](i)}$ and estimate its parameters:
           $\hat{\theta}^{[m]} = \arg\min_{\theta} \sum_{i=1}^{n} \left(u^{[m](i)} - h(x^{(i)}, \theta)\right)^2$
5:     Find the step length via:
           $\hat{\nu}^{[m]} = \arg\min_{\nu} \sum_{i=1}^{n} \rho\left(y^{(i)}, \eta^{[m-1]}(x^{(i)}) + \nu h(x^{(i)}, \theta^{[m]})\right)$
6:     Update
           $\hat{\eta}^{[m]}(x) = \hat{\eta}^{[m-1]}(x^{(i)}) + \hat{\nu}^{[m]} h(x^{(i)}, \hat{\theta}^{[m]})$
7: end for
8: Output $\hat{\eta}(x) = \hat{\eta}^{[M]}(x)$
```

Various base-learners $h(x, \theta)$ can be applied to the gradient boosting framework. A regression tree is such a common used base-learner in machine learning applications.

3.2 Componentwise Gradient Boosting

Componentwise gradient boosting is an extended version of gradient boosting, which aims at optimizing prediction accuracy and at obtaining statistical model estimates [Hofner et al., 2014]. The key property of this method is that it carries out variable selection during the learning process [Bühlmann, 2006]. Moreover, componentwise gradient boosting result in prediction rules that have the same interpretation ability as the common statistical models. Account for this features, *componentwise gradient boosting* is also often referred as *model-based boosting* or in short *mboost*.

Compared with the usual gradient boosting, which uses only one kind of base-learner, componentwise gradient boosting select the best learner from a set of base-learners in each iteration. In other words, in each iteration a set of base learners $h_j^{[m]}(x, \theta^{[m]}), j = 1, \cdots, J$ (where j is indexes the type of base learner) are used to fit the model, but only the arbitrary j-te best performing base learner $h_j^{[m]}(x, \theta^{[m]})$ will be finally used in the current iteration. Accordingly, the corresponding additive models become:

$$h_j^{[m]}(x, \theta^{[m]}) + h_j^{[m+m']}(x, \theta^{[m+m']}) = h_j(x, \theta^{[m]} + \theta^{[m+m']}). \tag{3.15}$$

In practise, only one type of base learners is used for gradient boosting, but these base learners are not defined on the whole predictive variables, but on only one variable x_j, i.e. $h_j^{[m]}(x_j, \theta^{[m]}), j = 1, \cdots, p$.

The critical feature of componentwise gradient boosting lies in that the variable selection mechanism is done simultaneously, because only the best performing learner is selected in each iteration, and for those variables which have little influence on the target variable will be ignored during the modeling. By this way, only the most informative explanatory variables instead of a learner with all variables will be included in the final model until stopping.

A formal definition of componentwise gradient boosting is given in Algorithm 3.

3.3 Regularization

Due to the aggressive loss minimization, it can easily overfit the data if gradient boosting runs for a large number of iterations. If a model is underfitted, it will be too simple to explain the variance in the explanatory variables, in other words, the intrinsic relationship between explanatory variables and dependent variables cannot be estimated good enough. However, if a model is overfitted, it tends to fit the noise behaved in the explanatory variables and failed for the generalization to new data.

There are two main methods for avoiding overfitting, and the one is limit the number of additive components by stopping the boosting iterations early m_{stop}. The other way is to shorten the step length $\nu^{[m]}$ in each iteration by multiplying a *shrinkage parameter* $\lambda \in (0, 1]$, so that the predictors can be in a more conservative manner updated.

Algorithm 3 Componentwise Gradient Boosting

```
1: Initialize $\hat{\eta}^{[0]}(x) = \arg\min_{\theta} \sum_{i=1}^{n} \rho(y^{(i)}, \theta)$
2: for $m = 1 \to M$ do
3:     For all $i \in \{1, \cdots, n\}$ calculate the pseudo-residuals:
           $$u^{[m](i)} = -\left[\frac{\partial \rho\left(y^{(j)}, \eta(x^{(j)})\right)}{\partial \eta(x^{(j)})}\right]_{\eta=\hat{\eta}^{[m-1]}}$$
4:     for $j = 1 \to J$ do
5:         Fit regression base learner $h_j$ to the pseudo-residuals $u^{[m](i)}$ and estimate its parameters:
               $$\hat{\theta}_j^{[m]} = \arg\min_{\theta_j} \sum_{i=1}^{n} \left(u^{[m](i)} - h_j(x^{(i)}, \theta_j)\right)^2$$
6:     end for
7:     Find the best fitting learner:
           $$j^* = \arg\min_{j} \sum_{i=1}^{n} \left(u^{[m](i)} - h_j(x^{(i)}, \theta_j)\right)^2$$
8:     Update:
           $$\hat{\eta}^{[m]}(x) = \hat{\eta}^{[m-1]}(x^{(i)}) + \nu h_{j^*}(x^{(i)}, \hat{\theta}_{j^*}^{[m]})$$
9: end for
10: Output $\hat{\eta}(x) = \hat{\eta}^{[M]}(x)$
```

3.3.1 Shrinkage parameter

The shrinkage parameter method is as introduced above nothing special but multiplying a small shrinkage effect λ to the base-learners, i.e. a modified update equation in Algorithm 2:

$$\hat{\eta}^{[m]}(x) = \hat{\eta}^{[m-1]}(x^{(i)}) + \lambda \hat{\nu}^{[m]} h(x^{(i)}, \hat{\theta}^{[m]}). \tag{3.16}$$

Obviously, λ strongly depends on the number of iterations M. For a sufficient large M, a small λ can be conservatively selected and vice versa.

Theoretically, the step length should be optimized according to the Algorithm 2, i.e. obtained by minimizing the empirical risk. By this way, the shrinkage parameter can shorten the step length in each iteration and affect the final model. However, the step length can also be manually given, which is also the most widely used method. In this situation, the shrinkage parameter λ seems to be a redundant setting, as one can take the shrinkage effect into consideration when setting the step length artificially.

In case of the reading confusion in following sections, we highlight the shrinkage settings here: Firstly, for the adaptive step length, the step length searched or calculated based on the risk minimization, we set the shrinkage parameter $\lambda = 0.1$. Secondly, if the step length is set artificially, we do not need to set the shrinkage parameter any more, or in other words, set it as 1.

3.3.2 Early stopping

The early stopping is another strategy used for regularization. According to the idea of the forward stagewise additive modeling, more components will be added into the models if lots of iterations have been performed. By limiting the number of boosting iterations, less but only the most essential informative variables can become the final predictors. The early stopping is mainly carried out by cross-validation and information criteria.

- Cross-Validation (CV)

  For cross-validation, the m_{stop} can be determined by the behaviour of prediction errors. In general, the prediction error will decrease with the increasing number of iterations before overfitting, as the predictive model can gather useful information from the additive components in these iterations. As long as the prediction errors start to increase, it can be regarded as a sign for overfitting and stop further learning.

  Let $\kappa : \{1, \cdots, n\} \mapsto \{1, \cdots, K\}$ be an indexing function that indicates the partition to which observation i is allocated by the randomization. The k-fold cross-validation estimate of prediction error is

$$\text{CV}(\hat{f}) = \frac{1}{n} \sum_{i=1}^{N} \rho(y_i, \hat{f}^{-\kappa(i)}(x_i)), \tag{3.17}$$

  where $\hat{f}^{-\kappa}(x)$ denotes the fitted function, computed with the k-th part of data removed [Hastie et al., 2009]. The CV with $K = n$ is also known as the *leave-one-out* cross-validation.

  The choice of K will influence not only the computing time but also the bias-variance tradeoff. For example, the leave-one-out CV is approximately unbiased for the correct expected prediction error; however, it can have high variance as the n training sets are quite similar to one another [Hastie et al., 2009]. At the same time, the required n times of learning method is also a considerable computational burden. On the other hand, with a relatively small value of K, the CV has even though lower variance, the possible higher bias depending on how the performance of the model varies with the size of the observations must be taken into consideration.

  Considering the computational complexity and the compromise between bias and variance, the folds recommended by some authors are 5 [Breiman and Spector, 1992] and 10 [Kohavi et al., 1995].

- Information Criteria (IC)

Information criteria is another common strategy used for determining the early stopping value. This strategy is computationally far less intensive than the CV. There are many variations of the information criteria (for more details, see [Bozdogan, 2000]), here we introduce only the generalized Akaike information criterion (GAIC).

The GAIC [Akaike, 1983] with a fixed penalty λ is given by

$$\mathrm{GAIC}(\lambda) = \mathrm{GD} + \lambda \mathrm{df} \tag{3.18}$$

where GD denotes the fitted global deviance, df is the degrees of freedom used in the model. The common choice for global deviance is $-2 \log L(\hat{\theta})$ [Sclove, 1987]. Akaike information criterion (AIC) [Akaike, 1974] is a special case of GAIC with $\lambda = 2$. And similarly, the Bayesian information criterion (BIC) or Schwarz Bayesian criterion (SBC) [Schwarz, 1978] with $\lambda = \log(n)$ are also a specific form of the GAIC. As n is in most cases extensive, the BIC tends to penalize more heavily than AIC, giving preference to simpler models in evaluation [Hastie et al., 2009].

No matter which form is used, the interpretation for each term in Eq.(3.18) is quite similar. The first term GD provides a measure of bias or model inaccuracy. The other term serves a penalty λ for the increased unreliability or compensation for the bias in the first term when additional free parameters are included in the model [Bozdogan, 2000]. Consequently, when evaluating the performance of a set of competing models, the values of GAIC can be computed and compared to select a model with the smallest GAIC. Using GAIC allows different penalties λ to be tried for different modelling purposes. The sensitivity of the chosen model to the choice of λ can also be investigated [Rigby and Stasinopoulos, 2005].

Note that for GAMLSS, the degrees of freedom is the total effective freedom that is used in the model [Rigby and Stasinopoulos, 2005]. In other words, the summation of all degrees of freedom used to fit the individual distribution parameters [Stasinopoulos et al., 2017], for example in the Gaussian distribution the $\mathrm{df} = \mathrm{df}_\mu + \mathrm{df}_\sigma$. Even though this thesis defines the degrees of freedom in this way, we still need to address that there is no commonly accepted approach to measure the degrees of freedom of a boosting fit [Hofner et al., 2016]. Due to the algorithmic nature of gradient boosting, which results in the regularized model fits, the complexity of the model is difficult to evaluate [Hastie, 2007]. As a result, the problem of deriving valid and easy-to-compute complexity measures for boosting remains largely unsolved [Bühlmann et al., 2014].

Thus, although we give the early stopping values specified by IC in the section 6, we still suggest using the CV as the primary regularization strategy.

When a model has fitted with the adaptive step length boosting algorithm, it is easy to fall into a misunderstanding, that the model is estimated better along with the increasing numbers of boosting iteration, the step length should converge to zero. A stopping criterion established on the adaptive step length might be possible. But this is just an intuition. In the section 5, we can find, that the adaptive step length is not an independent hyperparameter, and might not converge. Even some distribution parameters converge, its limit is also not zero. So we believe that it is impossible to build stopping criteria based on the adaptive step length.

4 Boosted GAMLSS

Though GAMLSS can be applied to a very general family of distributions, it still has some shortcoming from the penalized likelihood approach [Thomas et al., 2018]. Firstly, it is impossible to estimate the models that have more explanatory variables than observations. Secondly, the variable selection procedure is not embedded in the maximum likelihood estimation. Finally, either linear or non-linear predictors is not trivial to fit; unnecessary complexity increases the danger of overfitting and computing time.

4.1 Cyclical Boosted GAMLSS

The “cyclical” boosted GAMLSS [Mayr et al., 2012] were then introduced to overcome the shortcomings, because it does not rely on the generalized Akaike information criterion (GAIC) [Rigby and Stasinopoulos, 2005] for regularization, but provides a new method to estimate the GAMLSS prediction functions while simultaneously selecting appropriate sets of explanatory variables.

Recall the gradient boosting algorithm, the parameters in the base learner are estimated by minimizing the empirical risk (see Eq.(3.13)). Analogously, each predictor η with respect to each explanatory variable in cyclical boosted GAMLSS is obtained by minimizing the expectation of a loss function $\rho(\cdot)$:

$$\hat{\eta} = \arg\min_{\eta} \mathbb{E}[\rho(\mathbf{y}, \eta(\mathbf{X}))], \tag{4.1}$$

where $\mathbf{y}$ and $\mathbf{X}$ denote the response and explanatory variables respectively. Given a sample of observations $\{(y_i, x_i)\}, i \in \{1, \cdots, n\}$, it minimize the empirical risk

$$\hat{\eta} = \arg\min_{\eta} \sum_{i=1}^{n} \rho(y_i, \eta(x_i)). \tag{4.2}$$

For $\rho(\cdot)$ a L2 loss, Eq. (4.2) is identical to Eq. (3.13) and can be used to fit a conventional regression model. But the more common used loss function in GAMLSS model is the negative log-likelihood.

The whole algorithm is formally given in Algorithm 4 [Thomas et al., 2018].

According to the procedure of the algorithm, it is apparent that the data-driven mechanism for variable selection is included, as only the predictive model is updated through the best performing explanatory variable in each iteration. The less important variables will be ignored for a small value of m_{stop}. Another feature of the “cyclical” boosted GAMLSS algorithm lies in that it allows the situation for more explanatory variables than observations, as only one base learner is included in each iteration. This also avoided the problems of multicollinearity for high dimensional data [Mayr et al., 2012].

4.2 Non-Cyclical Boosted GAMLSS

As the levels of complexity of each distribution parameter in its prediction function are different and a various number of boosting iterations is required, separate stopping values for each

Algorithm 4 Cyclical componentwise gradient boosting in multiple dimensions

```
1: Initialize the additive predictors $\hat{\boldsymbol{\eta}}^{[0]} = (\hat{\eta}_{\theta_1}^{[0]}, \hat{\eta}_{\theta_2}^{[0]}, \hat{\eta}_{\theta_3}^{[0]}, \hat{\eta}_{\theta_4}^{[0]})$ with offset values.
2: For each distribution parameter $\theta_k, k = 1, \cdots, 4$, specify a set of base learners, i.e., for
   parameter $\theta_k$ define $h_{k1}(x^{(i)}), \cdots, h_{kJ_k}(x^{(i)})$ where $J_k$ is the cardinality of the set of base
   learners specified for $\theta_k$.
3: for $m = 1 \to \max(m_{\text{stop},1}, \cdots, m_{\text{stop},4})$ do
4:     for $k = 1 \to 4$ do
5:         if $m > m_{\text{stop, k}}$ then
6:             set $\hat{\eta}_{\theta_k}^{[m]} := \hat{\eta}_{\theta_k}^{[m-1]}$ and skip this iteration.
7:         else
8:             compute negative partial derivative $-\frac{\partial}{\partial \eta_{\theta_k}} \rho(y, \eta)$ and plug in the current estimates
   $\hat{\eta}^{[m-1]}(\cdot)$:
                 $$u_k = \left( -\frac{\partial}{\partial \eta_{\theta_k}} \rho(y, \eta) \Big|_{\eta = \hat{\eta}^{[m-1]}(x^{(i)}), y = y^{(i)}} \right)_{i=1,\cdots,n}$$
9:         end if
10:        Fit each of the base-learners $h_{kj}(\cdot)$ contained in the set of base-learners specified for
   the distribution parameter $\theta_k$ in step (2) to the negative gradient vector $u_k$.
11:        Select the component $j^*$ that best fits the negative partial derivative vector according
   to the residual sum of squares, i.e., select the base-learner $h_{kj^*}$ defined by
                 $$j^* = \underset{j \in 1, \cdots, J_k}{\arg\min} \sum_{i=1}^{n} \left( u_k^{(i)} - \hat{h}_{kj}(x^{(i)}) \right)^2 .$$
12:        Update the additive predictor $\eta_{\theta_k}$
                 $$\hat{\eta}_{\theta_k}^{[m]} = \hat{\eta}_{\theta_k}^{[m-1]} + \nu \cdot \hat{h}_{kj^*}(x),$$
   where $\nu$ is the step length, and update the current estimates for step (6):
                 $$\hat{\eta}_{\theta_k}^{[m-1]} = \hat{\eta}_{\theta_k}^{[m]}.$$
13:    end for
14: end for
```

parameter need to be specified. And the values of $m_{\text{stop},k}$ are not independent in the case of multi-dimensional, the usually applied grid search scales exponentially with the number of distribution parameters and can easily become computationally demanding [Thomas et al., 2018]. A "non-cyclical" approach [Thomas et al., 2018] was developed to solve the problem by updating only one distribution parameter in each iteration.

In "cyclical" boosted GAMLSS algorithm, the best fitting base-learners for all distribution parameters are selected by calculating the residual sum of squares with respect to the negative gradient vector (*inner loss*) in each iteration. In "non-cyclical" approach, the best performing distribution parameter must also be selected before the end of each iteration, and actually by comparing the empirical risk instead of the residual sum of squares, as the latter cannot be used to compare the fit of base-learners over different distribution parameters. Thus two approaches thus were introduced by the authors [Thomas et al., 2018]. The one is *inner loss* method. With this method, the best fitting distribution parameter is selected by comparing the empirical risk, but the best fitting base-learners are chosen by comparing the residual sum of squares. Nevertheless, they argue that choosing base-learners and parameters concerning two different optimization criteria may not always result in the best possible update, so they give another solution called "outer loss" method, which chooses the best performing base-learner also with the empirical risk.

The formal procedure of "non-cyclical" boosted GAMLSS is given in Algorithm 5.

Compared with the "cyclical" algorithm, the "non-cyclical" variants enable the m_{stop} to be scalar, and the distribution parameters are chosen adaptively. Thus the scalar optimization can be carried out very efficiently using standard cross-validation methods instead of the multi-dimensional grid search.

The original intention of developing this "non-cyclical" algorithm is not to improve computational efficiency, but to use the stability selection [Meinshausen and Bühlmann, 2010] on the boosted GAMLSS models. The stability selection approach is to run the base-learner selection algorithm on multiple subsamples of the original data. Highly relevant base-learners should be involved in (almost) all models learned from the subsamples [Thomas et al., 2018], (for more details about the combination of stability selection and boosting, see [Hofner et al., 2015]). They [Thomas et al., 2018] argued that, as all distribution parameters will be updated in each iteration in "cyclical" approach, the base-learners, which might have little importance than the base-learners for other distribution parameters, are also added to the model, the combination with stability selection will lead to severe problem.

However, the "non-cyclical" approach destroyed the selection balance among the distribution parameters. In extreme situations, some distribution parameters might never be selected within limited iterations. The reason is a mixture effect of the nature of the "non-cyclical" approach and the fixed step length used in gradient boosting. Take the linear additive models as an example: if the coefficients of some covariables concerning a distribution parameter are huge, their great improvements will make this parameter to be the best choice in the corresponding iterations. Moreover, as the fixed step length is usually small, the update in each iteration thus cannot gain many improvements. Consequently, the "non-cyclical" procedure will still select these parameters for updates.

Algorithm 5 Non-cyclical componentwise gradient boosting in multiple dimensions

```
1: Initialize the additive predictors $\hat{\eta}^{[0]} = (\hat{\eta}^{[0]}_{\theta_1}, \hat{\eta}^{[0]}_{\theta_2}, \hat{\eta}^{[0]}_{\theta_3}, \hat{\eta}^{[0]}_{\theta_4})$ with offset values.
2: For each distribution parameter $\theta_k, k = 1, \cdots, 4$, specify a set of base-learners, i.e., for
   parameter $\theta_k$ define $h_{k1}(\cdot), \cdots, h_{kJ_k}(\cdot)$ where $J_k$ is the cardinality of the set of base-learners
   specified for $\theta_k$.
3: for $m = 1$ to $m_{\text{stop}}$ do
4:     for $k = 1$ to 4 do
5:         Compute negative partial derivatives $-\frac{\partial}{\partial \eta_{\theta_k}} \rho(y, \eta)$ and plug in the current estimates
           $\hat{\eta}^{[m-1]}(\cdot)$:
           $$u_k = \left( -\frac{\partial}{\partial \eta_{\theta_k}} \rho(y, \eta) \Big|_{\eta = \hat{\eta}^{[m-1]}(x^{(i)}), y = y^{(i)}} \right)_{i=1,\cdots,n}$$
6:         Fit each of the base-learners $h_{kj}(\cdot)$ contained in the set of base-learners specified for
           the distribution parameter $\theta_k$ in step (2) to the negative gradient vector $u_k$.
7:         Select the best-fitting base-learner $h_{kj^*}$ either by
           • the inner loss, i.e., the residual sum of squares of the base-learner fit w.r.t. $u_k$:
             $$j^* = \underset{j \in 1, \cdots, J_k}{\arg\min} \sum_{i=1}^{n} \left( u_k^{(i)} - \hat{h}_{kj}(x^{(i)}) \right)^2$$
           • the outer loss, i.e., the negative log likelihood of the modeled distribution after the
             potential update:
             $$j^* = \underset{j \in 1, \cdots, J_k}{\arg\min} \sum_{i=1}^{n} \rho \left( y^{(i)}, \hat{\eta}^{[m-1]}_{\theta_k}(x^{(i)}) + \nu \cdot \hat{h}_{jk}(x^{(i)}) \right)$$
8:         Compute the possible improvement of this update regarding the outer loss
           $$\Delta\rho_k = \sum_{i=1}^{n} \rho \left( y^{(i)}, \hat{\eta}^{[m-1]}_{\theta_k}(x^{(i)}) + \nu \cdot \hat{h}_{kj^*}(x^{(i)}) \right)$$
9:     end for
10:    Update, depending on the value of the loss reduction $k^* = \arg\min_{k \in 1, \cdots, 4}$ only the overall
       best-fitting base-learner:
       $$\hat{\eta}^{[m]}_{\theta_{k^*}} = \hat{\eta}^{[m-1]}_{\theta_{k^*}} + \nu \cdot \hat{h}_{k^* j^*}(x)$$
11:    Set $\hat{\eta}^{[m]}_{\theta_k} := \hat{\eta}^{[m-1]}_{\theta_k}$ for all $k \neq k^*$.
12: end for
```

To overcome the unbalanced decisions in “non-cyclical” algorithm, we try to use the adaptive step length instead of the fixed one. With the adaptive step length, the updates in each iteration can gain an adaptive improvement. Thus, the great improvements exist in only some distribution parameters can decrease rapidly to a level that other parameters have. The selection in the afterwards iterations should “jump” between every distribution parameters.

5 Adaptive Step Length

Tuning in gradient boosting relies mainly on the number of iterations, the other flexible component step length ν of the algorithm is, in most cases, set to 0.1. Some authors [Friedman, 2001] has suggested estimating an adaptive step length in each iteration by performing a line search. Other authors [Bühlmann and Hothorn, 2007] have argued that this line search in general functional gradient descent algorithm can be omitted at the cost of doing more iterations but not necessarily more computing time based on empirical evidence and some mathematical reasoning, so they favoured to use a fixed step length.

This section will discuss more details about fixed and adaptive step length in boosted GAMLSS and induct a semi-analytical adaptive step length based on the Gaussian distribution.

5.1 Fixed Step Length (FSL)

As the name suggests, the fixed step length ν is constant when updating the additive predictor η_{θ_k} in all boosting iterations, a typical value is 0.1 (and 0.01 is also sometimes suggested). With FSL, one can focus on the other challenges of boosting algorithm, for example, the stopping values, and do not need to pay more attention to the tuning problems of step length as long as the step length is a small value.

Apparently, there is a negative relationship between the step length ν and the stopping iterations m_{stop}, i.e. a small ν yields a large m_{stop} and vice versa. The computing time of a boosting algorithm, on the one hand, depends on the complexity of base-learner, and the other hand, on the number of boosting iterations. Hence, the choice of step length is another challenge to the researchers when using boosting algorithms.

As introduced at the beginning of this section, the step length can be either adaptive or fixed. For an adaptive step length, its optimal value in each iteration is usually found by doing a line search. The fixed step length, however, is set artificially. Theoretically, the adaptive step length is a more reasonable choice, but it is just because of the additional computing required by line search, that make the adaptive solution not very popular in recent studies. In contrast, a small fixed step length is the widely used settings, which saves the compute time of the line search but may be at the cost of calculation on more iterations. But the research of [Bühlmann and Hothorn, 2007] showed that we do not need to concern more about the effectiveness of FSL, as for the task of minimizing the empirical risk, the gradient descent with FSL and a general loss function have the similar performance to L2-Boosting. Hence, the cost of additional boosting iterations can be covered by saving the line search. By this way, they argued that FSL does, of course, more iterations, but not necessarily more computing time. For more detailed mathematical induction, see Appendix A.1.

The mathematical evidence in [Bühlmann and Hothorn, 2007] proofed that the adaptive step length conducted by line search can be replaced with FSL without more computing cost, but this argument does not mean that the FSL is always the best choice. The effectiveness of their suggestion is based on the potential assumption that the FSL is somehow around or a little smaller than the step length suggested by the line search. Otherwise, a conservative small step length (e.g. $\nu = 0.0001$ or even smaller) would always be the best choice, which, on the other hand, causes the unnecessary computing cost of a large number of boosting iterations. Under the FSL settings, given a sequence of *sufficient small* step lengths $\{\nu_1, \nu_2, \cdots, \nu_n\}$, the maximum of which is the naturally best one, because it is not only small enough to update the predictors, but also large enough to avoid the unnecessary iterations.

