Jan Ulbricht & Gerhard Tutz

Combining Quadratic Penalization and Variable Selection via Forward Boosting

Technical Report Number 099, 2011
Department of Statistics
University of Munich

http://www.stat.uni-muenchen.de

Combining Quadratic Penalization and Variable Selection via Forward Boosting

Jan Ulbricht & Gerhard Tutz

Ludwig-Maximilians-Universität München

Akademiestraße 1, 80799 München

{jan.ulbricht, gerhard.tutz}@stat.uni-muenchen.de

January 18, 2011

Abstract

Quadratic penalties can be used to incorporate external knowledge about the association structure among regressors. Unfortunately, they do not enforce single estimated regression coefficients to equal zero. In this paper we propose a new approach to combine quadratic penalization and variable selection within the framework of generalized linear models. The new method is called Forward Boosting and is related to componentwise boosting techniques. We demonstrate in simulation studies and a real-world data example that the new approach competes well with existing alternatives especially when the focus is on interpretable structuring of predictors.

Keywords: Generalized linear models, Penalized likelihood inference, Variable selection, Boosting techniques.

1 Introduction

Generalized linear models (GLMs) have been introduced by Nelder and Wedderburn (1972). This model class relaxes the strict linearity assumption of linear models by permitting the expected value of the response to depend on a smooth monotonic function of the linear predictor, the so called *link function.* GLMs allow for response distributions other than normal, while the predictor is still linear in the explanatory variables. As a result, the relationship between regressors and response can be usually interpreted very easily.

The primary purpose of GLMs is to estimate the influence of p regressors on the conditional expectation μ of the response based on a sample of n randomly drawn observations.

Due to technical progress the amount of data that can be observed has increased dramatically. Consequently, the number of possible regressors has done so as well. Data sets in proteomics may contain tens of thousands of genes which might influence the state of tumor cells. Even if it is possible to fit GLMs with such a huge number of regressors the effective dimension of the regressor space is often much smaller. So the question arises: which of the p regressors are really needed to explain the response?

A methodological approach that can deal with this feature are *shrinkage methods*. Their central idea is to shrink the coefficient estimates towards zero by a *penalty term*. If the penalty is properly chosen, one achieves variable selection. In many cases such penalty terms are based on the L_1 norm. The most prominent example is the lasso penalty (Tibshirani, 1996), that has been also extended to fused lasso (Tibshirani et al., 2005) or grouped lasso (Yuan and Lin, 2006).

The lasso has some deficiencies, in particular for strongly correlated designs. Combinations of the L_1 penalty and a quadratic term have been shown to yield better selection procedures. In particular the addition of a quadratic term can be used to include features with known association structure. The elastic net (Zou and Hastie, 2005) combines the L_1 penalty and the ridge penalty. It enforces the grouping effect, that is one obtains joint selection of highly correlated regressors, which as a whole group get similar coefficients or are set equal to zero, respectively. The weighted fusion penalty from Daye and Jeng (2009) combines the L_1 penalty with a term that enforces fusion of regressors steered by information on the correlation structure among them. Slawski et al. (2009) consider a more general form where the quadratic penalty can include temporal or spatial closeness. Although such penalties most often show good results when applied to simulated or real data, their major drawback is the necessity to determine two or even three tuning parameters. When using cross-validation this procedure can become quite time-consuming.

In this paper we propose an alternative strategy to enforce variable selection in quadratically penalized estimation problems for GLMs. The basic idea is to use a structured subset-based boosting algorithm that ensures the convergence to the corresponding quadratically penalized estimator while some regression coefficients are set exactly to zero before the iteration terminates. The resulting method is called *Forward Boosting*.

We start in Section 2 with a short overview on basic tools for maximum likelihood estimation in GLMs. Section 3 briefly examines quadratic penalties in GLMs when the focus is on likelihood inference. We consider the P-IRLS algorithm and show the derivation of degrees of freedom. In Section 4 we demonstrate how to combine quadratic penalties and boosting iterations. As a result we derive the ForwardBoost algorithm and some of its basic properties. Practical applications are given in Section 5 where we compare the performance of ForwardBoost with its major competitors by simulation settings and a real data set from chemometrics. The basic results are finally summarized and discussed in Section 6.

2 Definitions and Notations

Consider a random sample $\{(y_i, \mathbf{x}_i)\}_{i=1}^n$ that consists of n observed independent realizations of a response y and of p regressors, the latter contained in the vector $\mathbf{x}_i = (1, x_{i1}, \ldots, x_{ip})^\top$. For notational convenience, we collect the regressors in the regressor matrix $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^\top$.

The conditional distribution of $y_i|\mathbf{x}_i$ belongs to a simple exponential family. That is, its density can be written

$$f(y_i|\theta_i, \phi, \omega_i) = \exp\left\{\frac{y_i\theta_i - b(\theta_i)}{\phi}\omega_i + c(y_i, \phi)\right\}, \tag{1}$$

where θ_i is the natural parameter, b, c are known functions, and ω_i is a known prior weight that varies between observations. We assume the dispersion parameter $\phi > 0$ to be constant for all y_i. A GLM utilizes the structural relationship

$$g(\mu_i) = \mathbf{x}_i^\top \mathbf{b} = \eta_i, \tag{2}$$

satisfying (i) $\mu_i = \mathbb{E}(y_i|\mathbf{x}_i) = db(\theta_i)/d\theta$, (ii) g is an injective and twicely differentiable link function with $g^{-1} = h$ and (iii) $\mathbf{b} = (\beta_0, \boldsymbol{\beta}^\top)^\top$ is the $(p+1)$-dimensional vector of coefficients, where β_0 represents the constant part and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^\top$ contains the weights of the p regressors. It holds that $\theta_i = \psi \circ h = \theta_i(\mathbf{b})$, where $\psi = (db(\theta_i)/d\theta)^{-1}$. Due to this relation the unknown parameter $\mathbf{b}$ also determines the natural parameters. Assuming ϕ to be known for the moment, the log-likelihood is

$$\ell(\mathbf{b}) = \ell(\mathbf{b}|\mathbf{y}, \mathbf{X}) = \sum_{i=1}^n \left\{\frac{y_i\theta_i(\mathbf{b}) - b \circ \theta_i(\mathbf{b})}{\phi}\omega_i + c(y_i, \phi)\right\}. \tag{3}$$

The gradient of (3) is denoted as score vector

$$\mathbf{s}(\mathbf{b}) = \frac{\partial \ell(\mathbf{b})}{\partial \mathbf{b}} = \frac{1}{\phi}\mathbf{X}^\top \mathbf{D} \boldsymbol{\Sigma}^{-1}(\mathbf{y} - \boldsymbol{\mu}), \tag{4}$$

where $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)^\top$, $\mathbf{D} = \mathbf{D}(\mathbf{b}) = \operatorname{diag}\{dh(\eta_1)/d\eta, \ldots, dh(\eta_n)/d\eta\}$ and $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}(\mathbf{b}) = \operatorname{diag}\left\{\omega_1^{-1}V(\mu_1), \ldots, \omega_n^{-1}V(\mu_n)\right\}$ with $V(\mu_i) = d^2b(\theta_i)/d\theta^2$. Another quantity that corresponds to the log-likelihood is the Fisher information matrix

$$\mathbf{F}(\mathbf{b}) = \frac{1}{\phi}\mathbf{X}^\top \mathbf{W} \mathbf{X}, \tag{5}$$

where $\mathbf{W} = \mathbf{D}\boldsymbol{\Sigma}^{-1}\mathbf{D}$.

Usually, maximum likelihood estimators (MLEs) of $\mathbf{b}$ are computed iteratively as solutions of the nonlinear likelihood equations $\mathbf{s}(\mathbf{b}) = \mathbf{0}$, which corresponds to local maxima of $\ell(\mathbf{b})$, provided the Hessian is negative definite. The update has the structure of a weighted or generalized least squares estimator. Therefore the MLE can be obtained by an iteratively reweighted least squares (IRLS) estimation scheme.

3 Quadratic Penalties

There are two scenarios that often appear when the focus is on variable selection:

(i) the $n \ll p$ case, that is much more regressors than observations are available, some of the regressors are irrelevant or insignificant,

(ii) the presence of multicollinearity, that is some regressors carry redundant information. This often leads to ill-conditioned estimation problems.