5.2 Adaptive Step Length (ASL)

In the original algorithm of gradient boosting [Friedman, 2001], the step length is adaptive and found by a line search. Line search is an optimization strategy that approaches to find a local minimum from a given objective function. It finds firstly a descent direction from which the objective function will be reduced, and then move alongside the direction with an appropriate step size. The common methods which determine the descent direction are *Newton's method*, *Quasi-Newton method* [Nocedal and Wright, 1999] and *gradient descent* [Ruder, 2016]. No matter which one of the mentioned methods is used, the calculation of the derivative of the objective function is a must. But there are some shortcomings when using the derivative: Firstly, it can be difficult or impossible to compute the derivative of the objective function, or even be difficult to approximate the derivatives. Secondly, it is difficult to avoid the inflexion points, i.e. the points at which a curve changes from being concave to convex or vice versa. And finally, the optimization with derivative is likely no more efficient than the one without derivative [Brent, 2013].

In this thesis, the suggested optimization strategy is a combination of *Golden Section Search* and *Successive Parabolic Interpolation* [Brent, 2013].

Golden section search method is used in a one-dimensional optimization problem with an iterative numerical approach. The main idea of this method is to narrow down the interval successively inside which the minimum is known to exist with the help of the golden ratio until the length of the remaining interval is smaller than a given tolerance. Though the convergence rate is not so fast (linear), but this method guarantees to converge to the actual minimum and requires no derivatives. The pseudo-code of the golden section search method is given in Algorithm 6.

Successive parabolic interpolation is another technique used for finding the minimum of an objective function by successively fitting parabolas (polynomials of degree 2) at three points [R Core Team, 2018]. The oldest point is replaced with a new one at which the minimum of the fitted parabola is located at each iteration. This new point is also the approximation of the solution at the end of the search. The procedure can be described with a simple example: given three points $(x_i, y_i), i = 1, 2, 3$ and a parabola $ax^2 + bx + c$, the coefficients of the parabola a, b and c can be definitively found by solving a linear system with the three points (by ignoring the "overlapping" situation). Afterwards, the "oldest" of the previous three points is replaced with the new point $x_{\text{new}} = -\frac{b}{2a}$ until the tolerance is met.

Similar to the golden section search method, successive parabolic interpolation does also not need derivatives but is even faster, namely approximately superlinear. But the convergence of successive parabolic interpolation does not guarantees to find a minimum when in isolation. A

Algorithm 6 Golden Section Search

```
1: Define the Golden Ratio: $\phi = \frac{\sqrt{5}-1}{2} \approx 0.618$.
2: Initialize a tolerance $\epsilon$.
3: Select an interval $[a, b]$ containing the minimum.
4: while $|b - a| > \epsilon$ do
5:     Evaluate $f(x_1)$ at $x_1 = a + (1 - \phi)(b - a)$.
6:     Evaluate $f(x_2)$ at $x_2 = a + \phi(b - a)$.
7:     if $f(x_1) < f(x_2)$ then
8:         $b \leftarrow x_2$
9:         $f_{\min} \leftarrow x_1$
10:    else
11:        $a \leftarrow x_1$
12:        $f_{\min} \leftarrow x_2$
13:    end if
14: end while
15: Return $f_{\min}$.
```

simple example is collinearity among the three points. The resulting parabola is then linear, i.e. $a = 0$, and will not provide a new candidate point.

Brent introduced an excellent solution which takes advantages of both methods in 1973, who uses the combination of a golden section search and successive parabolic interpolation [Brent, 2013], here we call it *Brent's method*. Brent's method takes the successive parabolic interpolation as the primary procedure, as it converges faster. As usual, the minimum of the parabola is considered as the candidate for the new point used to narrow down the interval. But this point will only be accepted when it lies within the bounds of the current interval. Otherwise, the search stops and switches to the golden section search. By this way, the convergence rate is still approximate superlinear if the optimized function has a second derivative, but the combination is more reliable than solely parabolic interpolation.

5.3 Semi-Analytical Adaptive Step Length (SAASL)

The adaptive step length is obtained by performing a line search on the empirical risk. In contrast with other line search methods, the computation cost using Brent's method can be reduced to some extent. But it seems more natural to use an analytical optimal step length by performing a mathematical induction instead of line search. Such an optimal step length would have lots of advantages, for example, the time complexity would reduce to constant, as the step length can be directly calculated from the analytical expression instead of the iterative searching. Moreover, the properties of the adaptive step length can be understood from the mathematical expression, which might help to answer some questions, for example, if it is possible to construct a stopping criterion via the step length, because, intuitively, the optimal step length would decrease along with the increasing number of boosting iterations.

Since there are many advantages of the analytical optimal step length, the remaining question is whether such an expression exists. This thesis studied the behaviour of the adaptive step length in GAMLSS with respect to the Gaussian distribution as well as the linear model and found that only the adaptive step length of the location parameter can be displayed with a mathematical expression, while that of the scale parameter can not. That means the adaptive step length of the scale parameter still needs to be optimized by doing a line search. The induction is given as follows.

Given the data points $(y_i, \mathbf{x}_i), i \in \{1, \cdots, n\}$, where $\mathbf{x}$ is a $n \times p$ matrix. Assume that the true generating mechanism is

$$y_i \sim N(\mu_i, \sigma_i) \tag{5.1}$$

$$\mu_i = \mathbf{x}_i \beta_\mu \tag{5.2}$$

$$\sigma_i = \exp(\mathbf{x}_i \beta_\sigma) \tag{5.3}$$

where β_μ and β_σ are the coefficients of the corresponding linear models. Then in GAMLSS, the distribution parameters can be modelled with two additive predictors η_μ and η_σ:

$$\mu = g_\mu^{-1}(\eta_\mu) = \eta_\mu \tag{5.4}$$

$$\sigma = g_\sigma^{-1}(\eta_\sigma) = \exp(\eta_\sigma) \tag{5.5}$$

where $g^{-1}(\cdot)$ is the inverse link function.

In the conventional componentwise gradient boosting algorithm, only one variable can be chosen to update the predictors. So we set the base-learners as

$$h_\mu(x_j)$$
$$h_\sigma(x_j)$$

where $j \in \{1, 2, \cdots, p\}$ represents the indices of the corresponding variables. The additive predictors can then be displayed as:

$$\hat{\eta}_\mu^{[m]} = \hat{\eta}_\mu^{[m-1]} + \nu_\mu^{[m]} \hat{h}_\mu^{[m]}(x) \tag{5.6}$$

$$\hat{\eta}_\sigma^{[m]} = \hat{\eta}_\sigma^{[m-1]} + \nu_\mu^{[m]} \hat{h}_\sigma^{[m]}(x) \tag{5.7}$$

where $\nu_\mu^{[m]}$ and $\nu_\sigma^{[m]}$ are the adaptive step length in the m-te iteration.

Take the negative log-likelihood as the loss function, then the loss can be displayed as

$$\rho(y, \{\mu, \sigma\}) = -\log\left[\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)\right] \tag{5.8}$$

$$= \frac{1}{2}\log(2\pi) + \log\sigma + \frac{(y-\mu)^2}{2\sigma^2} \tag{5.9}$$

$$= \frac{1}{2}\log(2\pi) + \eta_\sigma + \frac{(y-\eta_\mu)^2}{2\exp(2\eta_\sigma)} \tag{5.10}$$

where the distribution parameters are replaced with the additive predictors.

The negative gradient descent or pseudo-residuals $u_i^{[m]}$ for each distribution parameter in iteration m is then

$$\begin{aligned} u_\mu^{[m]} &= -\frac{\partial \rho(y, \mu^{[m-1]}, \sigma^{[m-1]})}{\partial \eta_\mu^{[m-1]}} && (5.11)\\ &= \frac{1}{\exp(2\eta_\sigma^{[m-1]})}(y - \eta_\mu^{[m-1]}) && (5.12)\\ u_\sigma^{[m]} &= -\frac{\partial \rho(y, \mu^{[m-1]}, \sigma^{[m-1]})}{\partial \eta_\sigma^{[m-1]}} && (5.13)\\ &= -1 + \frac{1}{\exp(2\eta_\sigma^{[m-1]})}(y - \eta_\mu^{[m-1]})^2 && (5.14) \end{aligned}$$

Regress the pseudo-residuals with the base-learner $\hat{h}^{[m]}$ on the best fitting covariable $x_{j^*}, j^* \in \{1, \cdots, n\}$:

$$\begin{aligned} u_{\mu i}^{[m]} &= \hat{h}_\mu^{[m]}(x_{ij^*}) + \epsilon_{\mu i} && (5.15)\\ u_{\sigma i}^{[m]} &= \hat{h}_\sigma^{[m]}(x_{ij^*}) + \epsilon_{\sigma i} && (5.16) \end{aligned}$$

where ϵ is the error term in regression models.

With all of these assumptions and calculations, the process of the optimization can be divided into two parts, the one is the adaptive step length for the location parameter μ, and the other is for the scale parameter σ.

5.3.1 ASL for μ

Firstly, focus on the adaptive step length of μ. The analytical ASL for μ in iteration m can be obtained through minimizing the empirical risk,

$$\begin{aligned} \nu_\mu^{[m]} &= \underset{\nu_\mu^{[m]}}{\arg\min} \sum_{i=1}^{n} \rho(y_i, \{\mu_i^{[m]}, \sigma_i^{[m-1]}\}) && (5.17)\\ &= \underset{\nu_\mu^{[m]}}{\arg\min} \sum_{i=1}^{n} \frac{\left(y_i - \hat{\eta}_\mu^{[m-1]} - \nu_\mu^{[m]} \hat{h}_\mu^{[m]}(x_{ij^*})\right)^2}{2\sigma_i^{2[m-1]}} && (5.18) \end{aligned}$$

Note that the expression $\sigma_i^{2[m-1]}$ represent for the square of the previous standard deviation, i.e. $\sigma_i^{2[m-1]} = (\sigma_i^{[m-1]})^2$.

The optimal value of $\nu_\mu^{[m]}$ can be accessed by letting the derivative of the equation equal zero, and we get

$$\nu_\mu^{[m]} = \frac{\sum_{i=1}^{n} \left(\hat{h}_\mu^{[m]}(x_{ij^*})\right)^2}{\sum_{i=1}^{n} \frac{(\hat{h}_\mu^{[m]}(x_{ij^*}))^2}{\sigma_i^{2[m-1]}}} \tag{5.19}$$

and this expression (5.19) is the analytical ASL of μ in Gaussian distribution with respect to the negative log-likelihood loss.

It is obviously, that the $\nu_\mu^{[m]}$ is not an independent parameter in GAMLSS but depend on the base-learner $\hat{h}_\mu^{[m]}(x_{ij^*})$ with respect to the best performing variable x_{j^*} and the variance in the previous iteration $\sigma_i^{2[m-1]}$. If the GAMLSS is replaced with an usual GLM, i.e. the scale parameter σ is constant and not longer of interest, i.e. $\sigma_i^{[m-1]} = \sigma, \forall i \in \{1, \cdots, n\}$ and $\forall m \in \{1, \cdots, M\}$, so we get

$$\nu_\mu^{[m]} = \frac{\sum_{i=1}^n \left(\hat{h}_\mu^{[m]}(x_{ij^*})\right)^2}{\frac{1}{\sigma^2}\sum_{i=1}^n (\hat{h}_\mu^{[m]}(x_{ij^*}))^2} = \sigma^2. \tag{5.20}$$

This equation provides an attractive property of the ASL, that is the analytical ASL for μ in Gaussian distribution concerning the GLM is constant and actually the variance.

Back to the GAMLSS situation, now we make the scale parameter no longer a constant value but varies according to the individuals. For a linear base-learner $h(\cdot)$, the slope of $\hat{h}_\mu^{[m]}(x_{ij^*})$ will converge to zero for $m \to \infty$, which means that for all $i \in \{1, \cdots, n\}$ the predicted values of the base-learner $\hat{h}_\mu^{[m]}(x_{ij^*})$ are almost the same, moreover, these values can be represented by a constant, i.e. for example $h = \hat{h}_\mu^{[m]}(x_{ij^*})$, the analytical ASL turns out to be

$$\nu_\mu^{[m]} = \frac{\sum_{i=1}^n h^2}{\sum_{i=1}^n \frac{h^2}{\sigma_i^{2[m-1]}}} = \frac{nh^2}{h^2 \sum_{i=1}^n \frac{1}{\sigma_i^{2[m-1]}}} = \frac{n}{\sum_{i=1}^n \frac{1}{\sigma_i^{2[m-1]}}}, \tag{5.21}$$

which is actually the harmonic mean of $\sigma_i^{2[m-1]}$, i.e. the harmonic mean of the variance in the previous iteration. This expression is only valid when the slope of the base-learner converge to zero or m_{stop} is very large. However, a large m_{stop} often results in overfitting, and in practice, the boosting usually stops before the slope of the base-learner converges, especially for complex models. But anyway, the strong positive relationship between the variance σ^2 or $\sigma_i^{2[m-1]}$ and the ASL $\nu_\mu^{[m]}$ can be observed from either Eq.(5.20) or Eq.(5.21).

Except for the convergence of the base-learner, to what extent the adaptive step length can be approximated through the harmonic mean also depends on how significant the variance of the response is. As the denominator is the sum of the fraction of the base-learner and the variance, a small variance will make the relative effect of the base-learner on the adaptive step length considerable. Consequently, the gap between the harmonic mean of the variances and the real adaptive step length will be kept in the long term. On the other hand, if the variance is very big, even though the base-learner might haven't converged yet, the final adaptive step length will be dominated by the variance.

Let's focus on the Eq.(5.19) again. The analytical ASL depends, not only on the variance of the response variables but also the base-learner concerning the best fitting j^*-te covariable. The adaptive step length in an iteration is, in fact, the ASL of the j^*-te covariable in that iteration. So it is worthy of analysing the effect of the best fitting variable on the ASL.

Given only one standardized predictor variable x_1. Assume that the location parameter is generated by

$$\mu = \alpha_1 x_1, \tag{5.22}$$

whereas the scale parameter is generated by either

$$\sigma = \beta_1 x_1 \quad \text{or} \quad \sigma = \beta_2 x_1, \tag{5.23}$$

and pick arbitrary $\beta_1 < \beta_2$. Moreover, assume that the predicted base-learner in m-te iteration is

$$\hat{h}_\mu^{[m]}(x_{i1}) = \hat{\alpha}_1 x_{i1}. \tag{5.24}$$

The variance in previous iteration can be transformed into the exponential display and be approximated with the power series to the first order, i.e.

$$\sigma^{2[m-1]} = \exp(2(\hat{\beta}_k x_1)) \approx 1 + 2\hat{\beta}_k x_1, \tag{5.25}$$

where $k \in \{1, 2\}$. Thus the denominator in Eq.(5.19) for x_1 can be rewritten as

$$\sum_{i=1}^{n} \left(\frac{\hat{h}_\mu^{[m]}(x_{i1})}{\sigma_i^{2[m-1]}} \right)^2 \approx \sum_{i=1}^{n} \left(\frac{\hat{\alpha}_1 x_{i1}}{1 + 2\hat{\beta}_k x_{i1}} \right)^2 \tag{5.26}$$

$$= \sum_{i=1}^{n} \left(\frac{1}{\frac{1}{\hat{\alpha}_1 x_{i1}} + \frac{2\hat{\beta}_k}{\hat{\alpha}_1}} \right)^2 := denom(\hat{\beta}_k) \tag{5.27}$$

Obviously, the coefficients $\hat{\beta}_k$ of the predictor for the scale parameter have a negative effect on the denominator of the analytical ASL, and of course positive effect on the analytical ASL. So under the same condition, if $\beta_1 < \beta_2$, then $denom(\hat{\beta}_1) > denom(\hat{\beta}_2)$, but the step length $\nu_\mu(\hat{\beta}_1) < \nu_\mu(\hat{\beta}_2)$. In other words, given a set of variables, that affect both distribution parameters in Gaussian distribution, if the coefficient of a variable is larger than the others, then its adaptive step length for location parameter should also larger than the others.

The validity of this statement is established on a potential assumption that $\hat{\beta}_k$ is approximately equal to the actual β_k. This assumption in practice means that enough number of iterations have been performed or the scale parameter is already relatively good fitted.

5.3.2 ASL for σ

The next step is to analyze the existence of the analytical ASL for the scale parameter σ in GAMLSS with respect to Gaussian distribution and its properties.

Analogously, the optimal step length can be obtained by minimizing the empirical risk. The difference to Eq.(5.17) is the index of iterations. In the optimization problem of ν_μ, the distribution parameters used for calculating the empirical risk is $\{\mu_i^{[m]}, \sigma_i^{[m-1]}\}$, because the ASL is obtained in the current iteration and will be used for updating the location parameter μ. Now it is the opposite situation, and the ASL will be achieved by updating the scale parameter in the current iteration, the existing location parameter, however, is that of the previous iteration. So we have

$$\nu_\sigma^{[m]} = \underset{\nu_\sigma^{[m]}}{\arg\min} \sum_{i=1}^{n} \rho(y_i, \{\mu_i^{[m-1]}, \sigma_i^{[m]}\}) \tag{5.28}$$

$$= \underset{\nu_\sigma^{[m]}}{\arg\min} \sum_{i=1}^{n} \left(\hat{\eta}_\sigma^{[m-1]} + \nu_\sigma^{[m]} \hat{h}_\sigma^{[m]}(x_{ij^*})\right) + \sum_{i=1}^{n} \frac{\left(y_i - \mu_i^{[m-1]}\right)^2}{2\exp\left(2\hat{\eta}_\sigma^{[m-1]} + 2\nu_\sigma^{[m]} \hat{h}_\sigma^{[m]}(x_{ij^*})\right)} \tag{5.29}$$

After examining the positivity of the second order derivative of the expression in Eq.(5.29), the optimal value can be accessed by letting the first order derivative equal to zero, and we get

$$\sum_{i=1}^{n} \hat{h}_\sigma^{[m]}(x_{ij^*}) - \sum_{i=1}^{n} \frac{\left(\hat{h}_\sigma^{[m]}(x_{ij^*}) + \epsilon_{\sigma i} + 1\right) \hat{h}_\sigma^{[m]}(x_{ij^*})}{\exp\left(2\nu_\sigma^{[m]} \hat{h}_\sigma^{[m]}(x_{ij^*})\right)} \overset{!}{=} 0 \tag{5.30}$$

Unfortunately, the Eq.(5.30) cannot be further simplified, which means that such an analytical ASL for the scale parameter σ in Gaussian distribution does not exist. Hence, the optimal ASL must be found by performing the conventional line search, for example, the Brent's method as discussed above. For more details about the induction, see Appendix A.3.

Even if we could not find an analytical solution, we can still find an interesting property by further studying the Eq.(5.30). Just like the induction of Eq.(5.21), the slope of the linear base learner $\hat{h}_\sigma^{[m]}(x_{ij^*})$ will converge to zero for $m \to \infty$. Analogously, the predicted values of which thus could be replaced with a constant, i.e. $h = \hat{h}_\sigma^{[m]}(x_{ij^*}), \forall i \in \{1, \cdots, n\}$, then the Eq.(5.30) turns out to be

$$\sum_{i=1}^{n} \hat{h}_\sigma^{[m]}(x_{ij^*}) - \sum_{i=1}^{n} \frac{\left(\hat{h}_\sigma^{[m]}(x_{ij^*}) + \epsilon_{\sigma i} + 1\right) \hat{h}_\sigma^{[m]}(x_{ij^*})}{\exp\left(2\nu_\sigma^{[m]} \hat{h}_\sigma^{[m]}(x_{ij^*})\right)} \overset{!}{=} 0 \tag{5.31}$$

$$\Leftrightarrow \sum_{i=1}^{n} h - \sum_{i=1}^{n} \frac{(h + \epsilon_{\sigma i} + 1)h}{\exp(2\nu_\sigma^{[m]} h)} = 0 \tag{5.32}$$

$$\Leftrightarrow \nu_\sigma^{[m]} = \frac{1}{2h} \log\left[h + 1 + \frac{1}{n}\sum_{i=1}^{n} \epsilon_{\sigma i}\right] \tag{5.33}$$

$$\Leftrightarrow \nu_\sigma^{[m]} = \frac{1}{2h} \log[h + 1] \tag{5.34}$$

where $\frac{1}{n}\sum_{i=1}^{n} \epsilon_{\sigma i} = 0$ in the simple linear regression. The expression in Eq.(5.34) can be further simplified by approximating the logarithm function with the Taylor series at $h = 0$, thus

$$\nu_\sigma^{[m]} \approx \frac{1}{2h}\left[h - \frac{h^2}{2} + O(h^3)\right] \tag{5.35}$$

$$= \frac{1}{2} - \frac{h}{4} \tag{5.36}$$

Theoretically, for $m \to \infty$, it is not only the slope of the linear base-learner that converges to zero, but also its intercept. Consequently, the previously assumed constant h should also converge to zero. As a result, the limit of the ASL for σ is constant and actually 0.5,

$$\lim_{m\to 0} \nu_{\sigma}^{[m]} = \lim_{h\to 0} \frac{1}{2} - \frac{h}{4} = \frac{1}{2}. \tag{5.37}$$

This gives a new property about the ASL for σ, i.e. no matter how the data is organized, the ASL will always converge to 0.5. Of course, considering the situation of overfitting, this value may not appear, as the boosting algorithm usually stops before convergence, but the trend that the ASL go towards 0.5 should be clearly.

5.3.3 Semi-Analytical ASL

As discussed above, the analytical ASL in GAMLSS can only apply to the location parameter μ with respect to Gaussian distribution, while for the scale parameter σ it needs to be found by doing a line search. For those ASLs for each distribution parameter in GAMLSS, that partial determined by the analytical solution and partial searched by performing the line search, we can call it the *Semi-Analytical Adaptive Step Length*, or *Semi-Analytical ASL* or just *SAASL*.

By applying the semi-analytical method to the optimizing problem, the time complexity of partial distribution parameters can be reduced to constant. Thus, the computation with the SAASL for the whole learning process will be faster than that with the conventional ASL.

The ASL method together with the line search can solve the problem of unbalanced decision making on a large scale. But as it requires an interval, in which the minimum is located, the boundaries of the interval needs to be carefully selected. If the interval is tiny, the local minimum might not be situated in it. But if it is tremendous, the unnecessary searching region will cause additional computing time. In general, the ASL at the beginning iterations should be big, and it will then decrease to a reasonable range because the estimation of its coefficients becomes better with the increasing boosting iterations. So it is not suggested to set a long interval. But the relatively small interval will, on the other hand, make the unbalanced decisions, as the step length is not large enough for the improvements of some variables, and these variables need to be more frequently selected until the improvements decreased to a level that other variables have. Moreover, after sufficient iterations have been performed, the adaptive step length for μ of a variable is also affected by its coefficient on σ (see Eq.(5.27)), if this coefficient is vast, it results in a sizeable adaptive step length for μ, and might larger than the upper boundary of the interval. It will also affect decision making.

However, this problem can be partially solved by the SAASL, as the limitations of the interval don't restrict the step length for μ calculated by this method. Though there exists not an analytical solution to the adaptive step length for σ, it is easier to set an interval as we have seen its convergence point and this point is unrestricted to which variable is used.

Even if the SAASL approach results in better computing efficiency as ASL, the line search for finding the ASL for σ is still annoying. As we have known the limit of $\nu_\sigma = 0.5$, an aggressive approach will be making this value as the step length for σ, just like the fixed step length. Accounting for the shrinkage parameter $\lambda = 0.1$, the real used step length in each iteration shall be 0.05. Thus, a new method can be established when updating the predictors in Gaussian distribution, i.e. update the location parameter μ with the analytical adaptive step length, and update the scale parameter σ with the fixed step length (0.05). And we call this approach the SAASL05.

The advantage of the SAASL05 mainly lies in the computing speed, as it removes the line search procedure. As the step length for σ is not the adaptive value, the balance of decisions should not be as well as the SAASL or the ASL, but should much better than the FSL. In contrast with the FSL, whose step length for σ is set as 0.1 according to the experience, the SAASL05 provide a more reasonable fixed step length 0.05 according to the mathematical induction. Comparing these two values, it is apparent, that 0.1 is an aggressive value, and theoretically, it might induce the overfitting.

Now, we formally present the algorithm of the non-cyclical componentwise gradient boosting in GAMLSS for Gaussian distribution with four types of step length, i.e. fixed step length (FSL), adaptive step length (ASL) and two semi-analytical step length (SAASL, SAASL05) approach, see Algorithm 7.

Algorithm 7 Non-cyclical componentwise gradient boosting in Gaussian distribution with different step lengths

```
1: Initialize the additive predictors $\hat{\eta}^{[0]} = (\hat{\eta}_\mu^{[0]}, \hat{\eta}_\sigma^{[0]})$ with offset values.
2: For each distribution parameter specify a set of base-learners: $\{h_{\mu 1}(\cdot), \cdots, h_{\mu J_\mu}(\cdot)\}$ and
   $\{h_{\sigma 1}(\cdot), \cdots, h_{\sigma J_\sigma}(\cdot)\}$, where $J_\mu$ and $J_\sigma$ are the cardinality of the set of base-learners specified
   for $\mu$ and $\sigma$.
3: for $m = 1$ to $m_{\text{stop}}$ do
4:     for $\theta = \mu$ to $\sigma$ do
5:         Compute negative partial derivatives $-\frac{\partial}{\partial \eta_\theta}\rho(y, \eta)$ and plug in the current estimates
           $\hat{\eta}^{[m-1]}(\cdot)$:
               $$u_\theta = \left( -\frac{\partial}{\partial \eta_\theta}\rho(y, \eta)\Big|_{\eta=\hat{\eta}^{[m-1]}(x^{(i)}), y=y^{(i)}} \right)_{i=1,\cdots,n}$$
6:         Fit each of the base-learners $h_{\theta j}(\cdot)$ contained in the set of base-learners specified for
           the distribution parameter $\theta$ in step (2) to the negative gradient vector $u_k$.
7:         Select the best-fitting base-learner $h_{\theta j^*}$ by the inner loss, i.e., the residual sum of
           squares of the base-learner fit w.r.t. $u_k$:
               $$j^* = \underset{j \in 1, \cdots, J_k}{\arg\min} \sum_{i=1}^{n} \left( u_k^{(i)} - \hat{h}_{\theta j}(x^{(i)}) \right)^2$$
8:         Set or find the step length $\nu_\theta^{[m]}$ by one of the followings:
           • Fixed step length (FSL), for example $\nu_\theta^{[m]} = 0.1$;
           • Adaptive step length (ASL):
               $$\nu_\theta^{[m]} = \underset{\nu}{\arg\min} \sum_{i=1}^{n} \rho \left( y^{(i)}, \hat{\eta}_\theta^{[m-1]}(x^{(i)}) + \nu \cdot \hat{h}_{\theta j^*}(x^{(i)}) \right);$$
```

```
          • Semi-analytical adaptive step length (SAASL):
            if $\theta = \mu$,
                $\nu_\mu^{[m]} = \frac{\sum_{i=1}^n \left(\hat{h}_{\mu j^*}(x^{(i)})\right)^2}{\sum_{i=1}^n \frac{(\hat{h}_{\mu j^*}(x^{(i)})^2}{\sigma_i^{2[m-1]}}}$,
            if $\theta = \sigma$, same with ASL.
          • Semi-analytical adaptive step length (SAASL05):
            if $\theta = \mu$, same with SAASL,
            if $\theta = \sigma$, $\nu_\theta^{[m]} = 0.5$.
 9:         Compute the possible improvement of this update regarding the outer loss
                $\Delta\rho_k = \sum_{i=1}^n \rho\left(y^{(i)}, \hat{\eta}_\theta^{[m-1]}(x^{(i)}) + \nu_\theta^{[m]} \cdot \hat{h}_{\theta j^*}(x^{(i)})\right)$
10:     end for
11:     Update, depending on the value of the loss reduction $\theta^* = \arg\min_{\theta \in \{\mu,\sigma\}}(\Delta\rho_\theta)$ only the
      overall best-fitting base-learner:
                $\hat{\eta}_{\theta^*}^{[m]} = \hat{\eta}_{\theta^*}^{[m-1]} + \nu_\theta^{[m]} \cdot \hat{h}_{\theta j^*}(x)$
12:     Set $\hat{\eta}_\theta^{[m]} := \hat{\eta}_\theta^{[m-1]}$ for all $\theta \neq \theta^*$.
13: end for
```

Note that the approaches listed in the Algorithm 7 didn't take the shrinkage parameter λ into consideration. As introduced in Section 3.3.1, there is no need to specify the shrinkage parameter for the FSL, but we need to multiply it to all adaptive step lengths. As the value 0.5 in SAASL05 is an adaptive value, it should also be shrunk to 0.05 if the $\lambda = 0.1$.

6 Simulation Study

After discussing the theories of boosted GAMLSS with various step length settings, we can evaluate the performance of each setting by performing the simulations. The first part of this section listed the to be compared algorithms and the model settings. And the second part examines the performance of each algorithm by various measures, including the prediction accuracy, the balance of the distribution parameters, the runtime or computational costs. Some unique properties, like optimal step length in GLM (see Eq.(5.20)), are also shown with graphics.

6.1 Simulation settings

6.1.1 Computational environment

As the simulations are very computing expensive, the tasks of simulations are parallelly computed with multiple CPUs (Intel Xeon CPU E7-4860 2.27GHz) on the R program (version 3.5.1).

6.1.2 Algorithm settings

The algorithm used for fitting the data is the non-cyclical componentwise gradient boosting in GAMLSS algorithm (see Algorithm 7) with all four types of step length. As the step length in adaptive methods is searched or computed automatically, no special settings are required for these two algorithms. Only the step length in FSL needs to be specified artificially, so we set it with the recommended value of 0.1 [Bühlmann and Hothorn, 2007].

To get a better understanding of the performance and properties of each type of step length, we added another algorithm into the comparison, which implemented in the R package `gamboostLSS` (i.e. *Boosting methods for "GAMLSS"* [Hofner et al., 2018]). The method used in this package is just the Algorithm 5 with the inner loss for selecting the best performing base-learner in each iteration. As it is also a fix step length situation, we set it with 0.1 as usual. Consequently, the comparison will be among the five algorithms:

- FSL: Fix step length ($\nu = 0.1$),
- ASL: Adaptive Step Length,
- SAASL: Semi-Analytical Adaptive Step Length,
- SAASL05: Semi-Analytical Adaptive Step Length with fixed $\nu_\sigma = 0.5$,
- gamboostLSS: Boosting methods for GAMLSS.

For the adaptive methods, an additional shrinkage parameter $\lambda = 0.1$ is multiplied to the optimal values.

The initial values for m_{stop} is 1000. The graphics below, which are used for illustration, displayed all 1000 iterations. However, the graphics, that aims at comparisons illustrated the corresponding metrics learned until the optimal stopping value specified by 10-folds CV.

6.1.3 Data and model description

The p explanatory variables $x_{ij}, i \in \{1, \cdots, n\}, j \in \{1, \cdots, p\}$ are drawn from the uniformly distribution, where n is the number of observations,

$$x_{ij} \sim \text{Uni}(-1, 1),$$

and the response variable $y_i, i \in \{1, \cdots, n\}$ is drawn from the Gaussian distribution

$$y_i \sim N(\mu_i, \sigma_i),$$

where μ_i and σ_i depends on:

- Scenario 1: only informative variables, and
- Scenario 2: not only informative, but also non-informative variables.

For each scenario, the models under 5 different cases are considered for the comparison:

- Balanced case A: in this case, the informative variables are identical for both distribution parameters,

$$\begin{aligned} \mu_i &= 1 + 2x_{i1} - x_{i2} - 3x_{i3} \\ \sigma_i &= \exp(1 - 0.5x_{i1} + x_{i2} + 2x_{i3}), \end{aligned}$$

- Unbalanced case B: the informative variables are partially identical or not identical for each distribution parameter,

$$\begin{aligned} \mu_i &= 1 + 2x_{i1} - x_{i2} - 3x_{i3} \\ \sigma_i &= \exp(1 - 0.5x_{i1} + x_{i2}), \end{aligned}$$

- Correlated case C: the informative variables are correlated,

$$\begin{aligned} \mu_i &= 1 + 2x_{i1} - x_{i2} - 3x_{i3} \\ \sigma_i &= \exp(1 - 0.5x_{i1} + x_{i2}), \end{aligned}$$

  where, all variables here are correlated, and their correlation is $\rho = 0.5$. For example, in scenario 1, only three explanatory variables are available, and they are correlated with each other, while in scenario 2, the correlation is active not only among the informative variables but also between the informative and non-informative variables. In other words, each variable is correlated with all the others.

- Extreme μ case D: in this case, the coefficients of some variables for μ are significantly larger or smaller than that of the others. By this way, the balance of selected distribution parameters in all iterations can be examined. In other words, exploring if some distribution parameters are chosen frequently than the others,

$$\begin{aligned} \mu_i &= 1 + 100x_{i1} - x_{i2} - 3x_{i3} \\ \sigma_i &= \exp(1 - 0.5x_{i1} + x_{i2}) \end{aligned}$$

- Extreme σ case E: similarly, in this case, the coefficients of x_{i2} is significantly differing from that of x_{i2}, even if it is not so different as in extreme μ case. That is because the σ is exponential to the linear predictor, small changes in the linear predictor can induce a vast difference in σ,

$$\mu_i = 1 + 2x_{i1} - x_{i2} - 3x_{i3}$$
$$\sigma_i = \exp(1 - 0.5x_{i1} + 5x_{i2})$$

By this way, ten different types of data sets in total are generated by associating the scenarios and cases. For simplicity, we use for example 1B, 2D (and so on) as the indices for each simulation.

The size of observations in each simulation is 1000. For the simulations with non-informative variables, i.e. in scenario 2, the total number of variables used for modelling is 100. The number of the informative variables of which varies according to the cases.

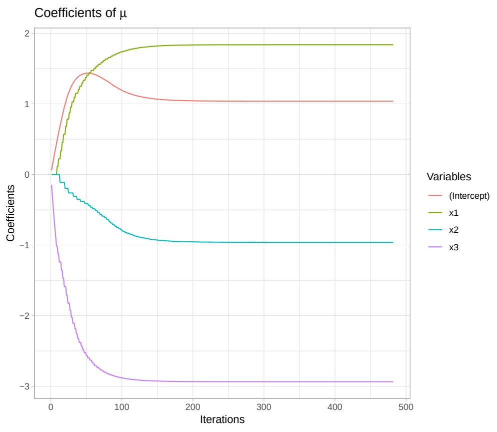
6.2 Results

As all algorithms are in fact variations of the non-cyclical componentwise gradient boosting, the performance of each algorithm is endogenous to the settings of its step length. Different data models will not influence their properties. So for simplicity, we use the unbalanced model with non-informative variables, i.e. the type 2B model, as the primary data model for the comparison and interpretation. For those who use other types of data models, we will describe them particularly.

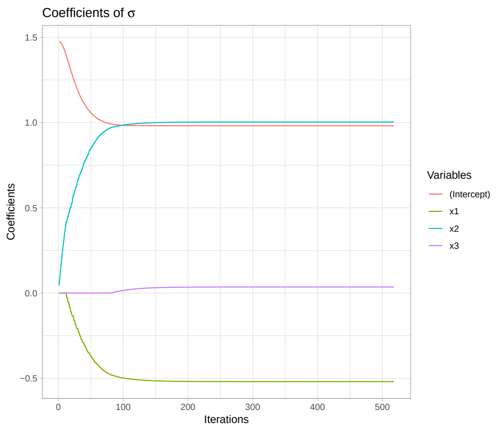
6.2.1 Overview

To roughly understand the non-cyclical componentwise boosting algorithm, the graphics in this part will present an overview of the key outputs.

Firstly, the most important outputs of the modelling are the estimated coefficients of each distribution parameter. Figure 1 showed the coefficients of μ and σ in all 1000 iterations with type 1B model (i.e. the data contains only the informative variables) using the ASL algorithm. 482 out of 1000 iterations are applied for updating the location parameter μ, while 518 for updating the scale parameter σ. At the beginning of boosting, as the coefficients of each variable are not good enough fitted, the algorithm will wrongly be thought that there was a correlation between the intercept and the other variables. The coefficient of intercept thus increases to a high value (e.g. at around 50 iterations for μ) and then decreases to a reasonable amount. Similar behaviour also exists for σ. After all updates have been performed, the estimated coefficients for both distribution parameters are acceptable.



(a) μ



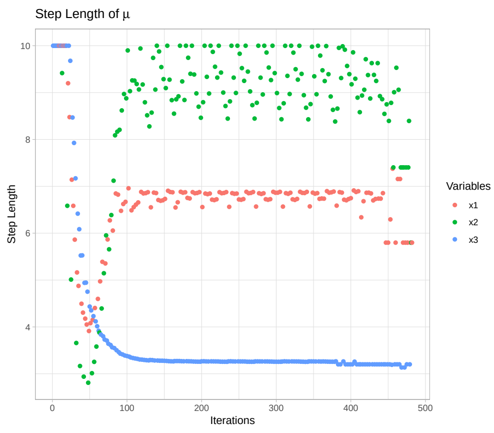
(b) σ

Figure 1: Estimated coefficients of μ and σ in each iterations of the data type 1B using ASL algorithm. Among 1000 iterations, 482 and 518 iterations are used to update μ and σ respectively.

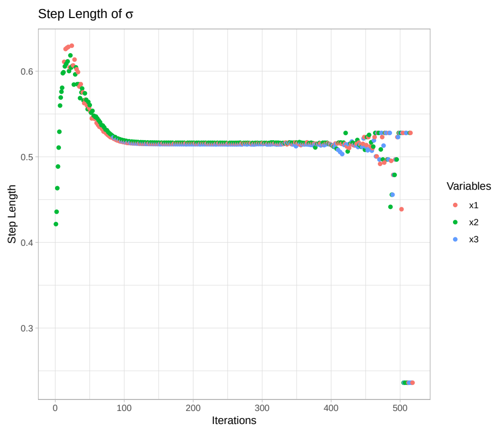
Another important output is the step length in each iteration. Figure 2 illustrated the step length in each iteration for both distribution parameters. As a line search requires an interval, in which the golden section algorithm (see Algorithm 6) or Brent's method can be performed, the step lengths for μ at the beginning are always 10, which is also the upper boundary for ν_μ. For more details, see the descriptions of Figure 2.

As discussed in Eq.(5.27), there is a positive relationship between the coefficients of the predictor of σ and the adaptive step length ν_μ as long as enough number of iterations have been performed. This relationship can be clearly observed from the Figure 2a. The true coefficient in the predictor of σ for x_{i2} is 1 which is larger than 0.5 of x_{i1}. Consequently, the adaptive step lengths $\nu_\mu(x_{i2})$ is larger in general than $\nu_\mu(x_{i1})$. Similarly, the variable x_{i3} have no effect on σ, the adaptive step length $\nu_\mu(x_{i3})$ is thus the smallest.

The property that the adaptive step length for σ tends to converge to 0.5 can be observed from Figure 2b. It is clear that no matter which variable is observed its adaptive step length converges to the 0.5. After a long term rounding 0.5, the ν_σ begins to diverge, similar phenomenon exists also for ν_μ. Although no evidence can be provided, we strongly believe that is caused by the computational accuracy.



(a) μ



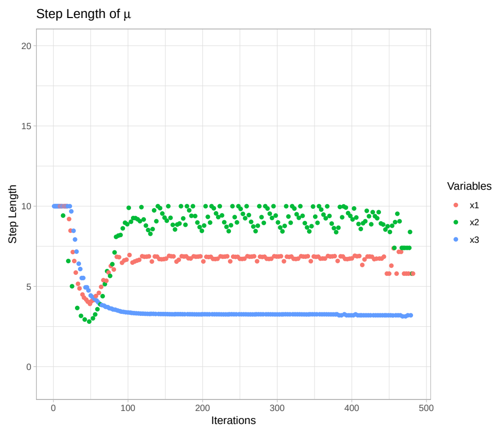
(b) σ

Figure 2: Step length of μ and σ in each iterations of the data type 1B model using ASL algorithm. The searching intervals for each distribution parameter are $\nu_\mu \in [-1, 10]$ and $\nu_\sigma \in [-1, 1]$ respectively.

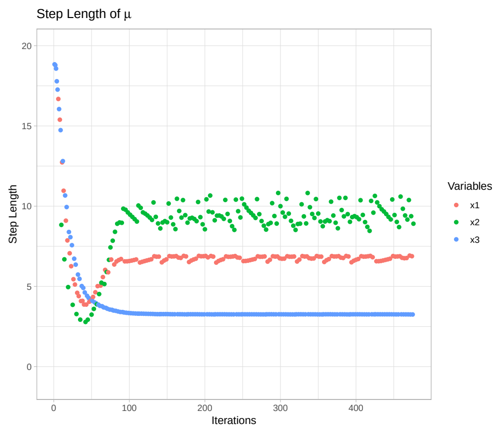
For comparison, Figure 3 illustrated the step length for μ computed by ASL and SAASL. Theoretically, the ASL and SAASL should result in the same outputs, because both methods try to find the optimum from a unimodal function. But as discussed above, a searching interval must be predefined in ASL. If the true optimal step length exceeds the predefined interval, the outputs of ASL and SAASL then can be different. And the results in Figure 3 are just this kind of case.

As an analytical expression finds the step length in SAASL instead of a line search procedure with a searching interval, the optimal step length in each iteration is no longer restricted. It can be found from the graphic, that the step length $\nu_\mu(x_{i2})$ at the beginning iterations as well as the at some points over the iterations are larger than 10, which is, as previously described, the upper boundary of the searching interval.

Recall the positive relationship between the coefficients $\beta_\sigma(x_j)$ and the step length $\nu_\mu(x_j)$, for those variable x_j, whose $\beta_\sigma(x_j)$ is larger than usual, or the coefficients for μ (i.e. $\beta_\mu(x_j)$) is directly extremely large, the step length $\nu_\mu(x_j)$ will always be 10 when using the ASL approach and 10 is set as the upper boundary. Although a continuous sequence of step length reaching the upper boundary is not problematic, it is not as efficient as the method without a boundary. Under extreme circumstances, for example, the simulation with type D or even worse, the boosting algorithm will select the same variable over all iterations if not enough number of iterations is pre-specified. A solution is to increase the searching boundary. However, this will make the searching time much longer and is not necessary for those whose coefficients are not extraordinary large or small. As a result, the boosting iterations used for updating μ is different from each other. The used iterations in ASL are 482, and the one in SAASL is 476. Though the difference between these two values is not significantly, fewer iterations are required by SAASL method as expected. Strong evidence will be provided in the following section with the simulation type D.



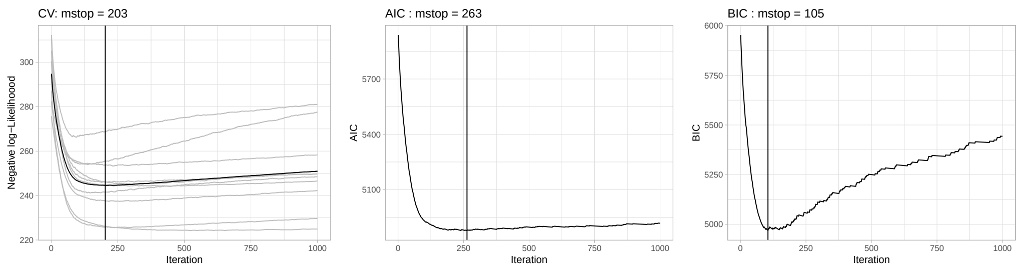
(a) ASL



(b) SAASL

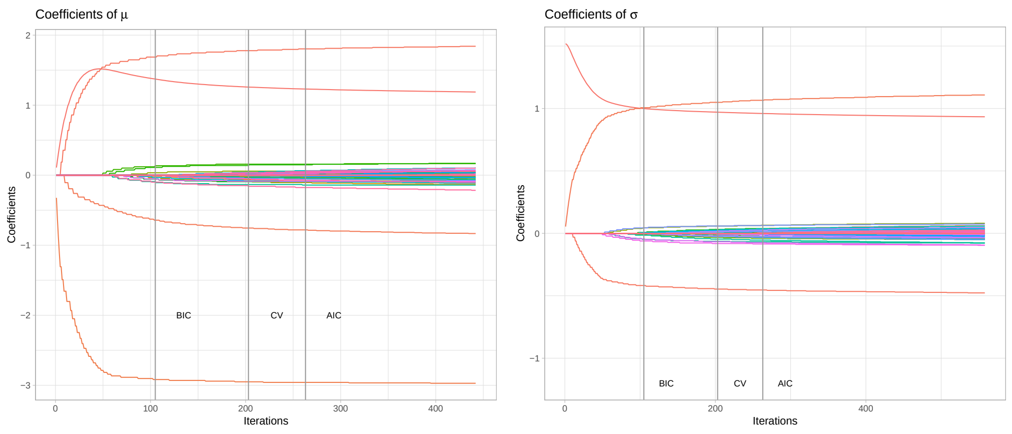
Figure 3: Comparison of the ASL and SAASL with the data type 1B. The upper searching boundary for ASL is 10. The boosting iterations used for updating μ are 482 and 476 for ASL and SAASL respectively.

The last outputs that deserve to have a look are the early stopping values. Figure 4 demonstrated the stopping values specified by 10-folds cross-validation, AIC and BIC using SAASL method. The data model used for illustration is 2B, i.e. the variables containing also the non-informative ones. As the number of observation is 1000 and it is a relatively large value, the BIC penalize more heavily. It tends to stop earlier than the position determined by CV and AIC. By comparing the coefficients at each stopping values in Figure 5, it can be found that the estimated coefficients for the informative variables at the stopping value specified by BIC rather underfitted. Meanwhile, it selected fewer false positive variables. On the other hand, stopping with AIC result in a better estimation of the informative variables at the cost of more non-informative included in the model. And CV stands at the compromised position between the other two. The comparison of the performance among the five methods below uses the 10-folds cross-validation as the stopping criterion, and all results are the values at the early stopping position.



(a) CV (b) AIC (c) BIC

Figure 4: Demonstration of early stopping. The data model type is 2B, i.e. 100 variables are simulated, 3 and 2 of which are informative for μ and σ respectively. The data are modelled with the SAASL algorithm. The cross validation is 10-folds.



(a) μ (b) σ

Figure 5: Demonstration of early stopping method on coefficients plot. The data and algorithm situation are same with Figure 4.

6.2.2 Accuracy

The estimation accuracy of a model can be measured by the mean squared error or directly comparing the difference between the estimated values and the true ones.

Figure 6 illustrated the estimated coefficients of the predictors for both distribution parameters from 100 simulations with data type 2B using five different algorithms. The estimated coefficients are the values at the early stop position specified by the 10-folds CV. The black horizontal line indicates the actual values of each variable. The closer a box to the black line, the more accrue is the corresponding method.

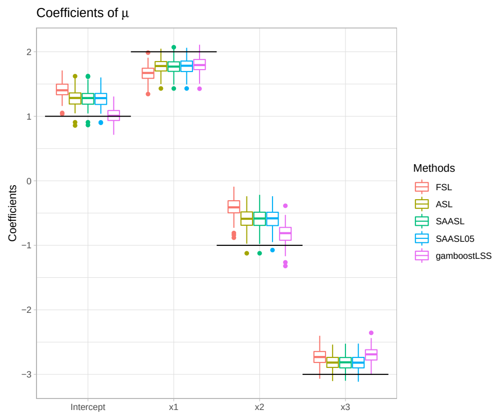
It can be observed from the graphic, that the ASL, SAASL and SAASL05 have similar accuracy. It is reasonable, as all these three methods use the adaptive step length to fit the model. The searching method of ν_σ in ASL is same with SAASL, and the ν_μ in SAASL and SAASL05 is calculated with the same mathematical expression. The FSL method, especially for μ, performed not as well as the other methods because of underfitting. The can be explained more clearly by observing the number of false positive variables in Figure 14. Until the stopping position, the FSL methods selected less non-informative variables, which, on the other hand, not good enough fitted the coefficients. The gamboostLSS method can reasonably estimate the coefficients of μ for some variables, e.g. for Intercepts and x_{i2}, but it performed a little worse than the adaptive methods, for example, the coefficients of x_{i3}. It is a little hard to understand the difference of the accuracy between FSL and gamboostLSS, as both methods using the noncyclical boosting algorithm with fixed step length $\nu = 0.1$. A plausible reason is that the stopping values specified by CV are random. From the Figure 14 we can find that both methods selected little false positive variables, a small change of the stopping values will add more false positive variables into the model and further influence the estimations. Another probable reason is the internal settings of the programs. The program of FSL is a fixed step length version of the adaptive step lengths methods (include ASL, SAASL and SAASL05), but not a rewrite of gamboostLSS.

Thus, the following points can be summarized in Figure 6:

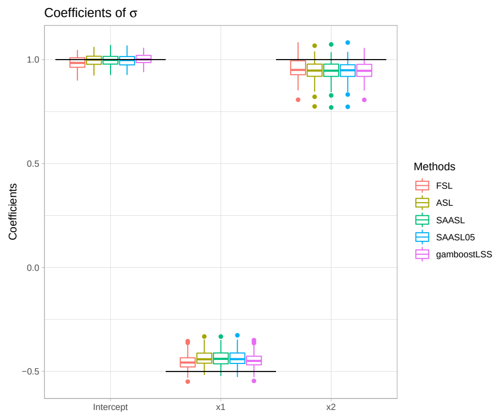
1. How to get the optimal step length in each iteration between ASL and SAASL have little effect on the estimated coefficients.

2. Choosing the aggressive constant step length $\nu_\sigma = 0.5$ in SAASL05 will not result in unacceptable estimations.

3. Although FSL underestimated the coefficients in this example, it is hard to say that FSL tends to underfit, as both FSL and gamboostLSS are fixed step length method and the latter estimated relatively better.



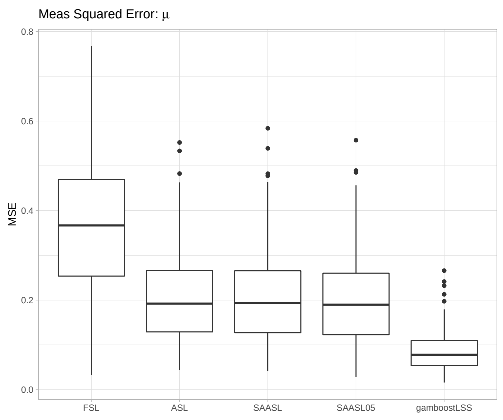
(a) μ



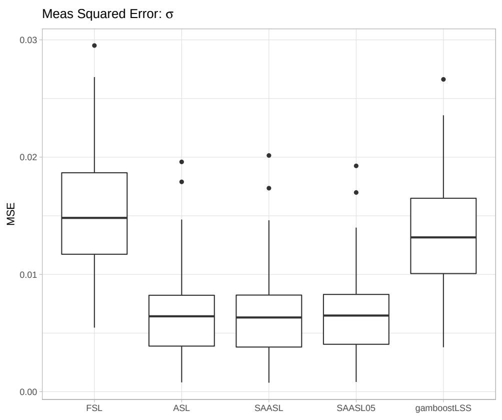
(b) σ

Figure 6: Comparison: Estimated coefficients of distribution parameters μ and σ. The data with type 2B is fitted with 5 noncyclical componentwise boosting methods. The black solid horizon line indicate the true coefficients of each variable.