An important family of penalty terms that can handle these scenarios are *quadratic penalties.* Usually they achieve a stable fit even in the presence of highly correlated regressors. In addition they can be used to include structural information.
A *quadratic penalty* is defined as

$$P_\lambda(\boldsymbol{\beta}) = \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{M}_\lambda \boldsymbol{\beta}, \tag{6}$$

where $\mathbf{M}_\lambda$ is a symmetric positive definite matrix that depends on a vector $\boldsymbol{\lambda}$ of nonnegative tuning parameters. $\mathbf{M}_\lambda$ is denoted as *penalty matrix.* Most often $\boldsymbol{\lambda}$ reduces to a single scalar. Some examples of quadratic penalties are the *ridge penalty*

$$P_\lambda^r(\boldsymbol{\beta}) = \frac{1}{2}\boldsymbol{\beta}^\top \operatorname{diag}\{\lambda 1_p\}\,\boldsymbol{\beta}, \tag{7}$$

where 1_p denotes the p-dimensional vector of ones, the *adaptive ridge penalty*

$$P_\lambda^{ar}(\boldsymbol{\beta}) = \frac{1}{2}\boldsymbol{\beta}^\top \operatorname{diag}\{\lambda_1, \dots, \lambda_p\}\,\boldsymbol{\beta}, \tag{8}$$

the *correlation-based penalty* as introduced in Tutz and Ulbricht (2009)

$$P_\lambda^{cb}(\boldsymbol{\beta}) = \frac{\lambda}{2}\sum_{i=1}^{p}\sum_{i<j}\left\{\frac{(\beta_i - \beta_j)^2}{1-\varrho_{ij}} + \frac{(\beta_i + \beta_j)^2}{1+\varrho_{ij}}\right\}, \quad |\varrho_{ij}| < 1, \tag{9}$$

where ϱ_{ij} denotes the (empirical) correlation between the i-th and the j-th regressor. Another example is the *correlation-driven penalty*

$$P_\lambda^{cd}(\boldsymbol{\beta}) = \frac{\lambda}{p}\sum_{i<j}\omega_{ij}\left\{\beta_i - \operatorname{sgn}(\varrho_{ij})\beta_j\right\}^2, \tag{10}$$

where $\omega_{ij} \geq 0$ are chosen weights. It can be linked to the correlation-based penalty by choosing the weights in dependence on the correlation. The correlation-driven penalty has been introduced by Daye and Jeng (2009) as linear combination with the L_1 penalty, yielding the *weighted fusion penalty.*

From a geometrical point of view, the matrix $\mathbf{M}_\lambda$ might be chosen to emphasize directions in the parameter space that align with the larger eigendirections of the empirical covariance matrix of the regressors. Figure 1 illustrates the solution paths of four different

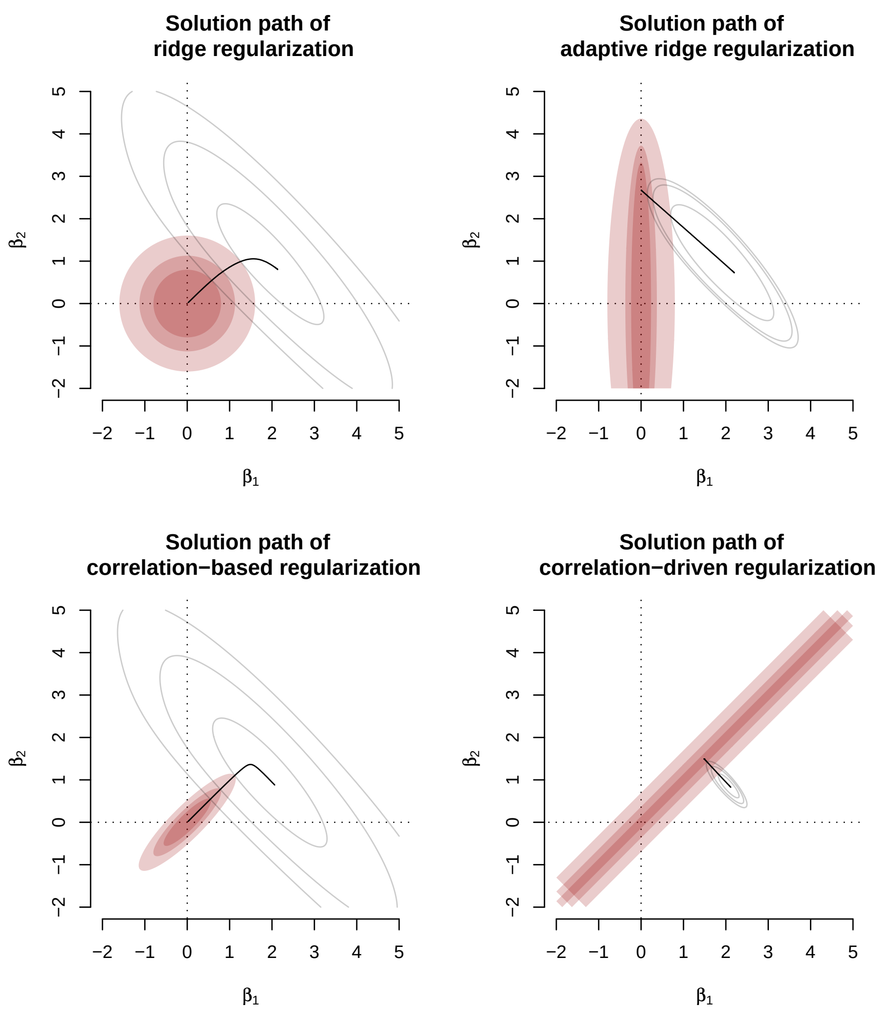


Figure 1: Solution paths of four different quadratic penalties for a GLM with Gamma response, log-link and dispersion parameter $\phi = 1/2$.

quadratic penalties for a GLM with Gamma response, log-link and dispersion parameter $\phi = 1/2$. The model contains an intercept and two regressors x_1 and x_2 with correlation $\varrho_{12} = 0.9$, the sample size is $n = 30$. It is seen that the type of the quadratic penalty strongly influences the solution path. The ridge penalty in the upper left panel does not give priority to any direction. The adaptive ridge penalty in the upper right panel uses fixed $\lambda_2 = 0.5$ yielding strong emphasis on the ordinate. For $\lambda_1 \to \infty$ the component $\hat{\beta}_2$ converges towards 2.671, while $\hat{\beta}_1 \to 0$. The correlation-based penalty in the lower left panel focuses on the bisecting line where its sign is driven by the sign of the correlation between the regressors. The solution path of the correlation-based penalty is similar to that of the ridge penalty with a more strong emphasis on forcing the regressor coefficients to be equal. The correlation-driven penalty (10) (lower right panel) uses weights $\omega_{ij} = |\varrho_{ij}|^5/(1 - |\varrho_{ij}|)$. This penalty forces the coefficients to be equal (up to sign).

We define

$$\mathbf{M}^*_\lambda = \begin{bmatrix} 0 & \mathbf{0}_p^\top \\ \mathbf{0}_p & \mathbf{M}_\lambda \end{bmatrix}, \tag{11}$$

where $\mathbf{0}_p$ is the p-dimensional null vector, in order to adjust the dimension of the penalty matrix to the dimension $p+1$ of the unknown coefficient vector $\mathbf{b}$.

Given a quadratic penalty, the resulting penalized MLE of $\mathbf{b}$ is the solution of the problem

$$\hat{\mathbf{b}} = \arg\min_{\mathbf{b}} \left\{ -\ell(\mathbf{b}) + \frac{1}{2}\mathbf{b}^\top \mathbf{M}^*_\lambda \mathbf{b} \right\}. \tag{12}$$

Since the quadratic penalty (6) is twice differentiable, iterative Newton-type methods can be used to solve (12). Let $\mathbf{H}$ denote the Hessian of the log-likelihood (3). Using $\mathbf{F}(\mathbf{b}) = -\mathbb{E}[\mathbf{H}(\mathbf{b})]$ as an approximation, the update of the penalized Fisher scoring is

$$\begin{aligned} \mathbf{b}_{(k+1)} &= \mathbf{b}_{(k)} - \left\{ \mathbf{F}(\mathbf{b}_{(k)}) + \mathbf{M}^*_\lambda \right\}^{-1} \left\{ -\mathbf{s}(\mathbf{b}_{(k)}) + \mathbf{M}^*_\lambda \mathbf{b}_{(k)} \right\} \\ &= \left(\mathbf{X}^\top \mathbf{W}_{(k)} \mathbf{X} + \mathbf{M}^*_\lambda \right)^{-1} \mathbf{X}^\top \mathbf{W}_{(k)} \tilde{\mathbf{y}}_{(k)}, \end{aligned}$$