Another method used to measure accuracy is the mean squared error (MSE). Figure 7 illustrated the MSE of both distribution parameters in all 100 simulations. By looking at the MSE of μ, gamboostLSS has the lowest values and thus the best estimations of the coefficients. ASL, SAASL and SAASL05 have no surprises the similar performance at a relatively worse level than gamboostLSS. However, this situation is reversed by observing the MSE of σ. It means that, in this example, though the adaptive methods could not estimate the coefficients of the location parameter μ as well as the fix step length methods, they perform better when fitting the scale parameter σ. When comparing the three adaptive approaches with the FSL, we will find that the former win in both cases. However, the cost of the victory is more false positive variables added into the final model. As discussed above, the FSL selected relatively less false positives than the adaptive methods (see Fig. 14).



(a) μ



(b) σ

Figure 7: Comparison: Mean Square Error (MSE). Boxplot of 100 simulations with data type 2B that are fitted with 5 different boosting methods.

An overall measure of the goodness of fitting is the empirical risk, which is the negative log-likelihood in GAMLSS. The empirical risk will not evaluate the goodness of each distribution parameters separately, but combines them and measure the loss of the whole model. Figure 8 compares the empirical risk of the five noncyclical componentwise boosting algorithms at the stopping values specified by CV. According to the graphic, though the three adaptive methods have slightly higher mean values of the negative log-likelihoods than that in FSL and gamboost-LSS, the difference is not significant and thus it's hard to say which method is better than the others. Recall the comparison of MSE, though the adaptive methods are not as competitive as gamboostLSS when estimating μ, the advantage when facing σ make them in overall no worse than the fixedwise methods.

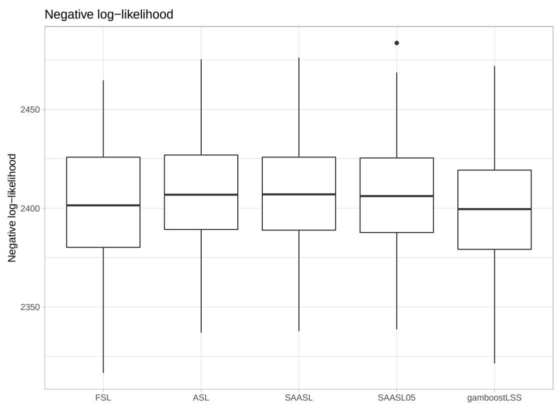


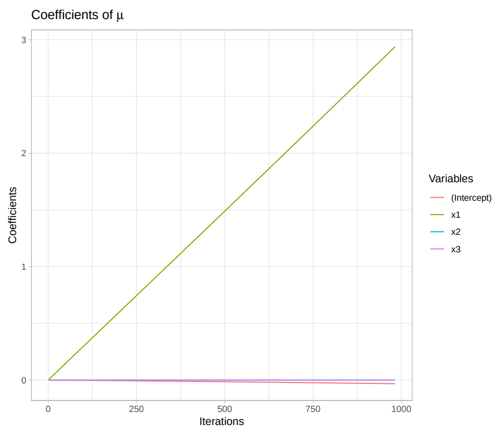
Figure 8: Comparison: Empirical risk. The empirical risk used in GAMLSS is the negative log-likelihoods. Boxes are established on 100 simulations.

6.2.3 Balance of decisions

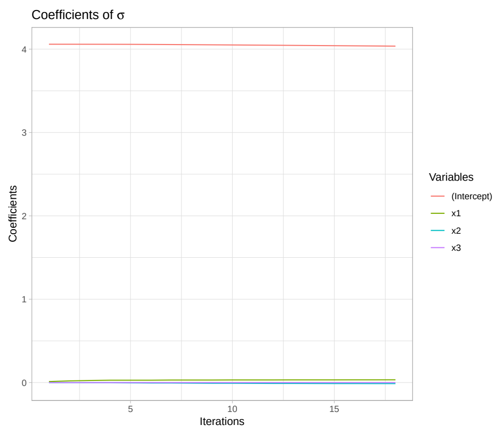
The balance of decisions is how the selected distribution parameters until the stopping values distributed. In other words, given finished n boosting iterations, the decision is balanced if the n iterations are approximated equally distributed to each distribution parameter, or the frequency of each distribution parameter used for updating the model is approximately same. Otherwise, if the boosting algorithm updates some distribution parameters more frequently than others, we say it makes unbalanced decisions. The balance concept exists only in the non-cyclical boosting algorithms, as the cyclical boosting methods update all distribution parameters in each iteration and thus the decisions are always balanced.

The balance of decisions is used to test if the computing sources are equally distributed to all parameters. Consider the boosting algorithms with a small fixed step length, if the coefficient of a variable in one of the distribution parameters is extremely large or small, the boosting algorithm will select this parameter and updating its coefficients in most or even in all iterations, as the empirical risk minimize the most by updating it, and the predictor in each iteration is updated with a small step. It, however, will be problematic, as, for example, the effect of other distribution parameters cannot be learned from the modelling at all, or despite learned, but far away from the expected values.

Such an unbalanced case can be observed in Figure 9. This example used the FSL method to fit the data type 1D, i.e. the coefficients of x_{i1} is 100, which is extremely larger than that of the other variables. Even though only 982 out of 1000 iterations are used for updating σ, the estimated coefficient of x_{i1}, which is about 3, is far away from the actual values. A solution to overcome this underfitting is to increase the boosting iterations. In this example, about 35,000 iterations are required until the coefficients are relatively good estimated. However, such a great number of iterations needs a very long runtime.



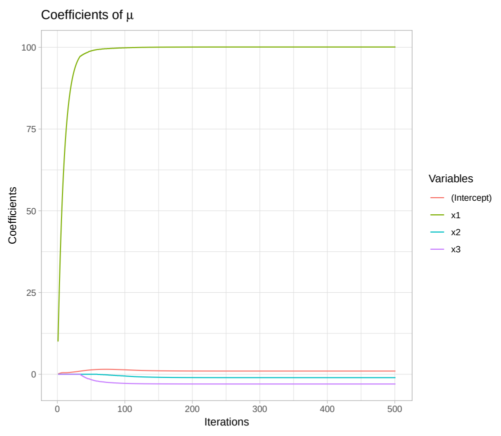
(a) μ



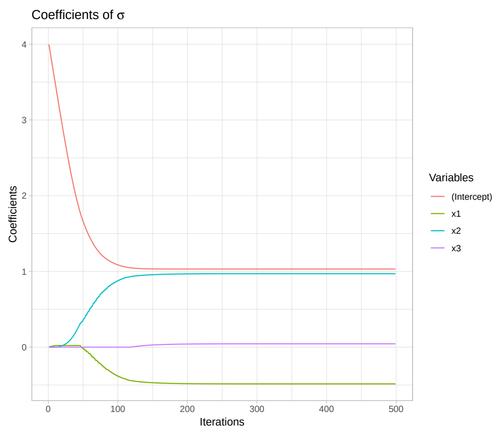
(b) σ

Figure 9: Demonstration of unbalanced decisions. Model settings: $\mu_i = 1 + 100x_{i1} - x_{i2} - 3x_{i3}$ and $\sigma_i = \exp(1 - 0.5x_{i1} + x_{i2})$. The FSL with step length 0.1 is applied to fit the data. 982 and 18 out of 1000 iterations are used to updating the distribution parameter μ and σ respectively.

As a comparison, Figure 10 illustrated the same model which is fitted with the SAASL algorithm. This time, the balance rate of the selected μ and σ in all 1000 iterations is 501:499, which can be regarded as a very well balanced case. As the step length in each iteration is adaptive, the variables whose coefficient is extremely large or small can be updated with a correspondingly step length. Thus the estimation can reach at a reasonable region within limited iterations. The rest computing sources can then be applied to find more precise values of all distribution parameters.



(a) μ



(b) σ

Figure 10: Demonstration of balanced decisions. Model settings: $\mu_i = 1 + 100x_{i1} - x_{i2} - 3x_{i3}$ and $\sigma_i = \exp(1 - 0.5x_{i1} + x_{i2})$. The SAASL method is applied to fit the data. 501 and 499 out of 1000 iterations are used to update the distribution parameter μ and σ respectively.

Figure 11 compares the balance of the 5 boosting algorithms when fitting the type 2B data. Obviously, the fixedwise methods (FSL and gamboostLSS) prefer to make unbalanced decisions, whereas the adaptive methods (ASL and SAASL) prefer to make a balanced decision. As explained above, the scale of the coefficients has a strong influence on the balance of the decisions when using the fixedwise methods. But such impact will be minimal when meeting the adaptive methods, as these methods will update the predictors with a big step. The balance in SAASL05 approach seems to be a compromise between fixedwise and adaptive methods when making decisions. On the one hand, it uses the analytical adaptive step length to update μ, which prefer the balanced ones. On the other hand, it uses the asymptotic fixed step length 0.5 to update σ, which prefer, however, the unbalanced ones. Another property of SAASL05 when making decisions is that it updates more frequently the σ, the reason for this phenomenon is that the artificial settings of the step length ν_σ cannot update the predictors correctly at the beginning iterations, whereas the predictors of μ can be updated fast within these iterations. Overall speaking, the unbalanced rate of SAASL05 is at an acceptable level.

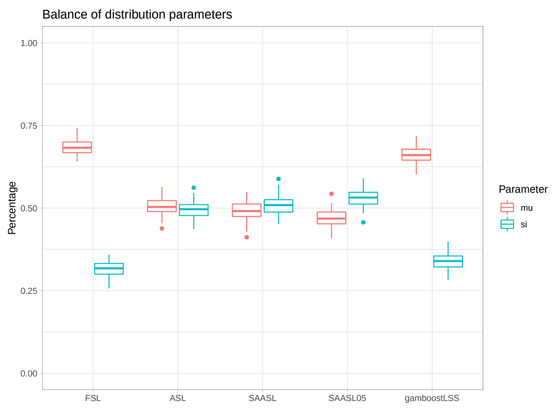


Figure 11: Comparison: Balance of decisions. The y-axis is the proportion of the decisions for each distribution parameter. 5 different componentwise boosting algorithms are applied to fit the data type 2B. The FSL and gamboostLSS make rather unbalanced decisions. The ASL and SAASL prefer to make balanced decisions. Though the SAASL05 makes a unbalanced decisions, it is quiet acceptable.

6.2.4 Computational costs

The computational costs is also an important point when evaluating an algorithm. With the same accuracy, the faster algorithm is a natural choice. Or under some circumstances, a faster algorithm is also preferred at the cost of inaccuracy as long as it is within the tolerance. The computational costs can be evaluated either by the stopping values or the computing time.

The early stopping value is actually not a good tool to measure the computational costs. As even though it shows intuitively how many boosting iterations are required to get a good estimation, and fewer iterations, of course, means a faster computation. However, the computation speed inside each iteration should also not be ignored. As discussed above, the fixed step length method requires usually more iterations than the step length found by doing a line search, but performing a line search needs more computation time. Figure 12 compares the m_{stop} specified by 10-folds CV for each boosting algorithm. As expected, the FSL and gamboostLSS need more iterations than the adaptive methods until a good estimation is met. The early stopping values among ASL, SAASL and SAASL05 have no significant differences. It is reasonable, as all methods using the adaptive step length to update the predictors and the estimations can reach at good values in little iterations. However, if we use the stopping values as the measure to evaluate the computational costs, we would tell that there are no difference between the three adaptive methods. That is apparently a wrong judgement, as ASL performs the line search, which indeed needs more computation, whereas SAASL performs only partial and SAASL05 not at all the searching algorithm.

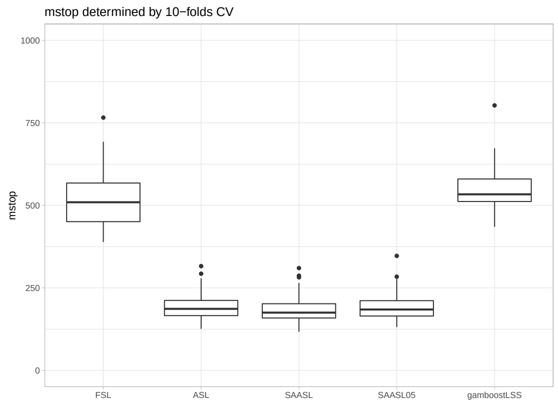
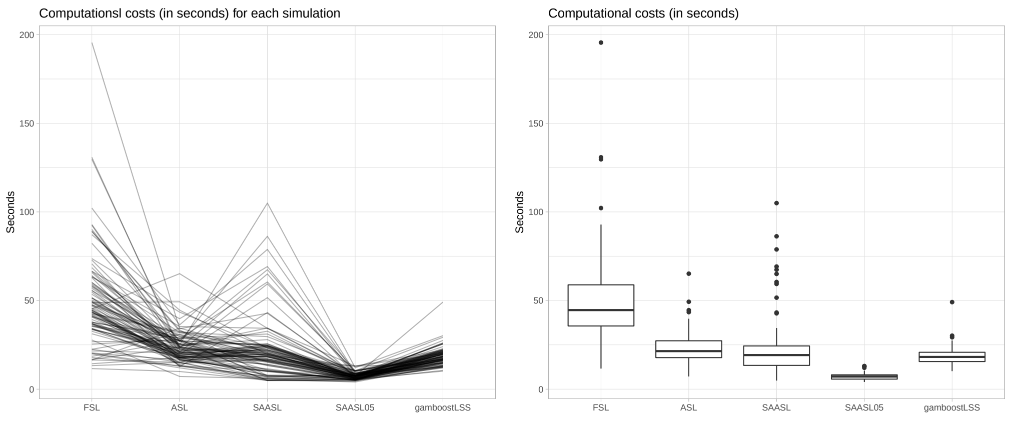


Figure 12: Comparison: m_{stop}. The m_{stop} is specified by the 10-folds CV. Until the m_{stop}, FSL and gamboostLSS need more boosting iterations, whereas the adaptive methods have the similar early stopping values.

Therefore, real-time comparison is more reasonable. Figure 13 illustrated the comparison of computing time (in seconds) among different algorithms. The left graphic showed the computing time of each simulation in all five algorithms, and the right graphic summarized them up and provide a clear picture of the difference of each method in computing time. By observing the left plot firstly, for most simulations, the computing time of ASL is decreased than that of FSL. When computing with SAASL, the computing time for some simulations increased again, but for most cases, the computing times stay still or slightly decreasing. As for SAASL05, all simulations are calculated with the fastest speed. And for gamboostLSS, the computing time increased again at the level of ASL and SAASL. Similar results can also be concluded from the boxplots. SAASL have a slightly lower mean value than ASL, and SAASL05 is the fastest algorithms.

A question about the computing time is, as both FSL and gamboostLSS are fixed step length algorithm, and the step length in both cases is set as 0.1, why the computing time behaves quite differently. The plausible reason is similar to the difference between the estimated coefficients of the two methods. As program code of FSL is not identical to that of gamboostLSS, the potential speedup methods in gamboostLSS exist not in the FSL, so it is not fair to compare the computing time of gamboostLSS and all the other algorithms directly. A fair comparison shall be among FSL, ASL, SAASL and SAASL05 because the structure of the codes among these methods are identical; the only difference lies in the setting on the step length. As a result, there is enough reason to believe, if the (analytical) adaptive step length settings are applied to the gamboostLSS package, similar results as in Figure 13 can be obtained.



(a) Computing time of each simulation (b) Boxplot of computing time

Figure 13: Comparison: Computing time

6.2.5 Overfitting

For different purpose, the accuracy alone is not enough to judge the performance of an algorithm. The overfitting should also be taken into consideration, because an overfitted model analyses too precisely to a particular set of data, that means the noise in the data are also fitted into the estimated model. As a result, it cannot be generalized to new data or predict further observations reliably.

A confusion matrix is a common tool for evaluating the overfitting in machine learning theory, especially for the statistical classification problem. In the confusion matrix, each row represents the predicted class, and each column represents the true one [Fawcett, 2006]. It makes it easy to see how many observations are right or wrongly classified. Table 1 demonstrated the confusion matrix. Some metrics often used in machine learning like *precison* $(= TP/(TP + FP))$ or *sensitivity* $(= TP/(TP + FN))$ will not be discussed in detail.

Two relatively important indices for our analysis is the false positives and false negatives. The former is often known as the type I error in statistics, which is checking a condition and wrongly gives a positive decision, while the latter is a type II error, which is a predicted result that a condition does not hold, however in fact it does.

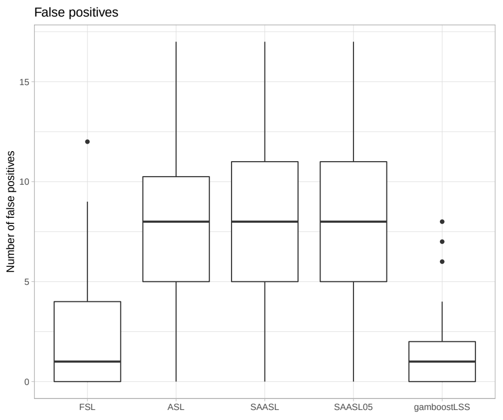
Back to our example, as the variable selection is performed parallelly with the model esti-

		True class	
		Condition positive	Condition negative
Predicted class	Predicted condition positive	True positive (TP)	False positive (FP)
	Predicted condition negative	False negative (FN)	True negative (TN)

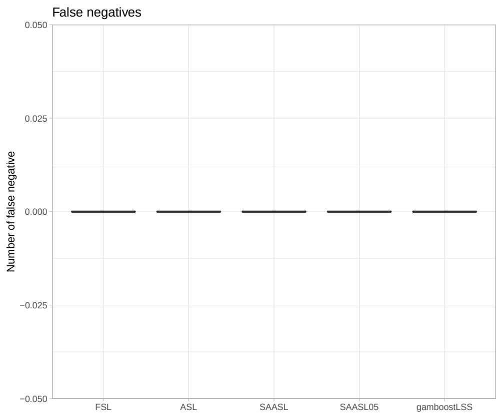
Table 1: Confusion matrix

mation in componentwise gradient boosting algorithm, a large number of iterations often result in more non-informative variables selected into the final predictors especially for high dimensional data. While the correct information contained in informative variables are extracted from each iteration, the effect of the remaining useful information is covered by the noise of the non-informative variables. As a result, more non-informative ones are selected into the final predictors. So if the number of informative and non-informative variables are treated as the true condition positive and negative class respectively, the confusion matrix can then be used to analyse the overfitting and underfitting behaviour of an algorithm.

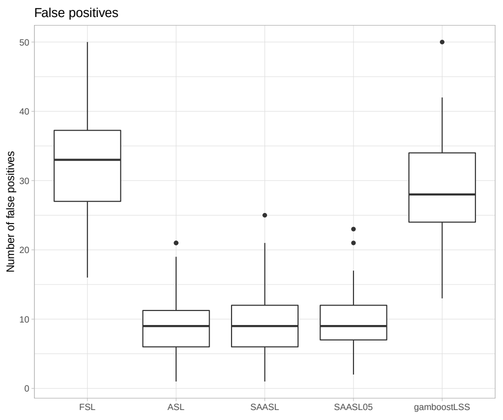
Figure 14 compares the false positives and false negatives among 100 simulations for both distribution parameters μ and σ among the five algorithms. Satisfying results are none of the algorithms ignored selecting the informative variables for both parameters. In other words, the informative variables are all selected by all methods. As for the false positives, for μ, the fixedwise methods selected less non-informative noise variables, while the adaptive methods selected rather more ones. This phenomenon is, however, reversed when seeing σ. The reason can be partially explained by the balance of decisions, see Figure 11. It can be observed from the balance comparison, that in fixedwise methods, more iterations are used to update the location parameter μ, whereas less used for updating σ. Even though the balance rate in SAASL05 is not as well as the other two approaches, it is significant that more iterations are applied for updating σ. Recall the step length settings in these algorithms. The real used step length of ν_σ in fixedwise methods is 0.1. The SAASL05 set the step length with a constant asymptotic value 0.5, but only 10% of it is used to update the predictors. That is to say, the real used step length in SAASL05 is 0.05, which is smaller than 0.1. Given all these pieces of information, we can believe that the fixed step length 0.1 for updating σ is a relatively aggressive value. Therefore, more non-informative variables in the predictors of σ are selected into the final model because of the relatively large step length. Similarly, as the adaptive methods update μ with the adaptive step lengths, though also only 10% of the optimal value is applied in each iteration, these values are in most cases larger than 0.1. So the false positive rates of the adaptive methods are larger than that of fixedwise ones.



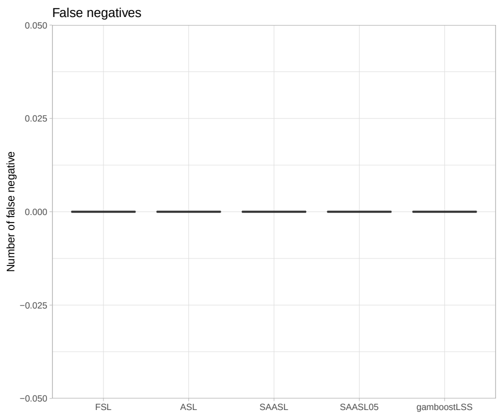
(a) μ: False positives



(b) μ: False negatives



(c) σ: False positives

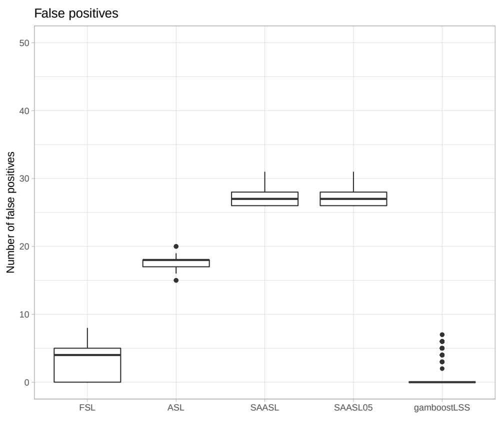


(d) σ: False negatives

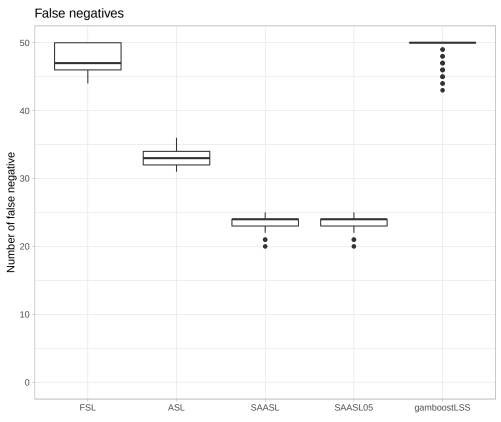
Figure 14: Comparison: False positives & False negatives I. As the data type is 2B, so 97 and 3 out of 100 variables are informative and non-informative variables for μ, 98 and 2 for σ. The fixedwise methods selected less non-informative variables for μ, while the adaptive methods selected less non-informative variables of σ.

Since there are only three and two informative variables in this model, and the coefficients for these variables are relatively large enough, it is easy for the algorithms to find their effectiveness. So it is, in fact, useless to compare the false negatives with the data type 2B in Figure 14, as all algorithms have the zero values and tell no difference. It is worthy of performing a more complex experiment and observing how different of the false negatives the algorithms behave.

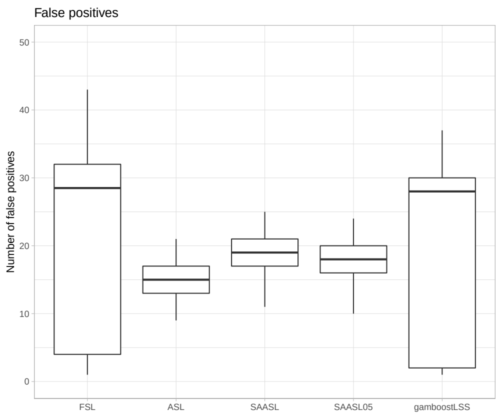
Given 100 variables $x_i, i \in \{1, \cdots, 100\}$, let the first 50 variables to be the informative variables, and the rest 50 be the non-informative noise variables. Given the coefficients set $\{-2, -1, 1, 2, 3\}$ for μ, and the coefficients set for σ are $\{-0.15, -0.1, -0.05, 0.15, 0.1\}$. The coefficients of the first ten informative variables are -2 and -0.15 for μ and σ respectively, and the coefficients of the next ten ones are -1 and -0.1, and so on. Then each algorithm fits the more complex data with maximal 1000 iterations. By this way, quite different results are presented in Figure 15.



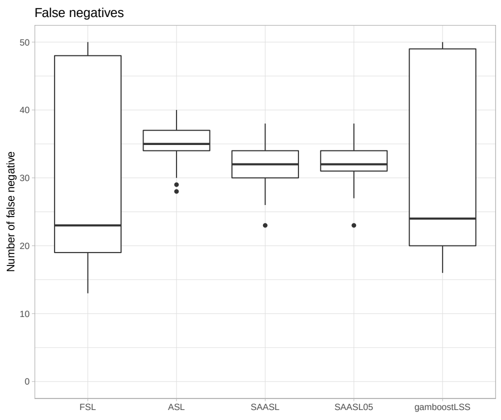
(a) μ: False positives



(b) μ: False negatives



(c) σ: False positives



(d) σ: False negatives

Figure 15: Comparison: False positives & False negatives II. For more details, see the main context.

There are two points that this data generating mechanism differs from the type B model. The first point is more informative variables are added into the model and actually 50 variables so that it will be more difficult for the algorithms to find all of them out. The other point is, as there are 50 variables in total affect not only the location parameter μ, but also the scale parameter σ, the typically used coefficient in simulation like 1 or 0.5 together with the 50 variables will make σ the astronomical figures of for example e^{50} or e^{25}. So the coefficients of the σ are set with the small values. But these small coefficients, on the other hand, make it more difficult for the algorithms to fit, because the impact of the noise variables becomes, in contrast, more considerable.