where $\mathbf{W}_{(k)} = \mathbf{W}(\mathbf{b}_{(k)}), \mathbf{D}_{(k)} = \mathbf{D}(\mathbf{b}_{(k)}), \boldsymbol{\mu}_{(k)} = \left\{ h(\mathbf{x}_1^\top \mathbf{b}_{(k)}), \ldots, h(\mathbf{x}_n^\top \mathbf{b}_{(k)}) \right\}^\top$. The dispersion parameter ϕ does not cancel out as in classical Fisher scoring. Since the tuning parameter $\boldsymbol{\lambda}$ has been treated as given we simply adjust the penalty matrix $\mathbf{M}_\lambda$ for ϕ, e.g. we assume that $\boldsymbol{\lambda}$ already includes the (unknown) dispersion parameter, and hence drop it in the estimation equations.

For $\mathbf{M}_\lambda = \mathbf{O}_{(p\times p)}$ the algorithm coincides with the IRLS algorithm. Hence, penalized Fisher scoring can be interpreted as a generalization of it. One crucial point for the application is the positive definiteness of $\mathbf{F}(\mathbf{b}_{(k)}) + \mathbf{M}^*_\lambda$. As $\mathbf{M}_\lambda$ is assumed to be already positive definite this condition is fulfilled even for a number of situations where $n \ll p$.

The tuning parameter $\boldsymbol{\lambda}$, which controls model complexity, must be determined. Consider for example the ridge penalty (7). When $\lambda = 0$ the model includes $p+1$ regression coefficients which typically are different from zero whereas in the limit $\lambda \to \infty$ the intercept β_0 is the only nonzero coefficient. If we regard the resulting estimate as a function of λ, the number of 'relevant' regression coefficients is also a function of λ. The value of the regularization parameter controls the amount of shrinkage *and* the amount of model complexity. Since model complexity of parametric models is related to the number of (regression) parameters it is often expressed in terms of degrees of freedom. In the classical linear model, the degrees of freedom are computed as the trace of the hat matrix. For quadratically penalized GLMs the hat matrix can be derived similar to unpenalized GLMs, see for example Fahrmeir and Tutz (2001). One obtains the form

$$\mathbf{H}_\lambda = \mathbf{W}^{\top/2} \mathbf{X} (\mathbf{X}^\top \mathbf{W} \mathbf{X} + \mathbf{M}^*_\lambda)^{-1} \mathbf{X}^\top \mathbf{W}^{1/2}. \tag{13}$$

The trace of the hat matrix, $\mathrm{df}(\boldsymbol{\lambda}) = \mathrm{tr}(\mathbf{H}_\lambda)$ is used as the degrees of freedom. As proposed by Wood (2006) we estimate the unknown dispersion parameter ϕ by the Pearson-like dispersion estimator

$$\hat{\phi} = \frac{1}{n - \mathrm{tr}(\mathbf{H}_\lambda)} \sum_{i=1}^{n} \frac{(y_i - \hat{\mu}_i)^2}{V(\hat{\mu}_i)}. \tag{14}$$

However, quadratically penalized MLEs cannot set some components of $\mathbf{b}$ to equal zero. Hence, our idea is to consider a specialized version of componentwise boosting in order to combine improvements by quadratic penalization with variable selection.

4 Forward Boosting

Although they can be used to tailor the shrinkage effect simple quadratically penalized MLEs do not set components of $\mathbf{b}$ to exactly zero. For this reason we will use them in the following to structure the learner of a specialized boosting method.

Boosting is a successful and flexible strategy which has been originally developed in the machine learning community. A starting point for our investigations has been componentwise boosting that merely improves one selected coefficient within one step, see for example Friedman et al. (2008), Bühlmann and Yu (2006), Bühlmann and Hothorn (2007), and Tutz and Binder (2007). One basic concept in boosting is the use of a weak learner. A weak learner is a fitting procedure that starts from previous estimates and improves the fit only weakly. The final estimate is obtained by applying the stepwise fit repeatedly. In the following we first consider how quadratic penalties work if used directly as a learner and then show how the learner has to be modified to obtain the intended structuring effect.