Firstly, observing the outputs of μ. Until the early stopping iterations specified by 10-folds CV, the FSL and gamboostLSS methods selected relatively less false positives. The adaptive methods selected in contrast more non-informative variables to the predictors. However, it does not mean, that the fixedwise methods have better performance because when we observe the picture of the false negatives, the fixedwise methods also selected no informative variables. In total, the fixedwise methods didn't update the predictors of μ at all. The reason can be explained by the limitation of $m_{\text{stop}} = 1000$. Recall the step length settings in fixedwise methods, the step length set for both distribution parameters is 0.1. As we've seen in the theory sections that after a sufficient number of iterations have been performed, the optimal step length for σ should be 0.5, and in most cases, the optimal values should also not too far away from 0.5. Take 10% of 0.5 as the real used updating step length, the fixed step length 0.1 is then, in contrast, a relatively large value ($0.1 > 0.05$). Consequently, updating σ will usually reduce more empirical risks than updating μ, because the optimal step length for μ in each iteration is usually larger than 1, as we have seen in Figure 2, whose coefficients are similar with this experiment. Even though only 10% of the optimal values are applied, they are still much bigger than the predefined fixed value 0.1. Additional information, that is not displayed in the graphics, should be mentioned here is the early stopping values for all algorithms in most cases is 1000, which is a sign of underfitting. In other words, both fixedwise approaches distributed nearly all the computing sources to updating σ; this corresponds to their property of making unbalanced decisions. In contrast, the coefficients of μ in adaptive methods with higher step length in each iteration have more chances to be updated, as its risk reduction is competitive as that of the σ. Theoretically, the false positive/negative behaviour of the adaptive methods should be quiet similar, but the ASL seems to have a lower value in false positives and a higher value in false negatives. It is also reasonable, as there is an upper boundary (actually 10) of the step length in the ASL method, and the optimal values, however, can greater than 10. And in these iterations, the risk reduction of μ should be less than σ. So the ASL method update more frequently the predictor of σ than μ within the limited 1000 iterations. As long as more iterations are performed, the ASL method will, of course, select more informative variables, but of course, also more non-informative variables. So we can still believe, that there are no significant differences among the three adaptive methods.

Then let's observe the σ. Accounting for the little unbalanced decisions made by ASL, there is, in fact, no difference among the adaptive methods. The comparison should be between the adaptive methods and the fixedwise methods. At first glance, the variation of fixedwise methods is significant not only in false positive but also in false negatives. After examining the early stopping values (see Appendix Figure B.3), some simulations with the fixedwise methods have very small early stopping values. It is probably caused by the small coefficients of σ and the relatively large updating step length 0.1. As the real coefficients are very small, it is easy to overfit the coefficient of a variable and failed in generalization. The CV then determines to stop early in these situations. Fortunately, the median lies in a reasonable region. So we focus on the median and compare them. The medians of false positives of fixedwise methods are at a high level, whereas a low level of false negatives. It is also caused by the step length settings.

As the fixed step length 0.1 is larger than the approximate adaptive step length 0.05, besides, 1000 iterations are not enough for learning a good model, the fixedwise methods can select more informative variables than the adaptive ones within 1000 iterations. But the cost of this risky step length is, of course, more non-informative variables are included in the final model.

6.2.6 Special cases

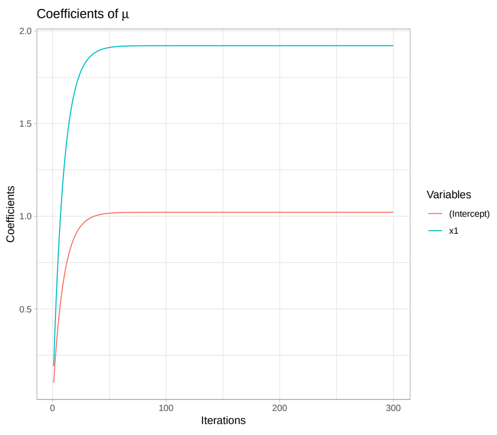
This part will present two simulation outputs related to the properties of the adaptive step length of ν_μ proofed in the previous sections.

- Generalized additive models (GAM)
  The first one is the application of adaptive step length in GAM. As proved in Eq.(5.20), the adaptive step length for μ in GLM is the variance of the response. Assume a model with following structures,

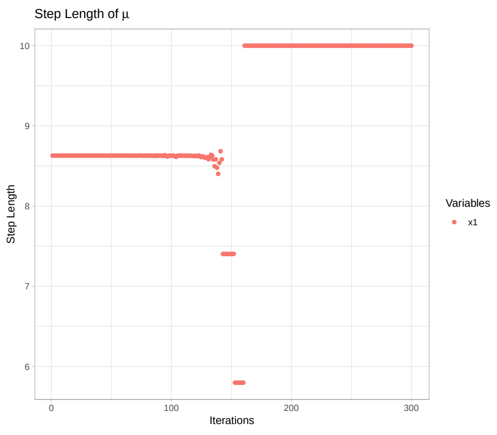
$$\begin{aligned}\mu_i &= 1 + 2x_i \\ \sigma_i &= \exp(1),\end{aligned}$$

  as the σ is constant for all i, the GAMLSS is reduced to GAM or the simple linear model.

  Figure 16 illustrated the coefficients and step length ν_μ when applying the ASL algorithms to fit the data, i.e. the step length is found by doing a line search instead of an analytical calculation.



(a) Coefficients



(b) Step Length

Figure 16: Coefficients and step length of GAM

  As expected, though the adaptive step length is not calculated by the analytical solution, it is still a constant value 8.63, the half of its logarithm is 1.08, which is approximate to the actual value 1. The variation after about 130 iterations should be caused by the computational accuracy. As the model has been already well fitted at about 50 iterations, further updates help little and require high computing ability.

- Harmonic mean

  Another special case is that after a sufficient number of iterations have been performed, the adaptive step length in GAMLSS is approximately equal to the harmonic mean of the variance in the previous iteration, see Eq.(5.21). As discussed before, this relationship will be more clear if relatively large standard deviations are given, as the side effect of the base-learn will be covered. The simulation with the model structure is operated,

$$\begin{aligned}\mu_i &= x_i\\ \sigma_i &= \exp(4x_i).\end{aligned}$$

  The corresponding output is demonstrated in Figure 17. The model is fitted with the SAASL algorithm and compared the analytical adaptive step length of μ and the harmonic mean of σ_i. From the picture, it can be observed clearly, that both values are roughly identical after about 100 iterations.

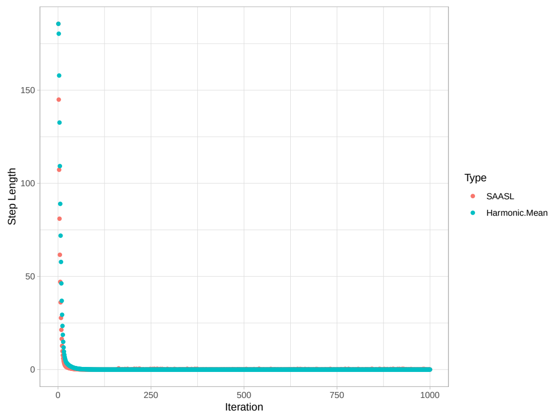


Figure 17: Harmonic mean

6.2.7 Summaries of the simulations in the appendix

As there are many outputs exported from the ten models, it will be orderless if they are all listed in the main context. So we briefly summarize the findings from their results attached in the appendix.

For model 1a and 1b, i.e. the balanced and unbalanced cases without non-informative variables, their performances are similar to each other. The coefficients in both models are very well estimated in all algorithms, and their corresponding mean squared error, as well as the empirical risk, are almost on the same level. The decisions made by fixedwise methods are unbalanced and by adaptive methods rather balanced. Unlike the model 2b, the runtime of gamboostLSS in not only these two cases but also all the without non-informative variables scenario data cases is longer than the other four methods. The probable reason lies in the complicated internal function call and the completed test on the input structures, which on the other hand, slows down the computing speed when fitting a simple model. For model 1a, the runtime of FSL is a little shorter than ASL, which might be evidence of the argument of [Bühlmann and Hothorn, 2007]. The speed ranking of the three adaptive methods fulfils the theoretical judgement.

As for the correlated cases 1c and 2c, the adaptive methods are proofed to have worse performance than the fixedwise ones (or more precisely gamboostLSS, as the performances of FSL are also not satisfied). As the predictors are updated with the very bold steps, the early stopping values specified by CV are very small, and in fact, for most cases less than 10, which means the algorithms probable stop before the informative variables chosen into the model and before the coefficients relatively well estimated. Consequently, the performances of the adaptive methods have a considerable variation.

Just as discussed in 6.2.3, only the adaptive methods can produce a reasonable estimation of the models with extremely coefficients. The SAASL and SAASL05 have similar performances in all aspects. Theoretically, the ASL should also deliver a good result. However, within the predefined 1000 iterations, the upper boundary is still small for solving the given models. The analysis of the runtime, which is faster than ASL or even quicker than SAASL05 in these cases, is meaningless, as its coefficients are still far away from the actual values.

6.2.8 Summary

Overall speaking, even though the adaptive method is not always the best choice in all situations, it provides a fast, balanced and stable solution to the GAMLSS. Table 2 summarized the comparison of each algorithm according to the simulations.

The main advantages of the adaptive methods lie in computational efficiency and making balanced decisions. Its drawback is the tendency of overfitting the location parameter, as the optimal step length in each iteration is usually large. Assume that the observations are drawn from a multi-dimensional Gaussian distribution, the adaptive step length found by doing a line search should be replaced with the semi-analytical ones, as firstly they are in essence the same thing and the results have no significant difference, and secondly, the semi-analytical solution provides more balanced solutions. If the balance of decisions is not a very important point during analysis, a much faster alternative SAASL05 is the best choice without worrying about the compromise in accuracy.

The strength of the fixed step length algorithm is that it tends to select fewer false positives of the location parameter into the final model; this is because of the carefully small step length 0.1. On the other hand, the same step length for the scale parameter is a relatively careless one; at least it is large in Gaussian distribution. This problem can be solved by changing the step length for σ to 0.05 manually. It will also make the decisions more balanced, but still hard to be as well as the balance rate in adaptive methods. Though [Bühlmann and Hothorn, 2007] argued that the fixed step length have the similar efficiency as the adaptive ones, which, however, can not be proofed in this thesis properly, but it should be slower than the SAASL and much slower than the SAASL05.

	FSL	ASL	SAASL	SAASL05	gamboostLSS
Accuracy	(+)	+	+	+	(+)
Balance of decisions	-	(+)	+	(+)	-
Computational efficient	-	(+)	+	+	(-)
False positives (μ)	+	-	-	-	+
False positives (σ)	-	+	+	+	-
False negatives (μ)	(-)	(+)	(+)	(+)	(-)
False negatives (σ)	+	-	-	-	+

Table 2: Summary of the comparisons. The "+" sign stands for the positive result, and the "-" stands for the negative. The brace around each sign represent the opinion of the author, i.e. there are not enough evidences from the simulations support the conclusion.

7 Conclusion

This thesis briefly introduced the theory of GAMLSS and gradient boosting. Account for the shortcomings of the conventional inference methods, we introduced the “non-cyclical” approach which embedded the variable selection procedure into the learning process. As this approach might lead to unbalanced decisions with fixed step length, we analyzed the influence of the step length on the balance and found that the adaptive step length can solve the problem properly.

For Gaussian distribution, according to the analytical solution to the adaptive step length for the location parameter, we introduced a new semi-analytical approach (SAASL), and its variation (SAASL05) by replacing the adaptive step length for scale parameter with the asymptotic value of 0.5. Although the adaptive step length approaches cannot estimate the correlated models correctly, their effectiveness and balance are impressive. It also suggests a more reasonable fixed step length (0.05) for the scale parameter in Gaussian distribution when using the fixedwise approaches.

The creation of the SAASL05 also gives a new idea, that even though the analytical solution not always exists for every parameter in each distribution, it is worthy of inducing them as far as possible and trying to find a more reasonable fixed step length according to their properties. This thesis induced only the optimal step length in Gaussian distribution; more works on other distributions still need to be done.

References

[Akaike, 1974] Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723.

[Akaike, 1983] Akaike, H. (1983). Information measures and model selection. *Int Stat Inst*, 44:277–291.

[Bozdogan, 2000] Bozdogan, H. (2000). Akaike's information criterion and recent developments in information complexity. *Journal of mathematical psychology*, 44(1):62–91.

[Breiman and Spector, 1992] Breiman, L. and Spector, P. (1992). Submodel selection and evaluation in regression. the x-random case. *International Statistical Review / Revue Internationale de Statistique*, 60(3):291–319.

[Brent, 2013] Brent, R. P. (2013). *Algorithms for minimization without derivatives.* Courier Corporation.

[Bühlmann, 2006] Bühlmann, P. (2006). Boosting for high-dimensional linear models. *Ann. Statist.*, 34(2):559–583.

[Bühlmann et al., 2014] Bühlmann, P., Gertheiss, J., Hieke, S., Kneib, T., Ma, S., Schumacher, M., Tutz, G., Wang, C.-Y., Wang, Z., and Ziegler, A. (2014). Discussion of "the evolution of boosting algorithms" and "extending statistical boosting". *Methods of information in medicine*, 53:436–445.

[Bühlmann and Hothorn, 2007] Bühlmann, P. and Hothorn, T. (2007). Boosting algorithms: Regularization, prediction and model fitting. *Statist. Sci.*, 22(4):477–505.

[Bühlmann and Yu, 2003] Bühlmann, P. and Yu, B. (2003). Boosting with the l2 loss. *Journal of the American Statistical Association*, 98(462):324–339.

[Chen and Guestrin, 2016] Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754.

[Fawcett, 2006] Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861 – 874. ROC Analysis in Pattern Recognition.

[Freund and Schapire, 1996] Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, ICML'96, pages 148–156, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Freund and Schapire, 1997] 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, 2001] Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Ann. Statist.*, 29(5):1189–1232.

[Hastie, 2007] Hastie, T. (2007). Comment: Boosting algorithms: Regularization, prediction and model fitting. *Statist. Sci.*, 22(4):513–515.

[Hastie et al., 2009] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference and prediction.* Springer, 2 edition.

[Hasties and Tibshirani, 1990] Hasties, T. and Tibshirani, R. (1990). Generalized additive models. *London: Chapm ana n dHal1*, pages 137–173.

[Hofner et al., 2015] Hofner, B., Boccuto, L., and Göker, M. (2015). Controlling false discoveries in high-dimensional situations: boosting with stability selection. *BMC Bioinformatics*, 16(1):144.

[Hofner et al., 2018] Hofner, B., Mayr, A., Fenske, N., and Schmid, M. (2018). *gamboostLSS: Boosting Methods for GAMLSS Models.* R package version 2.0-1.

[Hofner et al., 2014] Hofner, B., Mayr, A., Robinzonov, N., and Schmid, M. (2014). Model-based boosting in r: A hands-on tutorial using the r package mboost. *Comput. Stat.*, 29(1-2):3–35.

[Hofner et al., 2016] 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, Articles*, 74(1):1–31.

[Ke et al., 2017] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 3146–3154. Curran Associates, Inc.

[Kohavi et al., 1995] Kohavi, R. et al. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *Ijcai*, 14(2):1137–1145.

[Mayr et al., 2012] Mayr, A., Fenske, N., Hofner, B., Kneib, T., and Schmid, M. (2012). Generalized additive models for location, scale and shape for high dimensional data—a flexible approach based on boosting. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 61(3):403–427.

[Meinshausen and Bühlmann, 2010] Meinshausen, N. and Bühlmann, P. (2010). Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473.

[Nocedal and Wright, 1999] Nocedal, J. and Wright, S. J. (1999). *Numerical Optimization.* Springer, New York, NY, USA.

[R Core Team, 2018] R Core Team (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

[Rigby and Stasinopoulos, 2005] Rigby, R. A. and Stasinopoulos, D. M. (2005). Generalized additive models for location, scale and shape. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 54(3):507–554.

[Ripley, 2004] Ripley, B. D. (2004). *SELECTING AMONGST LARGE CLASSES OF MODELS*, pages 155–170. Methods and Models in Statistics.

[Ruder, 2016] Ruder, S. (2016). An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747.

[Schapire, 1990] Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, 5(2):197–227.

[Schwarz, 1978] Schwarz, G. (1978). Estimating the dimension of a model. *Ann. Statist.*, 6(2):461–464.

[Sclove, 1987] Sclove, S. L. (1987). Application of model-selection criteria to some problems in multivariate analysis. *Psychometrika*, 52(3):333–343.

[Stasinopoulos and Rigby, 2007] Stasinopoulos, D. and Rigby, R. (2007). Generalized additive models for location scale and shape (gamlss) in r. *Journal of Statistical Software, Articles*, 23(7):1–46.

[Stasinopoulos et al., 2017] Stasinopoulos, D., Rigby, R., Heller, G., Voudouris, V., and De Bastiani, F. (2017). *Flexible regression and smoothing: Using GAMLSS in R*. Chapman and Hall/CRC.

[Thomas et al., 2018] Thomas, J., Mayr, A., Bischl, B., Schmid, M., Smith, A., and Hofner, B. (2018). Gradient boosting for distributional regression: faster tuning and improved variable selection via noncyclical updates. *Statistics and Computing*, 28(3):673–687.

A Mathematical induction

A.1 Efficiency of the fixed step length

Consider the empirical risk at iteration m,

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n}\rho\left(y^{(i)},\eta^{[m]}(x^{(i)})\right) && \text{(A.1)}\\
\approx&\frac{1}{n}\sum_{i=1}^{n}\rho\left(y^{(i)},\eta^{[m-1]}(x^{(i)})\right)+\frac{1}{n}\sum_{i=1}^{n}\underbrace{\frac{\partial}{\partial\eta^{[m-1]}}\rho\left(y^{(i)},\eta^{[m-1]}(x^{(i)})\right)}_{=-u^{(i)}}\underbrace{\left[\eta^{[m]}(x^{(i)})-\eta^{[m-1]}(x^{(i)})\right]}_{=\nu\hat{h}^{[m]}(x^{(i)})} && \text{(A.2)}\\
=&\frac{1}{n}\sum_{i=1}^{n}\rho\left(y^{(i)},\eta^{[m-1]}(x^{(i)})\right)-\nu\frac{1}{n}\sum_{i=1}^{n}u^{(i)}\hat{h}^{[m]}(x^{(i)}) && \text{(A.3)}
\end{aligned}$$

using the first-order Taylor expansion at $\eta^{[m-1]}(x^{(i)})$ and the definition of pseudo-residuals $u^{(i)}$. With the componentwise linear squares base procedure and with the standardized predictor variables,

$$\hat{h}^{[m]}(x)=\frac{\sum_{i=1}^{n}x^{(i)}u^{(i)}}{\sum_{i=1}^{n}(x^{(i)})^2}x=x\frac{1}{n}\sum_{i=1}^{n}u^{(i)}x^{(i)} \tag{A.4}$$

as for standardized predictor variables, $\frac{1}{n}\sum_{i=1}^{n}(x^{(i)})^2=1$. The expression in (A.3) becomes

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n}\rho\left(y^{(i)},\eta^{[m]}(x^{(i)})\right) && \text{(A.5)}\\
\approx&\frac{1}{n}\sum_{i=1}^{n}\rho\left(y^{(i)},\eta^{[m-1]}(x^{(i)})\right)-\nu\frac{1}{n}\sum_{i=1}^{n}u^{(i)}\left[x^{(i)}\frac{1}{n}\sum_{i=1}^{n}u^{(i)}x^{(i)}\right] && \text{(A.6)}\\
=&\frac{1}{n}\sum_{i=1}^{n}\rho\left(y^{(i)},\eta^{[m-1]}(x^{(i)})\right)-\nu\left(\frac{1}{n}\sum_{i=1}^{n}u^{(i)}x^{(i)}\right)^2 && \text{(A.7)}
\end{aligned}$$

For adjusted L2 loss, i.e. $\rho(y,\eta)=\frac{1}{2}(y-\eta)^2$, the empirical risk can be directly derived as

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n}\rho\left(y^{(i)},\eta^{[m]}(x^{(i)})\right) && \text{(A.8)}\\
=&\frac{1}{n}\sum_{i=1}^{n}\rho\left(y^{(i)},\eta^{[m-1]}(x^{(i)})+\nu\hat{h}^{[m]}(x^{(i)})\right) && \text{(A.9)}\\
=&\frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\left(y^{(i)}-\eta^{[m-1]}(x^{(i)})-\nu\hat{h}^{[m]}(x^{(i)})\right)^2 && \text{(A.10)}\\
=&\frac{1}{n}\sum_{i=1}^{n}\left[\frac{1}{2}\left(y^{(i)}-\eta^{[m-1]}(x^{(i)})\right)^2-\underbrace{\left(y^{(i)}-\eta^{[m-1]}(x^{(i)})\right)}_{=u^{(i)},\text{ see Eq.(3.12)}}\nu\hat{h}^{[m]}(x^{(i)})+\frac{1}{2}\nu^2\left(\hat{h}^{[m]}(x^{(i)})\right)^2\right] && \text{(A.11)}
\end{aligned}$$

$$=\frac{1}{n}\sum_{i=1}^{n}\rho\left(y^{(i)},\eta^{[m-1]}(x^{(i)})\right)-\nu\frac{1}{n}\sum_{i=1}^{n}u^{(i)}\hat{h}^{[m]}(x^{(i)})+\frac{\nu^2}{2}\frac{1}{n}\sum_{i=1}^{n}\left(\hat{h}^{[m]}(x^{(i)})\right)^2 \tag{A.12}$$

$$=\frac{1}{n}\sum_{i=1}^{n}\rho\left(y^{(i)},\eta^{[m-1]}(x^{(i)})\right)-\nu\frac{1}{n}\sum_{i=1}^{n}\left[u^{(i)}\left(x^{(i)}\frac{1}{n}\sum_{i=1}^{n}u^{(i)}x^{(i)}\right)\right]+\frac{\nu^2}{2}\frac{1}{n}\sum_{i=1}^{n}\left[x^{(i)}\frac{1}{n}\sum_{i=1}^{n}u^{(i)}x^{(i)}\right]^2 \tag{A.13}$$

$$=\frac{1}{n}\sum_{i=1}^{n}\rho\left(y^{(i)},\eta^{[m-1]}(x^{(i)})\right)-\nu\left(\frac{1}{n}\sum_{i=1}^{n}u^{(i)}x^{(i)}\right)^2+\frac{\nu^2}{2}\underbrace{\frac{1}{n}\sum_{i=1}^{n}\left(x^{(i)}\right)^2}_{=1}\left(\frac{1}{n}\sum_{i=1}^{n}u^{(i)}x^{(i)}\right)^2 \tag{A.14}$$

$$=\frac{1}{n}\sum_{i=1}^{n}\rho\left(y^{(i)},\eta^{[m-1]}(x^{(i)})\right)-\nu\left(1-\frac{\nu}{2}\right)\left(\frac{1}{n}\sum_{i=1}^{n}u^{(i)}x^{(i)}\right)^2 \tag{A.15}$$

Given a small value of step length ν, the gradient descent with a general loss function (Eq.(A.7)) behaves very similarly to L2-Boosting (Eq.(A.15)) with respect to minimizing the empirical risk. Consider the additinoal computing cost of line search, [Bühlmann and Hothorn, 2007] believe that a small fixed step length is sufficient for boosting algorithms.

A.2 ASL for μ

The analytical ASL for μ in iteration m can be obtained through minimizing the empirical risk,

$$\begin{aligned}
\nu_\mu^{[m]} &= \underset{\nu_\mu^{[m]}}{\arg\min} \sum_{i=1}^{n} \rho(y_i, \{\mu_i^{[m]}, \sigma_i^{[m-1]}\}) && \text{(A.16)}\\
&= \underset{\nu_\mu^{[m]}}{\arg\min} \sum_{i=1}^{n} \rho(y_i, \{\hat{\eta}_\mu^{[m-1]} + \nu_\mu^{[m]}\hat{h}_\mu^{[m]}(x_{ij^*}), \sigma_i^{[m-1]}\}) && \text{(A.17)}\\
&= \underset{\nu_\mu^{[m]}}{\arg\min} \sum_{i=1}^{n} -\log\left[\frac{1}{\sqrt{2\pi}\sigma_i^{[m-1]}}\exp\left(-\frac{\left(y_i - \hat{\eta}_\mu^{[m-1]} - \nu_\mu^{[m]}\hat{h}_\mu^{[m]}(x_{ij^*})\right)^2}{2\sigma_i^{2[m-1]}}\right)\right] && \text{(A.18)}\\
&= \underset{\nu_\mu^{[m]}}{\arg\min} \sum_{i=1}^{n} \frac{1}{2}\log(2\pi) + \sum_{i=1}^{n}\log\sigma_i^{[m-1]} + \sum_{i=1}^{n}\frac{\left(y_i - \hat{\eta}_\mu^{[m-1]} - \nu_\mu^{[m]}\hat{h}_\mu^{[m]}(x_{ij^*})\right)^2}{2\sigma_i^{2[m-1]}} && \text{(A.19)}\\
&= \underset{\nu_\mu^{[m]}}{\arg\min} \sum_{i=1}^{n} \frac{\left(y_i - \hat{\eta}_\mu^{[m-1]} - \nu_\mu^{[m]}\hat{h}_\mu^{[m]}(x_{ij^*})\right)^2}{2\sigma_i^{2[m-1]}} && \text{(A.20)}
\end{aligned}$$

Note that the expression $\sigma_i^{2[m-1]}$ represent for the square of the previous standard deviation, i.e. $\sigma_i^{2[m-1]} = (\sigma_i^{[m-1]})^2$.

The optimal value of $\nu_\mu^{[m]}$ can be accessed by letting the derivative of the equation equal zero, so we get

$$\begin{aligned}
&\frac{\partial}{\partial\nu_\mu^{[m]}} \sum_{i=1}^{n} \frac{\left(y_i - \hat{\eta}_\mu^{[m-1]} - \nu_\mu^{[m]}\hat{h}_\mu^{[m]}(x_{ij^*})\right)^2}{2\sigma_i^{2[m-1]}} && \text{(A.21)}\\
=&\frac{\partial}{\partial\nu_\mu^{[m]}} \sum_{i=1}^{n} \frac{\left(u_{\mu i}^{[m]}\sigma_i^{2[m-1]} - \nu_\mu^{[m]}\hat{h}_\mu^{[m]}(x_{ij^*})\right)^2}{2\sigma_i^{2[m-1]}} && \text{(A.22)}\\
=&\frac{\partial}{\partial\nu_\mu^{[m]}} \sum_{i=1}^{n} \frac{(u_{\mu i}^{[m]}\sigma_i^{2[m-1]})^2 - 2\nu_\mu^{[m]}\hat{h}_\mu^{[m]}(x_{ij^*})u_{\mu i}^{[m]}\sigma_i^{2[m-1]} + (\nu_\mu^{[m]})^2(\hat{h}_\mu^{[m]}(x_{ij^*}))^2}{2\sigma_i^{2[m-1]}} && \text{(A.23)}\\
=&\frac{\partial}{\partial\nu_\mu^{[m]}} \sum_{i=1}^{n} \left(\frac{1}{2}u_{\mu i}^{2[m]}\sigma_i^{2[m-1]} - \nu_\mu^{[m]}\hat{h}_\mu^{[m]}(x_{ij^*})u_{\mu i}^{[m]} + \frac{(\nu_\mu^{[m]})^2(\hat{h}_\mu^{[m]}(x_{ij^*}))^2}{2\sigma_i^{2[m-1]}}\right) && \text{(A.24)}\\
=&\sum_{i=1}^{n}\left[-\hat{h}_\mu^{[m]}(x_{ij^*})u_{\mu i}^{[m]} + \nu_\mu^{[m]}\frac{(\hat{h}_\mu^{[m]}(x_{ij^*}))^2}{\sigma_i^{2[m-1]}}\right] \overset{!}{=} 0 && \text{(A.25)}\\
\Leftrightarrow \nu_\mu^{[m]} =& \frac{\sum_{i=1}^{n}\hat{h}_\mu^{[m]}(x_{ij^*})u_{\mu i}^{[m]}}{\sum_{i=1}^{n}\frac{(\hat{h}_\mu^{[m]}(x_{ij^*}))^2}{\sigma_i^{2[m-1]}}} && \text{(A.26)}\\
=& \frac{\sum_{i=1}^{n}\hat{h}_\mu^{[m]}(x_{ij^*})\left(\hat{h}_\mu^{[m]}(x_{ij^*}) + \epsilon_{\mu i}\right)}{\sum_{i=1}^{n}\frac{(\hat{h}_\mu^{[m]}(x_{ij^*}))^2}{\sigma_i^{2[m-1]}}} && \text{(A.27)}
\end{aligned}$$

$$= \frac{\sum_{i=1}^{n} \left(\hat{h}_{\mu}^{[m]}(x_{ij^*})\right)^2 + \sum_{i=1}^{n} \hat{h}_{\mu}^{[m]}(x_{ij^*})\epsilon_{\mu i}}{\sum_{i=1}^{n} \frac{(\hat{h}_{\mu}^{[m]}(x_{ij^*}))^2}{\sigma_i^{2[m-1]}}} \tag{A.28}$$

$$= \frac{\sum_{i=1}^{n} \left(\hat{h}_{\mu}^{[m]}(x_{ij^*})\right)^2}{\sum_{i=1}^{n} \frac{(\hat{h}_{\mu}^{[m]}(x_{ij^*}))^2}{\sigma_i^{2[m-1]}}} \tag{A.29}$$

where $\sum_{i=1}^{n} \hat{h}_{\mu}^{[m]}(x_{ij^*})\epsilon_{\mu i} = 0$ in Eq.(A.28), because the residuals are uncorrelated with the fitted values.

A.3 ASL for σ

The analytical ASL for σ in iteration m can be obtained through minimizing the empirical risk,

$$\nu_\sigma^{[m]} = \arg\min_{\nu_\sigma^{[m]}} \sum_{i=1}^n \rho(y_i, \{\mu_i^{[m-1]}, \sigma_i^{[m]}\}) \tag{A.30}$$

$$= \arg\min_{\nu_\sigma^{[m]}} \sum_{i=1}^n \rho\left(y_i, \left\{\mu_i^{[m-1]}, \exp\left(\hat{\eta}_\sigma^{[m-1]} + \nu_\sigma^{[m]}\hat{h}_\sigma^{[m]}(x_{ij^*})\right)\right\}\right) \tag{A.31}$$

$$= \arg\min_{\nu_\sigma^{[m]}} \sum_{i=1}^n -\log\left[\frac{1}{\sqrt{2\pi}\exp\left(\hat{\eta}_\sigma^{[m-1]} + \nu_\sigma^{[m]}\hat{h}_\sigma^{[m]}(x_{ij^*})\right)} \exp\left(-\frac{\left(y_i - \mu_i^{[m-1]}\right)^2}{2\exp\left(2\hat{\eta}_\sigma^{[m-1]} + 2\nu_\sigma^{[m]}\hat{h}_\sigma^{[m]}(x_{ij^*})\right)}\right)\right] \tag{A.32}$$

$$= \arg\min_{\nu_\sigma^{[m]}} \sum_{i=1}^n \frac{1}{2}\log(2\pi) + \sum_{i=1}^n \left(\hat{\eta}_\sigma^{[m-1]} + \nu_\sigma^{[m]}\hat{h}_\sigma^{[m]}(x_{ij^*})\right) + \sum_{i=1}^n \frac{\left(y_i - \mu_i^{[m-1]}\right)^2}{2\exp\left(2\hat{\eta}_\sigma^{[m-1]} + 2\nu_\sigma^{[m]}\hat{h}_\sigma^{[m]}(x_{ij^*})\right)} \tag{A.33}$$

$$= \arg\min_{\nu_\sigma^{[m]}} \sum_{i=1}^n \left(\hat{\eta}_\sigma^{[m-1]} + \nu_\sigma^{[m]}\hat{h}_\sigma^{[m]}(x_{ij^*})\right) + \sum_{i=1}^n \frac{\left(y_i - \mu_i^{[m-1]}\right)^2}{2\exp\left(2\hat{\eta}_\sigma^{[m-1]} + 2\nu_\sigma^{[m]}\hat{h}_\sigma^{[m]}(x_{ij^*})\right)} \tag{A.34}$$

The optimal value can be accessed by calculating the derivative of the Eq.(A.34) and letting it equal zero. And of course, these expression must be proofed to be a convex function. Firstly, the derivative of Eq.(A.34) is

$$\frac{\partial}{\partial \nu_\sigma^{[m]}}\left[\sum_{i=1}^n \left(\hat{\eta}_\sigma^{[m-1]} + \nu_\sigma^{[m]}\hat{h}_\sigma^{[m]}(x_{ij^*})\right) + \sum_{i=1}^n \frac{\left(y_i - \mu_i^{[m-1]}\right)^2}{2\exp\left(2\hat{\eta}_\sigma^{[m-1]} + 2\nu_\sigma^{[m]}\hat{h}_\sigma^{[m]}(x_{ij^*})\right)}\right] \tag{A.35}$$

$$= \sum_{i=1}^n \hat{h}_\sigma^{[m]}(x_{ij^*}) - \sum_{i=1}^n \left(y_i - \mu_i^{[m-1]}\right)^2 \hat{h}_\sigma^{[m]}(x_{ij^*}) \exp\left(-2\hat{\eta}_\sigma^{[m-1]} - 2\nu_\sigma^{[m]}\hat{h}_\sigma^{[m]}(x_{ij^*})\right). \tag{A.36}$$

The corresponding second derivative is positive, because

$$\frac{\partial}{\partial \nu}\left[\sum_{i=1}^n \hat{h}_\sigma^{[m]}(x_{ij^*}) - \sum_{i=1}^n \left(y_i - \mu_i^{[m-1]}\right)^2 \hat{h}_\sigma^{[m]}(x_{ij^*}) \exp\left(-2\hat{\eta}_\sigma^{[m-1]} - 2\nu_\sigma^{[m]}\hat{h}_\sigma^{[m]}(x_{ij^*})\right)\right] \tag{A.37}$$

$$= \sum_{i=1}^n 2 \underbrace{\left(y_i - \mu_i^{[m-1]}\right)^2}_{>0} \underbrace{\left(\hat{h}_\sigma^{[m]}(x_{ij^*})\right)^2}_{>0} \underbrace{\exp\left(-2\hat{\eta}_\sigma^{[m-1]} - 2\nu_\sigma^{[m]}\hat{h}_\sigma^{[m]}(x_{ij^*})\right)}_{>0} > 0 \tag{A.38}$$

So the Eq.(A.34) is a convex function, and its minimum is gained by letting the derivative equal to zero, i.e.

$$\sum_{i=1}^n \hat{h}_\sigma^{[m]}(x_{ij^*}) - \sum_{i=1}^n \left(y_i - \mu_i^{[m-1]}\right)^2 \hat{h}_\sigma^{[m]}(x_{ij^*}) \exp\left(-2\hat{\eta}_\sigma^{[m-1]} - 2\nu_\sigma^{[m]}\hat{h}_\sigma^{[m]}(x_{ij^*})\right) \tag{A.39}$$

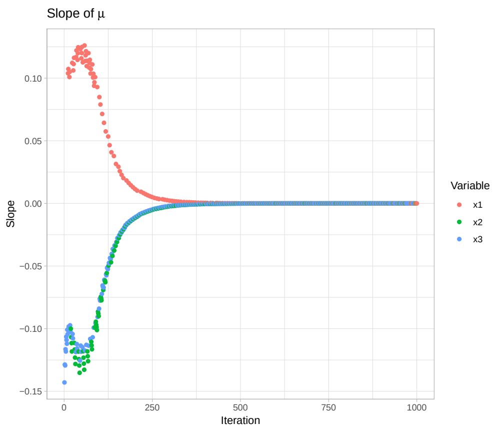
$$= \sum_{i=1}^{n} \hat{h}_{\sigma}^{[m]}(x_{ij^*}) - \sum_{i=1}^{n} \frac{u_{\sigma i}^{[m]} + 1}{\exp\left(-2\hat{\eta}_{\sigma}^{[m-1]}\right)} \hat{h}_{\sigma}^{[m]}(x_{ij^*}) \exp\left(-2\hat{\eta}_{\sigma}^{[m-1]} - 2\nu_{\sigma}^{[m]}\hat{h}_{\sigma}^{[m]}(x_{ij^*})\right) \tag{A.40}$$

$$= \sum_{i=1}^{n} \hat{h}_{\sigma}^{[m]}(x_{ij^*}) - \sum_{i=1}^{n} \left(u_{\sigma i}^{[m]} + 1\right) \hat{h}_{\sigma}^{[m]}(x_{ij^*}) \exp\left(-2\nu_{\sigma}^{[m]}\hat{h}_{\sigma}^{[m]}(x_{ij^*})\right) \tag{A.41}$$

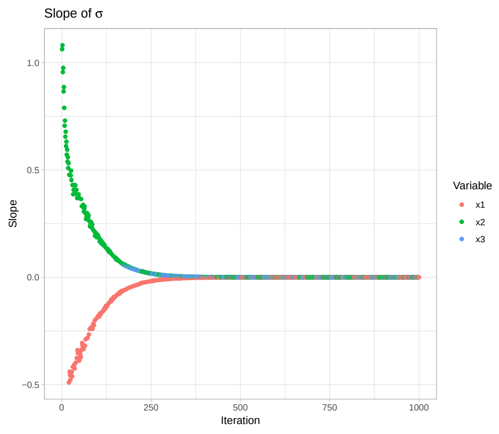
$$= \sum_{i=1}^{n} \hat{h}_{\sigma}^{[m]}(x_{ij^*}) - \sum_{i=1}^{n} \frac{\left(\hat{h}_{\sigma}^{[m]}(x_{ij^*}) + \epsilon_{\sigma i} + 1\right) \hat{h}_{\sigma}^{[m]}(x_{ij^*})}{\exp\left(2\nu_{\sigma}^{[m]}\hat{h}_{\sigma}^{[m]}(x_{ij^*})\right)} \overset{!}{=} 0 \tag{A.42}$$

B Figures

B.1 Extras



(a) μ



(b) σ

Figure B.1: Slope of μ and σ over iterations

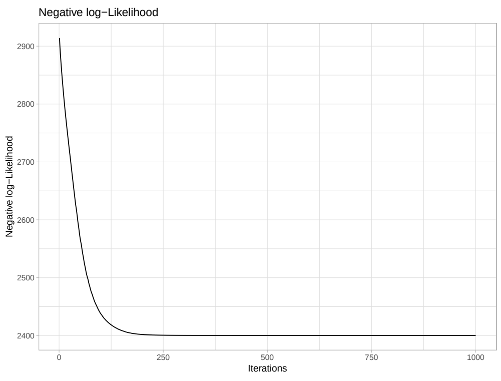


Figure B.2: Negative log-likelihood over iterations of type 1B model

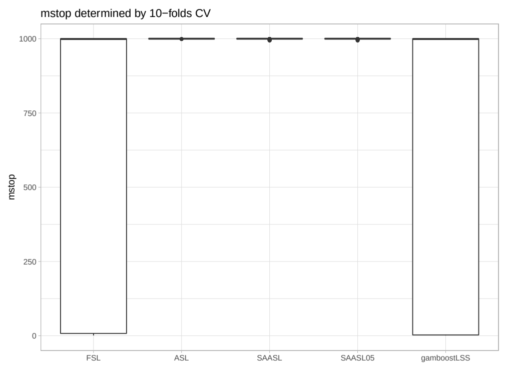
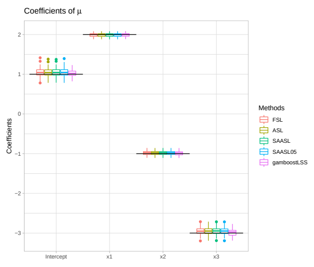
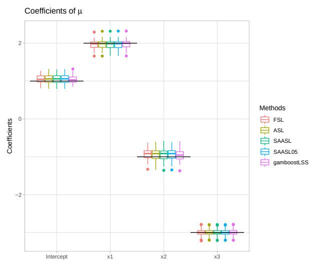


Figure B.3: m_{stop} of Comparison False positives/negatives II

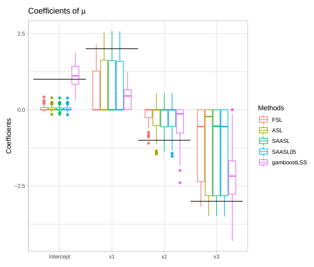
B.2 Accuracy



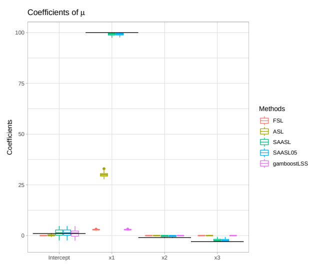
(a) Model 1a



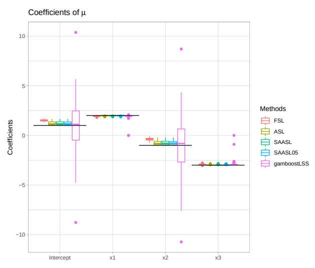
(b) Model 1b



(c) Model 1c

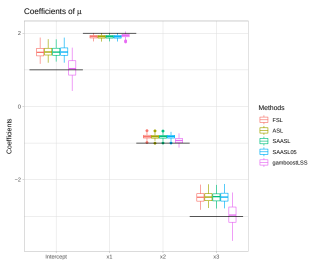


(d) Model 1d

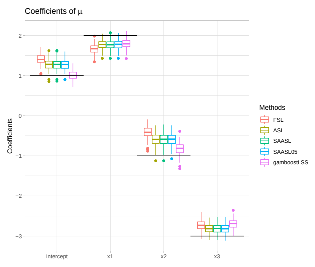


(e) Model 1e

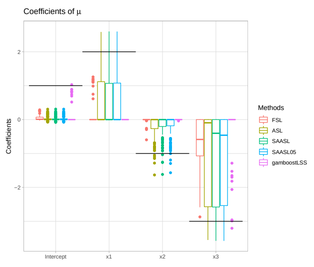
Figure B.4: Comparison: Estimated coefficients of μ



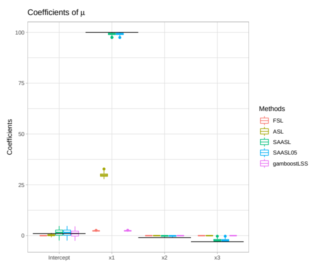
(a) Model 2a



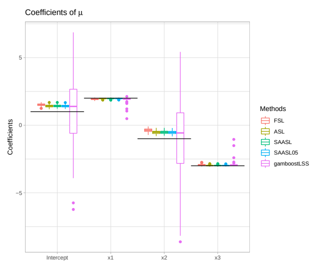
(b) Model 2b



(c) Model 2c

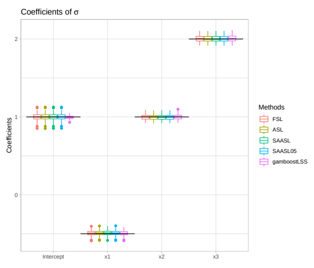


(d) Model 2d

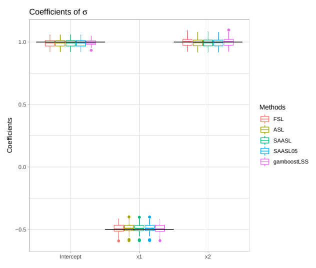


(e) Model 2e

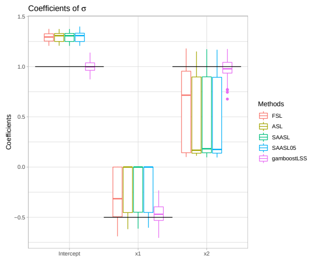
Figure B.5: Comparison: Estimated coefficients of μ



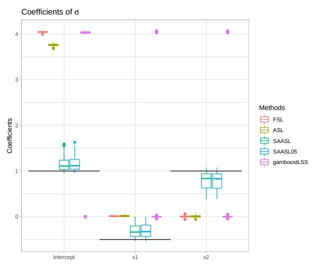
(a) Model 1a



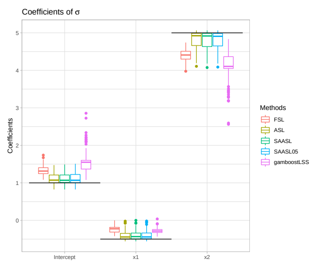
(b) Model 1b



(c) Model 1c

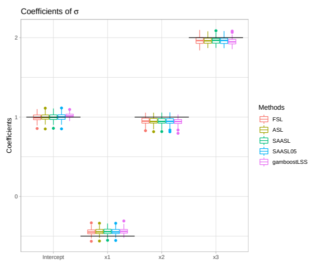


(d) Model 1d

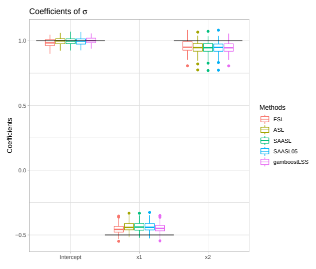


(e) Model 1e

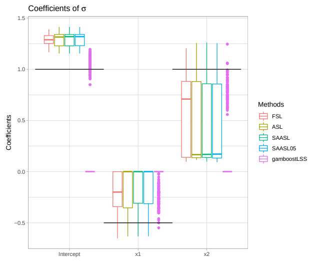
Figure B.6: Comparison: Estimated coefficients of σ



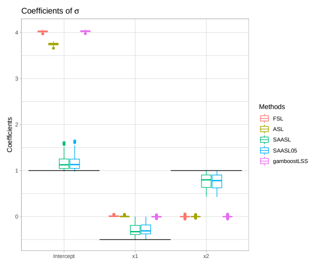
(a) Model 2a



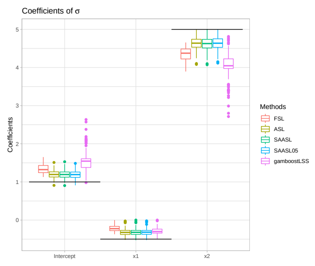
(b) Model 2b



(c) Model 2c

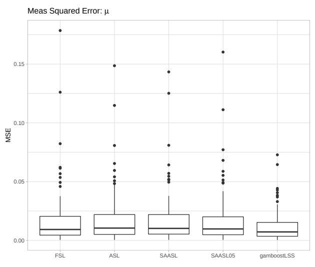


(d) Model 2d

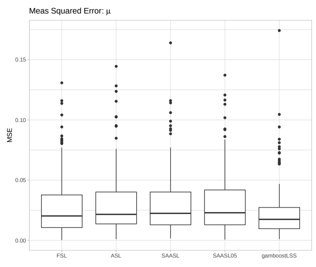


(e) Model 2e

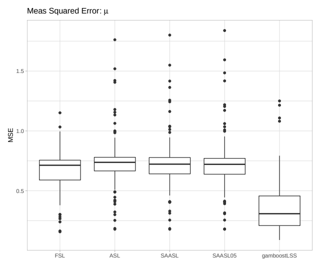
Figure B.7: Comparison: Estimated coefficients of σ



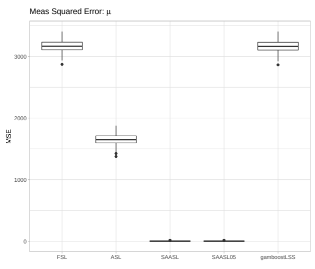
(a) Model 1a



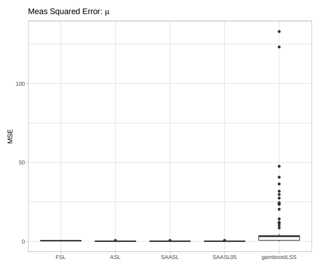
(b) Model 1b



(c) Model 1c

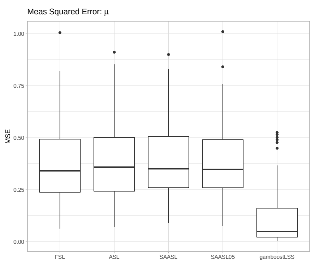


(d) Model 1d

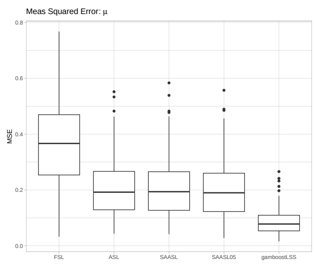


(e) Model 1e

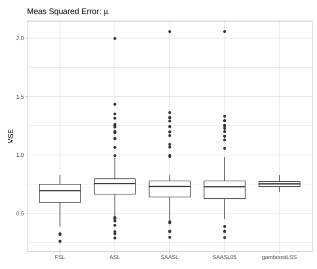
Figure B.8: Comparison: Mean Square Error (MSE) of μ



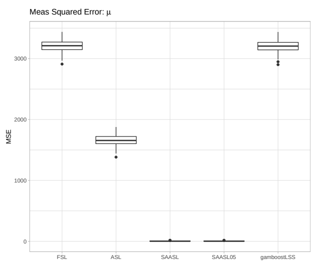
(a) Model 2a



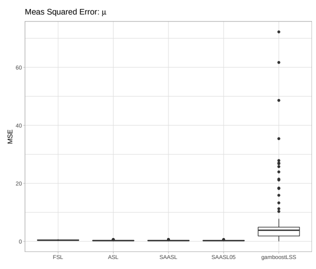
(b) Model 2b



(c) Model 2c

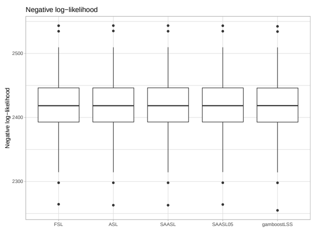


(d) Model 2d

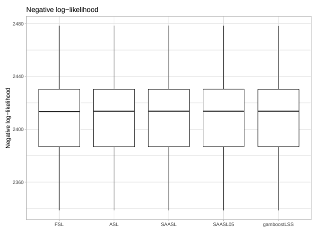


(e) Model 2e

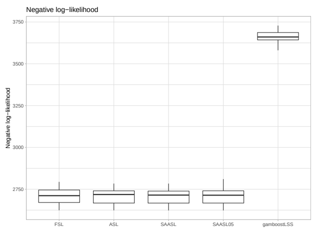
Figure B.9: Comparison: Mean Square Error (MSE) of μ



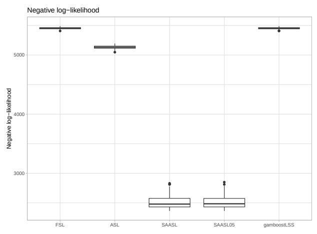
(a) Model 1a



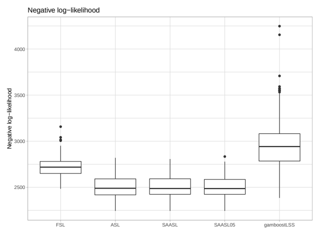
(b) Model 1b



(c) Model 1c

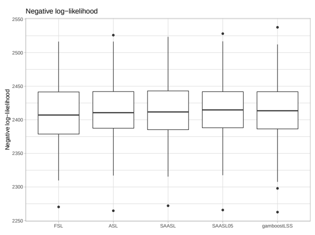


(d) Model 1d

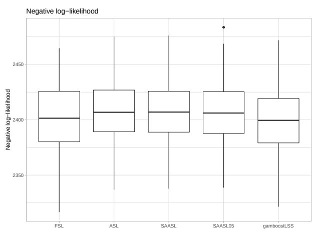


(e) Model 1e

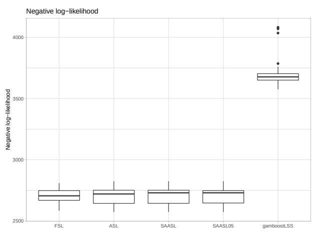
Figure B.10: Comparison: Empirical risk.