Consider a structured weak learner that uses a quadratic penalty of the form $P_\lambda(\boldsymbol{\beta}) = \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{M}_\lambda \boldsymbol{\beta}$, for example the correlation-based penalty (9). Let $\mathbf{x}_{\mathcal{A}_{(l)}}$ denote a subset of covariates (including the intercept) that is attained in the l-th iteration of a boosting algorithm, and $\boldsymbol{\gamma}_{\mathcal{A}_{(l)}}$ the corresponding (sub-)vector of coefficients. Boosting improves the fit within one step by fitting the model

$$\mu = h\left(\hat{\eta}^{(l-1)} + \mathbf{x}_{\mathcal{A}_{(l)}}^\top \boldsymbol{\gamma}_{\mathcal{A}_{(l)}}\right), \tag{15}$$

where the predictor $\hat{\eta}^{(l-1)} = \mathbf{x}^\top \hat{\mathbf{b}}_{(l-1)}$ as estimated from the previous step is treated as an offset, or more precisely as a fixed constant. The question is how to estimate the update $\boldsymbol{\gamma}_{\mathcal{A}_{(l)}}$ when one wants to account for the assumed association structure as represented in the quadratic penalty.

As common in componentwise likelihood-based boosting, one might update by utilizing one-step Fisher scoring with $\hat{\mathbf{b}}_{(l-1)}$ as initial value. Based on the training data set $\{(y_i, \mathbf{x}_i)\}_{i=1}^n$ one obtains

$$\hat{\boldsymbol{\gamma}}_{\mathcal{A}_{(l)}} = (\mathbf{X}_{\mathcal{A}_{(l)}}^\top \mathbf{W}(\hat{\boldsymbol{\eta}}^{(l-1)})\mathbf{X}_{\mathcal{A}_{(l)}} + \mathbf{M}_{\lambda,\mathcal{A}_{(l)}}^*)^{-1}\mathbf{X}_{\mathcal{A}_{(l)}}^\top \mathbf{D}(\hat{\boldsymbol{\eta}}^{(l-1)})\boldsymbol{\Sigma}^{-1}(\hat{\boldsymbol{\eta}}^{(l-1)})(\mathbf{y} - \hat{\boldsymbol{\mu}}^{(l-1)}),$$

where $\hat{\boldsymbol{\eta}}^{(l-1)} = (\hat{\eta}_1^{(l-1)}, \ldots, \hat{\eta}_n^{(l-1)})^\top$, $\hat{\eta}_i^{(l-1)} = \mathbf{x}_i^\top \hat{\mathbf{b}}_{(l-1)}$, $\hat{\boldsymbol{\mu}}^{(l-1)} = \{h(\hat{\eta}_1^{(l-1)}), \ldots, h(\hat{\eta}_n^{(l-1)})\}^\top$ and $\mathbf{M}_{\lambda,\mathcal{A}_{(l)}}^*$ contains the submatrix of $\mathbf{M}_\lambda^*$ that corresponds to the indices in $\mathcal{A}_{(l)}$. The main problem with this approach is its restriction to the elements in $\mathcal{A}_{(l)}$ and the consequences when the partition $\mathbf{M}_{\lambda,\mathcal{A}_{(l)}}^*$ is used instead of the full penalty matrix. Consider for example the common case of componentwise boosting where $|\mathcal{A}_{(l)}| = 1$ for all l. Then the partition of $\mathbf{M}_\lambda^*$ consists of a single element. This is algebraically equivalent to using componentwise ridge boosting, no matter what penalty we have originally chosen. Hence, the assumed association structure will not be taken into account. Alternatively, consider the application of the correlation-based penalty even when $|\mathcal{A}_{(l)}| > 1$. What happens when the corresponding elements of $\mathbf{M}_\lambda^*$ are used is that covariates which are strongly correlated with others are much more penalized then only slightly correlated ones. As a result, the potential update $\hat{\boldsymbol{\gamma}}_{\mathcal{A}_{(l)}}$ is much more closer to zero when $\mathcal{A}_{(l)}$ consists of highly correlated covariates. This in turn lowers the model fitting ability, and hence reduces the possibility to get chosen for a coefficient update. Consequently, this would facilitate uncorrelated covariates to enter the model much easier than highly correlated ones. But that contradicts the motivation behind the correlation-based penalty.