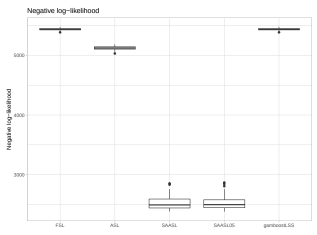
(a) Model 2a



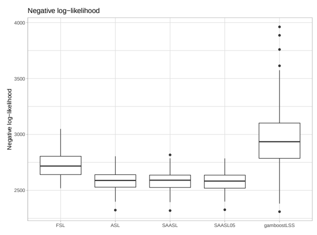
(b) Model 2b



(c) Model 2c



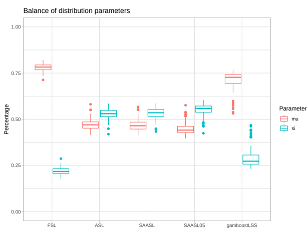
(d) Model 2d



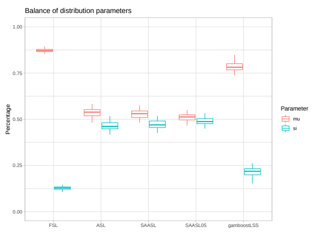
(e) Model 2e

Figure B.11: Comparison: Empirical risk.

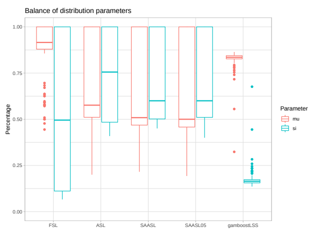
B.3 Balance of decisions



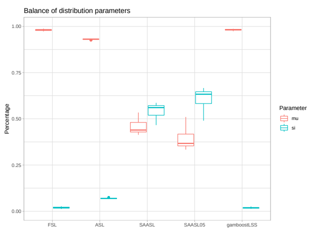
(a) Model 1a



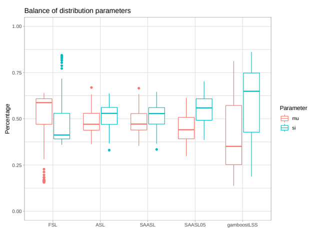
(b) Model 1b



(c) Model 1c

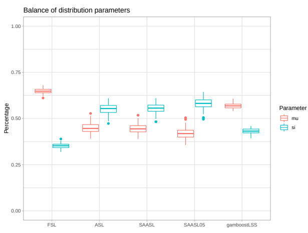


(d) Model 1d

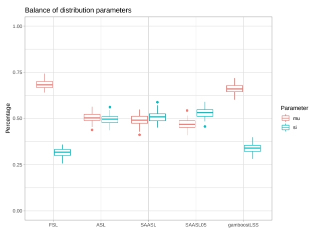


(e) Model 1e

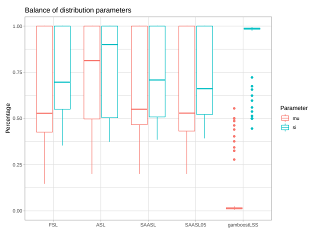
Figure B.12: Comparison: Balance of decisions.



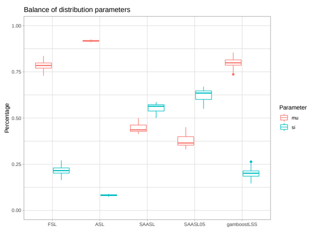
(a) Model 2a



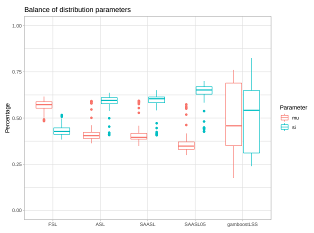
(b) Model 2b



(c) Model 2c



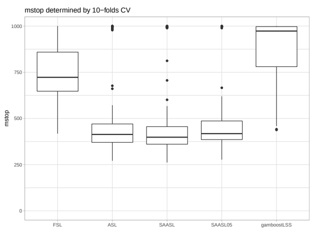
(d) Model 2d



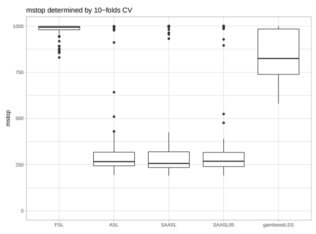
(e) Model 2e

Figure B.13: Comparison: Balance of decisions.

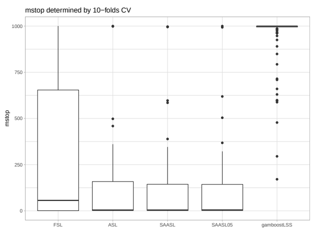
B.4 Computational costs



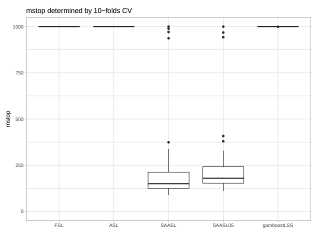
(a) Model 1a



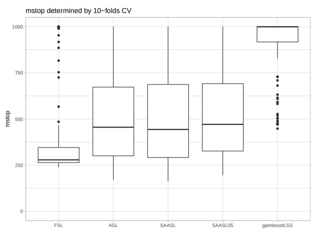
(b) Model 1b



(c) Model 1c

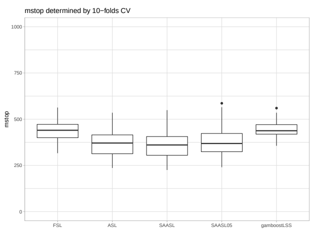


(d) Model 1d

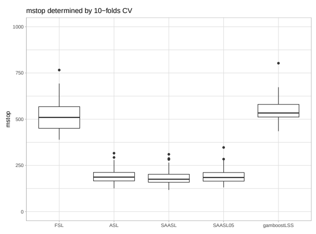


(e) Model 1e

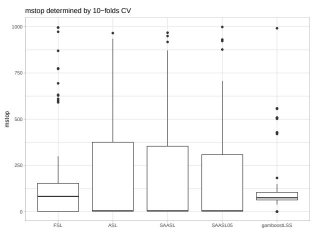
Figure B.14: Comparison: m_{stop}. The m_{stop} is specified by the 10-folds CV



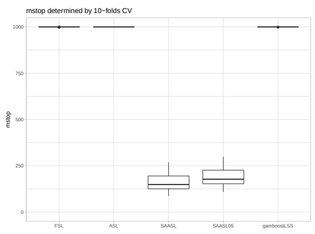
(a) Model 2a



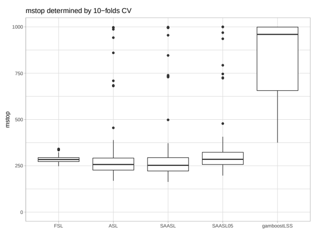
(b) Model 2b



(c) Model 2c

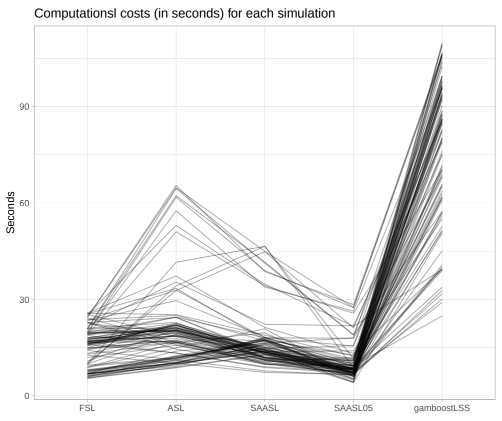


(d) Model 2d

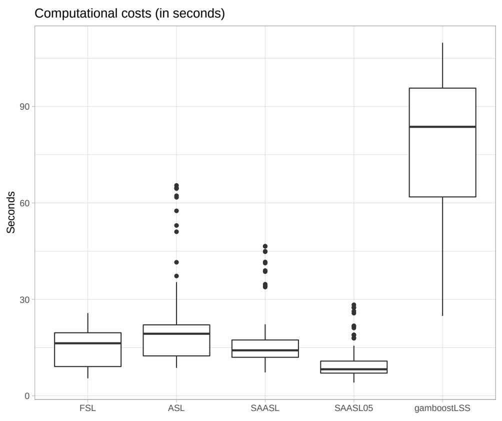


(e) Model 2e

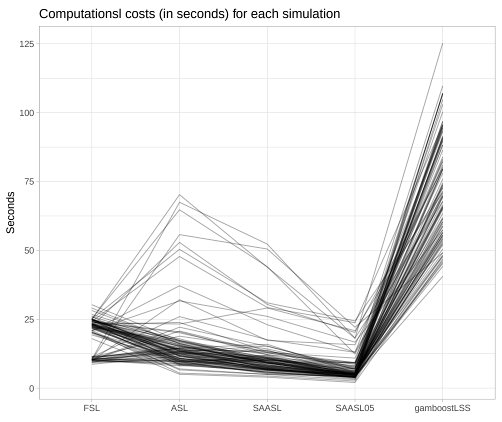
Figure B.15: Comparison: m_{stop}. The m_{stop} is specified by the 10-folds CV



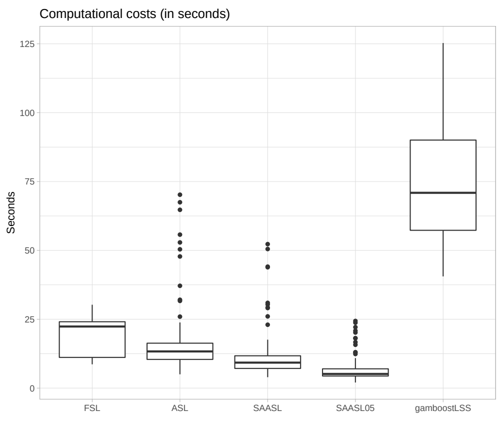
(a) Model 1a



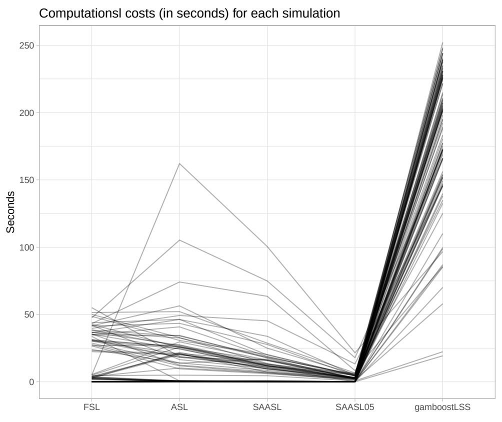
(b) Model 1a



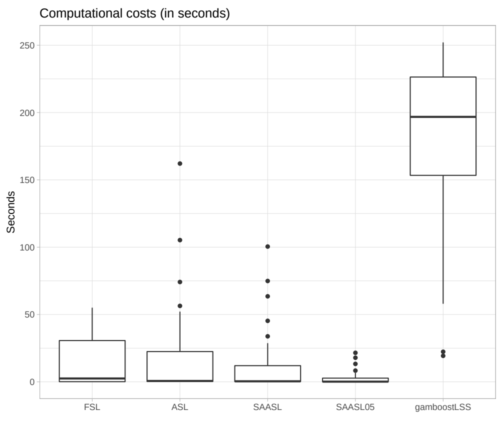
(c) Model 1b



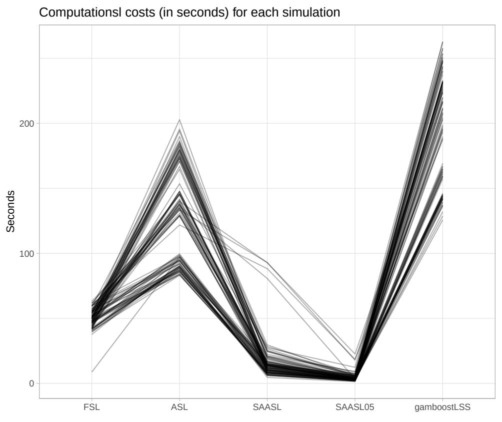
(d) Model 1b



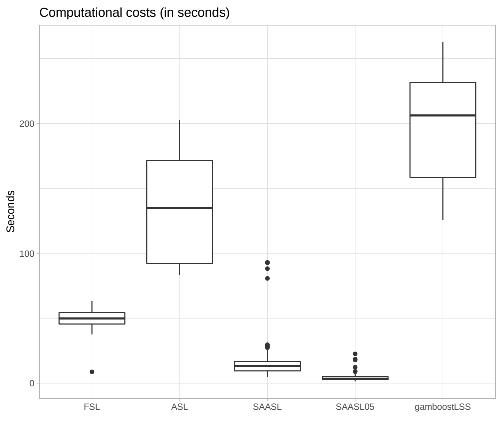
(e) Model 1c



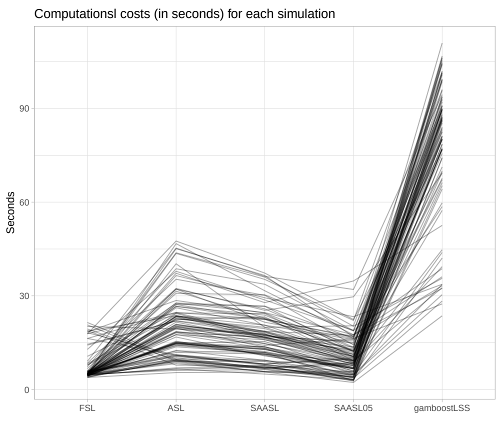
(f) Model 1c



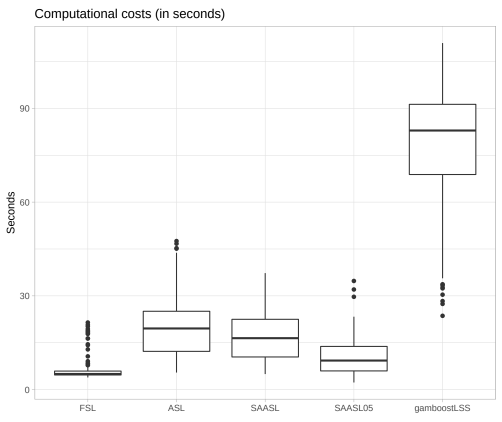
(g) Model 1d



(h) Model 1d

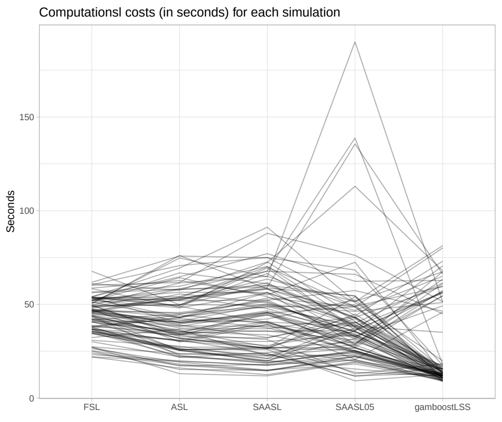


(i) Model 1e

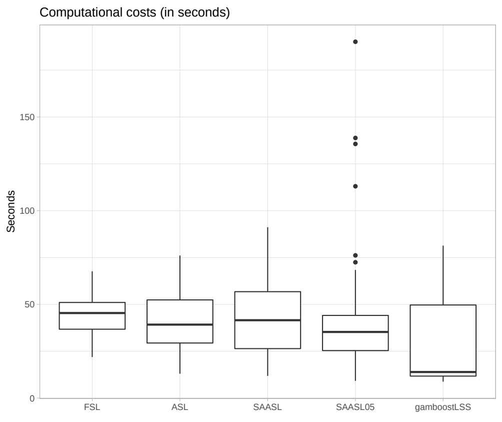


(j) Model 1e

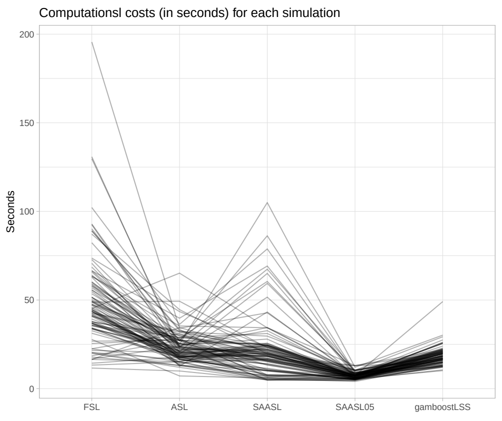
Figure B.16: Comparison: Computing time



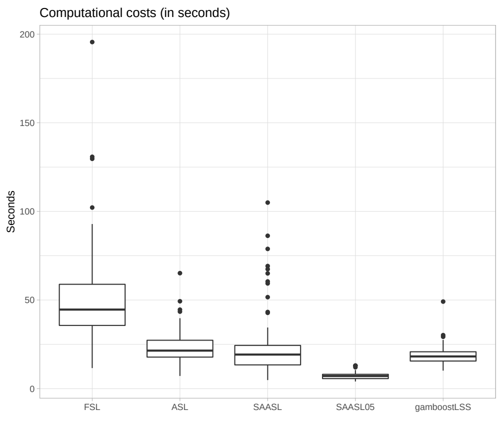
(a) Model 2a



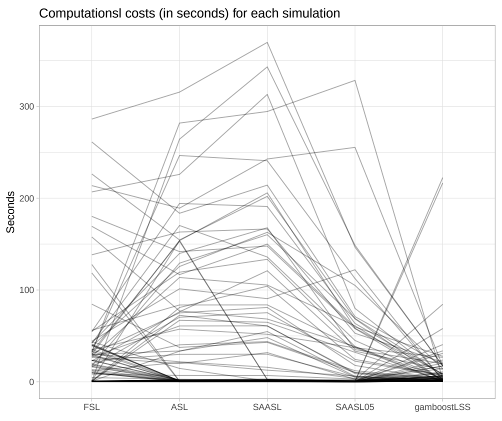
(b) Model 2a



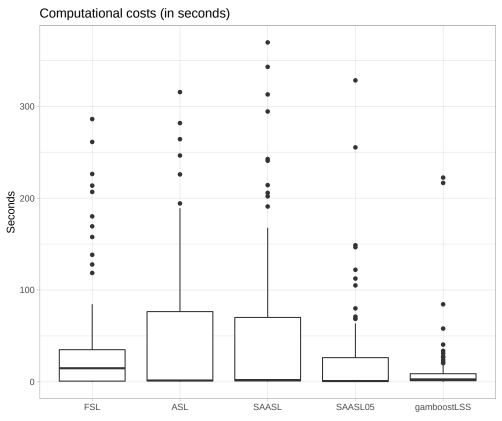
(c) Model 2b



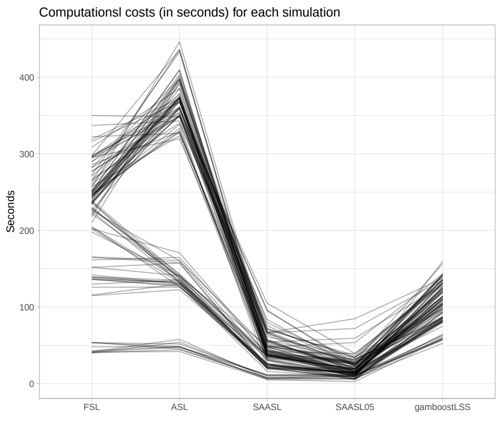
(d) Model 2b



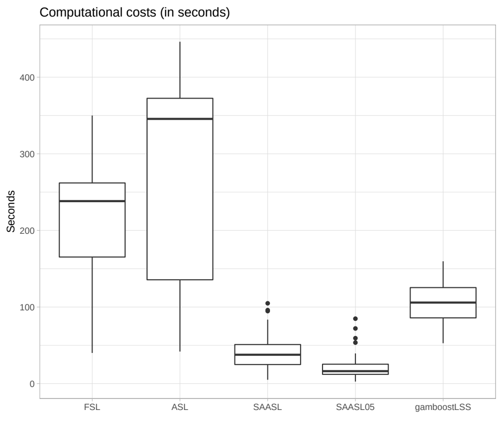
(e) Model 2c



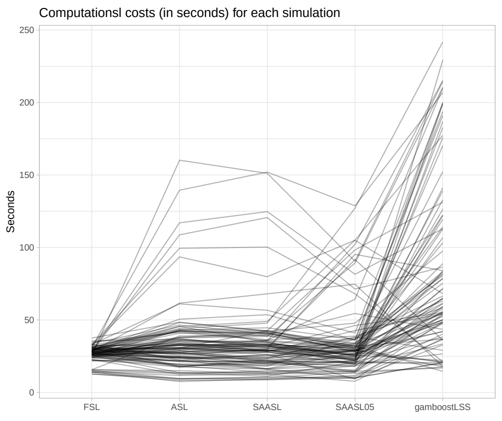
(f) Model 2c



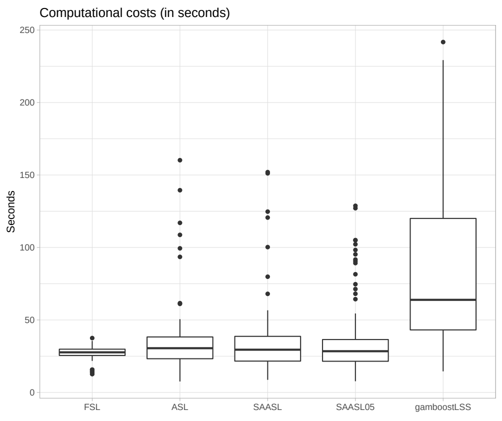
(g) Model 2d



(h) Model 2d



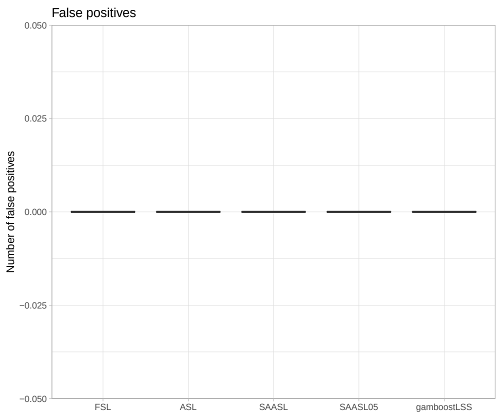
(i) Model 2e



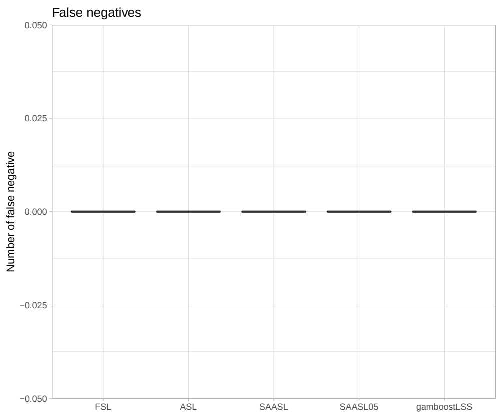
(j) Model 2e

Figure B.17: Comparison: Computing time

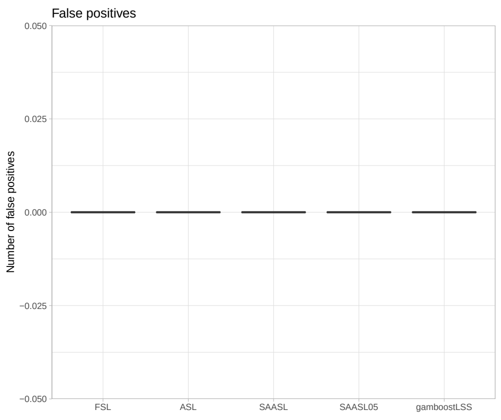
B.5 Overfitting



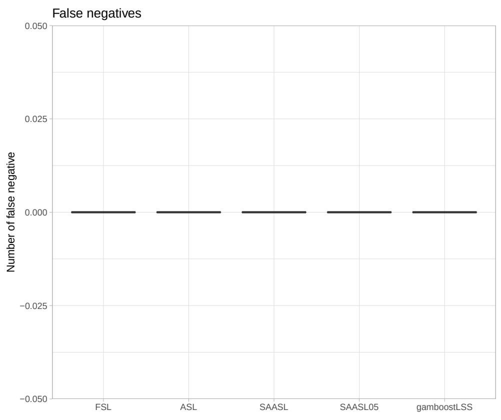
(a) Model 1a



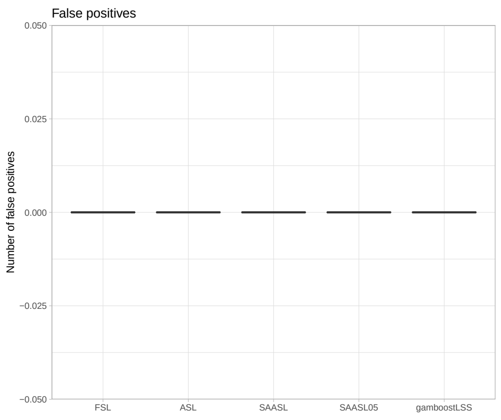
(b) Model 1a



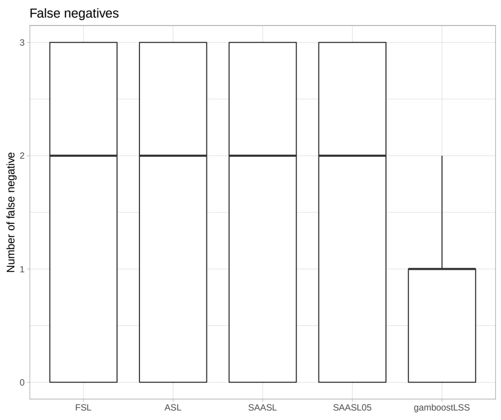
(c) Model 1b



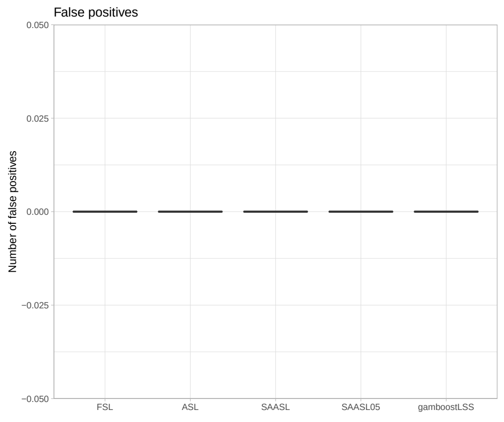
(d) Model 1b



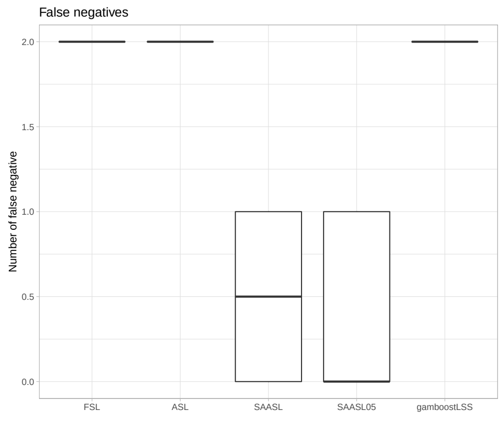
(e) Model 1c



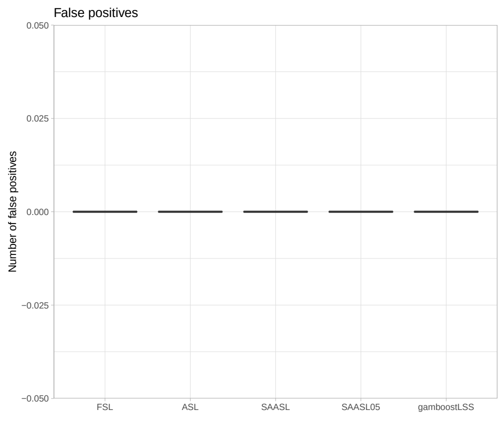
(f) Model 1c



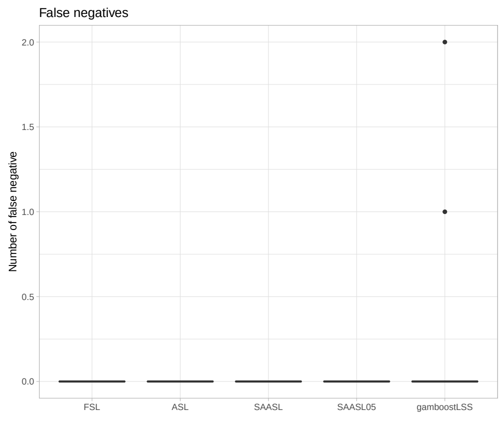
(g) Model 1d



(h) Model 1d

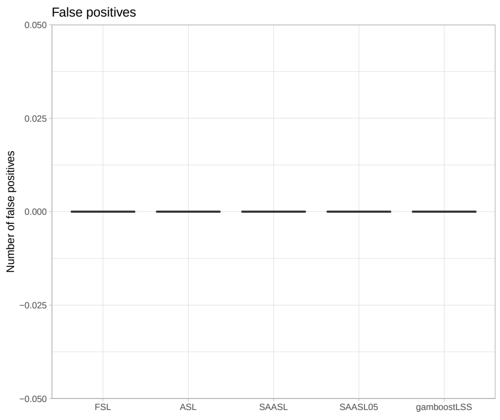


(i) Model 1e

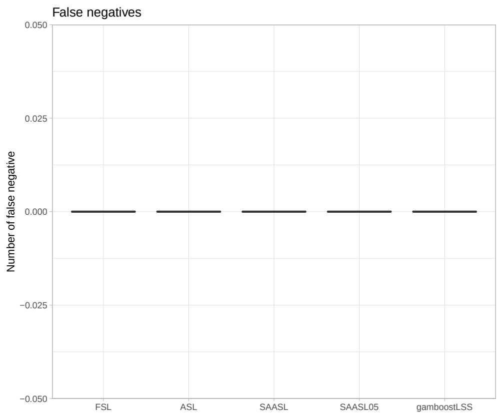


(j) Model 1e

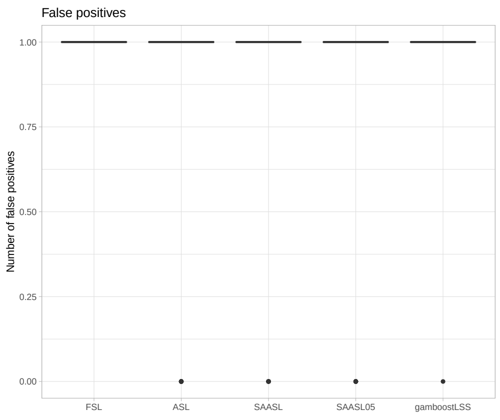
Figure B.18: Comparison: False positives & False negatives of μ



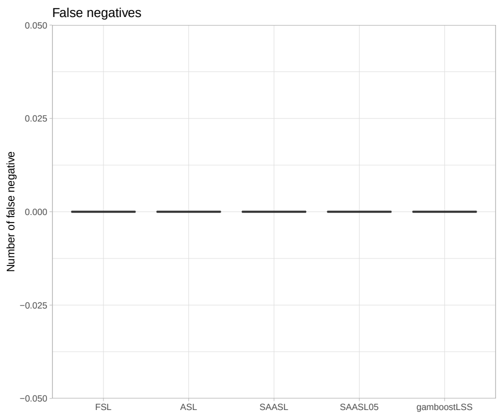
(a) Model 1a



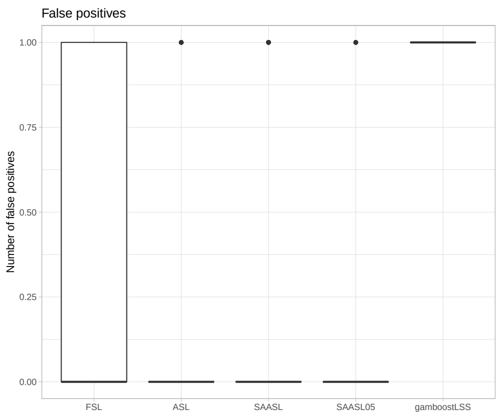
(b) Model 1a



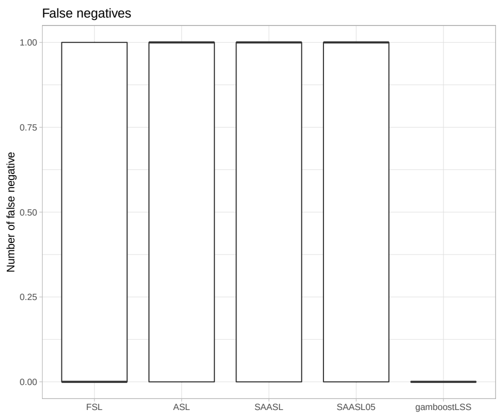
(c) Model 1b



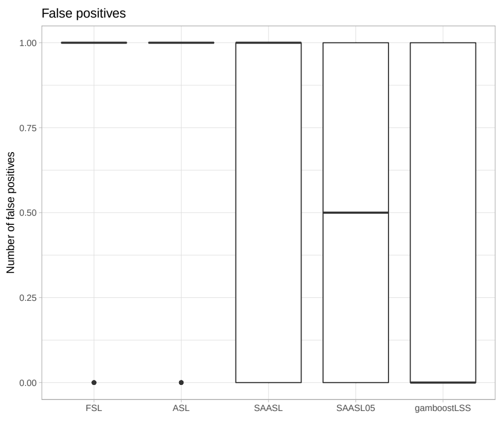
(d) Model 1b



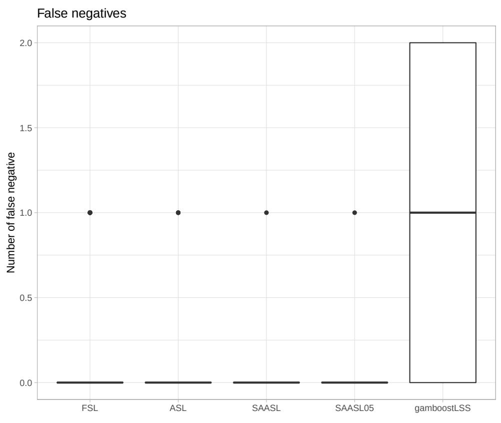
(e) Model 1c



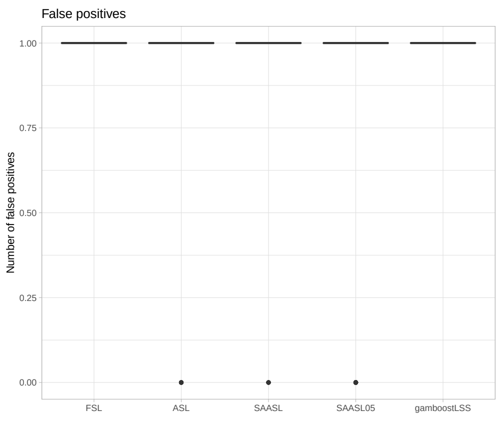
(f) Model 1c



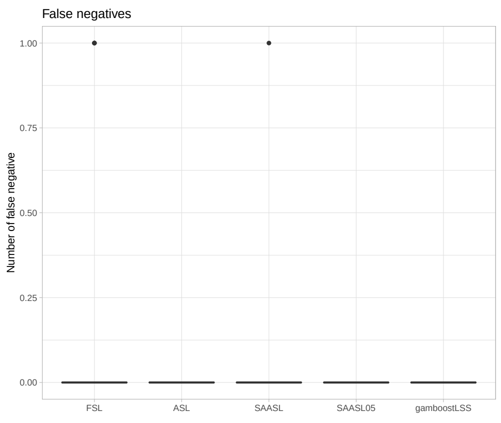
(g) Model 1d



(h) Model 1d

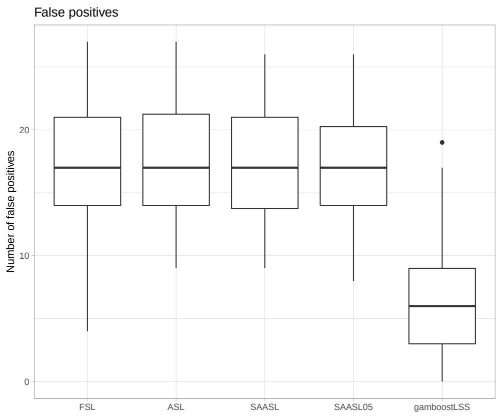


(i) Model 1e

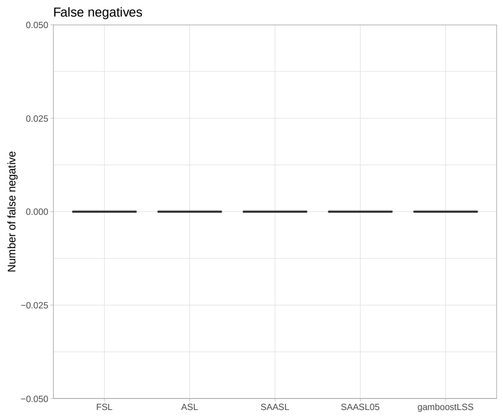


(j) Model 1e

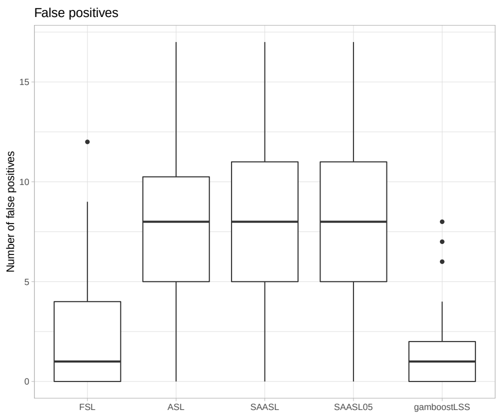
Figure B.19: Comparison: False positives & False negatives of σ



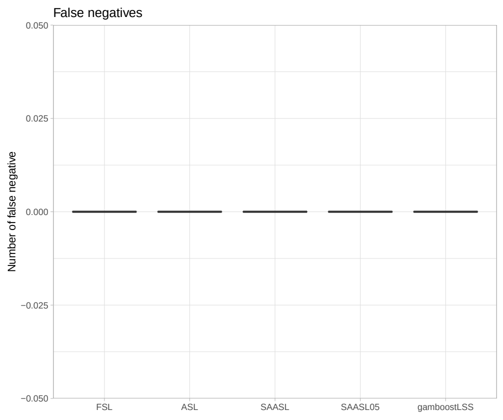
(a) Model 2a



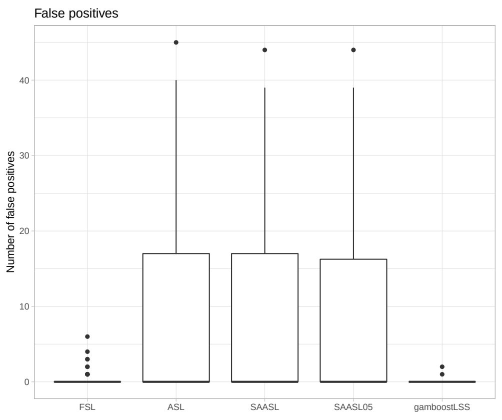
(b) Model 2a



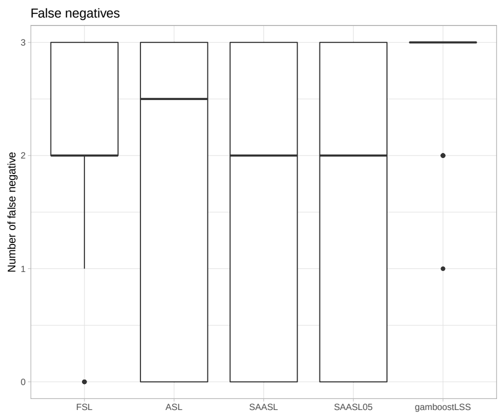
(c) Model 2b



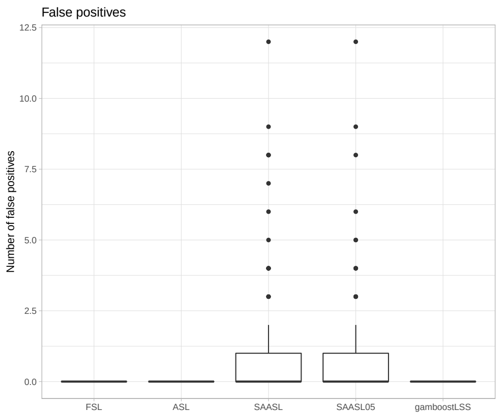
(d) Model 2b



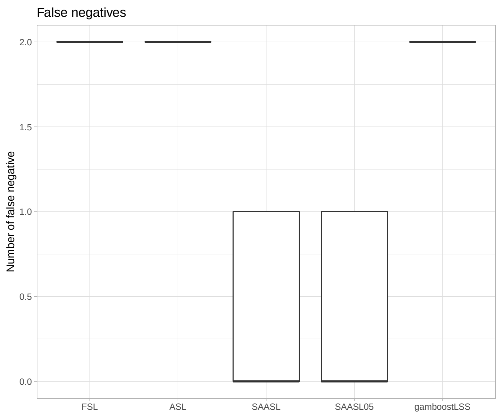
(e) Model 2c



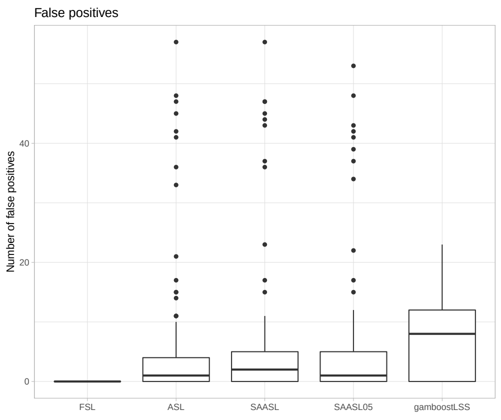
(f) Model 2c



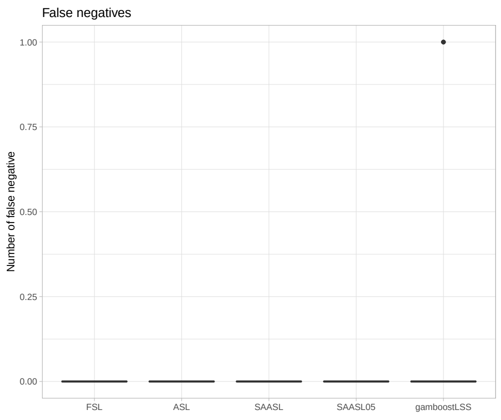
(g) Model 2d



(h) Model 2d

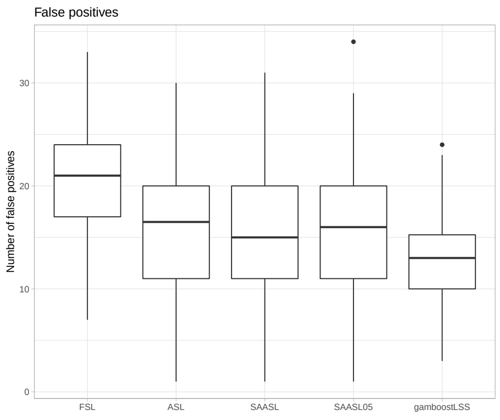


(i) Model 2e

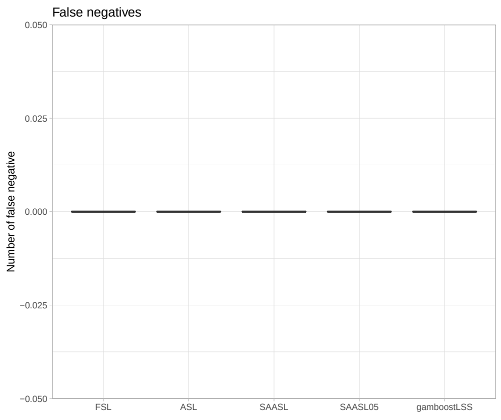


(j) Model 2e

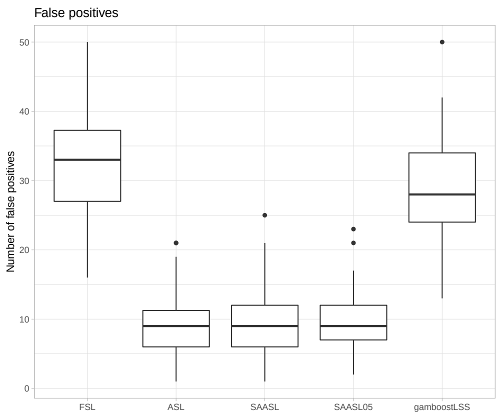
Figure B.20: Comparison: False positives & False negatives of μ



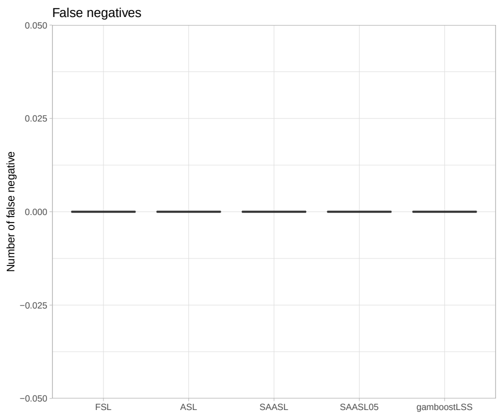
(a) Model 2a



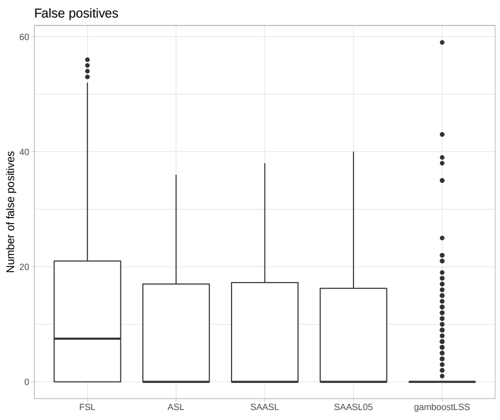
(b) Model 2a



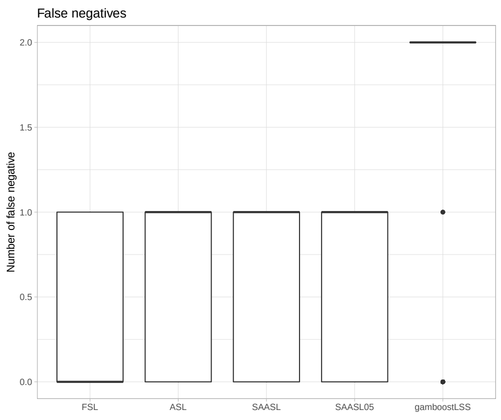
(c) Model 2b



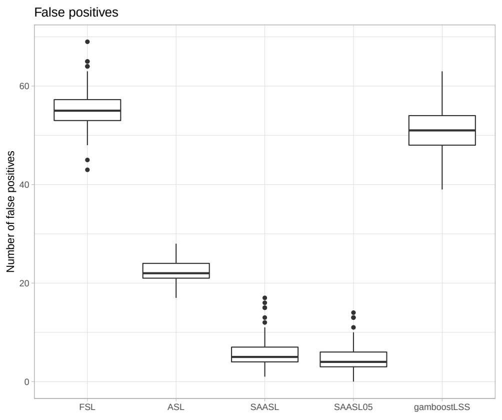
(d) Model 2b

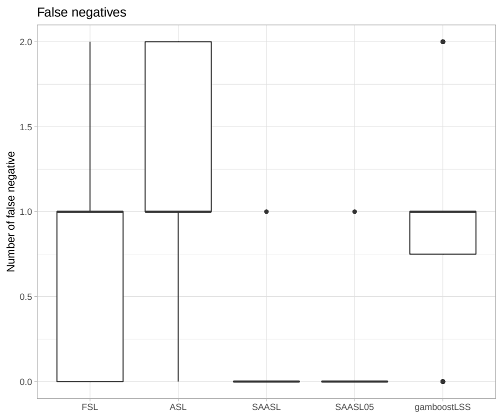


(e) Model 2c



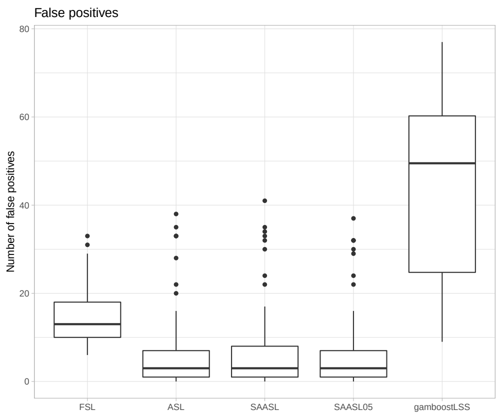
(f) Model 2c

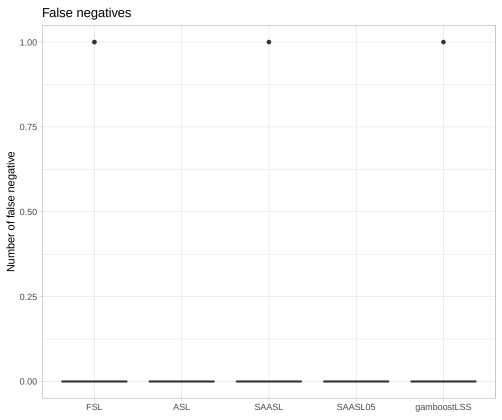




(g) Model 2d

(h) Model 2d





(i) Model 2e

(j) Model 2e

Figure B.21: Comparison: False positives & False negatives of σ

C Electronic Appendix

- An electronic form of this thesis.
- `R` folder:
  - `Methods` folder: contains the programs of each step length approaches.
  - `Simulation` folder: contains some simulations required by some programs in the `analysis` folder.
  - `Analysis` folder:
    - `utils.R`: contains the utility functions used for transforming data and creating graphics.
    - `simulation_study_overview.R`: creates the graphics required in section overview.
    - `simulation_study_specialCase.R`: creates the graphics required in section special cases.
    - programs begin with `bm`: are used for doing benchmark experiments. The name of each program indicate the types of the step lengths and the types of simulations. Their outputs are saved in `Output` folder.
    - programs begin with `ana_apx`: are used for analysing the outputs of benchmark experiments. All outputs are listed in appendix. The outputs of the program `ana_apx_mod_2b.R` as the main example have been explained in detail in the context of the thesis.
  - `Output` folder:
    - `Benchmark_server` folder: contains the .RData exported from the `bm` programs.
    - `Figure` folder: contains all graphics used in this thesis.

Declaration

I declare that I have developed and written the enclosed Master Thesis entirely by myself, and have not used sources or materials without declaration in the text. Any thoughts from others or literal quotations are clearly marked. This thesis was not used in the same or in a similar version to achieve academic grading or is being published elsewhere.

Boyao Zhang
May 9, 2019