As a consequence, one has to find a way to incorporate the assumed structure among *all* covariates into the penalty term. One approach to find an appropriate update is

$$\hat{\boldsymbol{\gamma}}_{(l)} = \mathbf{I}_{\mathcal{A}_{(l)}}(\mathbf{X}^\top \mathbf{W}(\hat{\boldsymbol{\eta}}^{(l-1)})\mathbf{X} + \mathbf{M}_\lambda^*)^{-1}\mathbf{X}^\top \mathbf{D}(\hat{\boldsymbol{\eta}}^{(l-1)})\boldsymbol{\Sigma}^{-1}(\hat{\boldsymbol{\eta}}^{(l-1)})(\mathbf{y} - \hat{\boldsymbol{\mu}}^{(l-1)}), \tag{16}$$

where the full vector $\hat{\boldsymbol{\gamma}}_{(l)}$ is used and one just selects the elements to be updated with the diagonal matrix $\mathbf{I}_{\mathcal{A}_{(l)}}$ where ones are located at the positions corresponding to the elements of $\mathcal{A}_{(l)}$ and zeros elsewhere. Then the whole assumed association structure among all regressors is regarded, but unfortunately one simply incorporates the association structure left in $\hat{\boldsymbol{\eta}}^{(l-1)}$ and $\hat{\boldsymbol{\mu}}^{(l-1)}$, but not the complete structure underlying the original training data. Consequently, using a quadratic penalty solely based on the current update estimate $\hat{\boldsymbol{\gamma}}_{(l)}$ cannot adjust for the magnitudes of $\hat{\mathbf{b}}_{(l-1)}$. Some evident problems arise e.g. when we apply the correlation-based penalty. With the current approach it is nearly impossible to gain grouping effects if the variables of one group are not all in or out of the update set $\mathcal{A}_{(l)}$ together. Unfortunately, we seldom a priori know which covariables constitute a (significant) group. So this must be done adaptively as in the GBlockBoost algorithm (Ulbricht and Tutz, 2008) what in turn can become very time-consuming when there is a large number of covariates.

In the third (and final) approach we will additionally include the previously estimated coefficients $\hat{\mathbf{b}}_{(l-1)}$ in the penalty term. Therefore let us consider the parameter vector

$$\mathbf{b}_{(l)}^* = \mathbf{b}_{(l-1)} + \boldsymbol{\gamma}_{(l)}, \tag{17}$$

where $\mathbf{b}_{(l-1)}$ is treated as fixed constant and $\boldsymbol{\gamma}_{(l)}$ is the unknown update part. The resulting penalty term is

$$\frac{1}{2}\mathbf{b}_{(l)}^{*\top}\mathbf{M}_\lambda^*\mathbf{b}_{(l)}^* = \frac{1}{2}\mathbf{b}_{(l-1)}^\top\mathbf{M}_\lambda^*\mathbf{b}_{(l-1)} + \boldsymbol{\gamma}_{(l)}^\top\mathbf{M}_\lambda^*\mathbf{b}_{(l-1)} + \frac{1}{2}\boldsymbol{\gamma}_{(l)}^\top\mathbf{M}_\lambda^*\boldsymbol{\gamma}_{(l)}. \tag{18}$$

Note that this penalty directly incorporates the magnitude of $\mathbf{b}_{(l-1)}$. For example, if one uses the correlation-based penalty then two highly correlated regressors x_i and x_j with $i \in \mathcal{A}_{(l-1)}$, $j \in \mathcal{A}_{(l)}$, $i \neq j$ show comparable absolute values of their corresponding coefficient estimates in $\mathbf{b}_{(l)}^*$, even if $\mathcal{A}_{(l-1)}$ and $\mathcal{A}_{(l)}$ are disjoint.

As seen from (17), $\mathbf{b}_{(l)}^*$ is composed of two components. Consequently, we need *two* different initial values for the one-step Fisher scoring update. Using $\mathbf{b}_{(l-1)} = \hat{\mathbf{b}}_{(l-1)}$ and

$\boldsymbol{\gamma}_{(l)} = 0$ as initial values we obtain

$$\mathbf{s}_p(\mathbf{b}^*_{(l)}) = \mathbf{X}^\top \mathbf{D}(\hat{\boldsymbol{\eta}}^{(l-1)})\boldsymbol{\Sigma}^{-1}(\hat{\boldsymbol{\eta}}^{(l-1)})(\mathbf{y} - \hat{\boldsymbol{\mu}}^{(l-1)}) - \mathbf{M}^*_\lambda \hat{\mathbf{b}}_{(l-1)}$$

and

$$\mathbf{F}_p(\mathbf{b}^*_{(l)}) = \mathbf{X}^\top \mathbf{W}(\hat{\boldsymbol{\eta}}^{(l-1)})\mathbf{X} + \mathbf{M}^*_\lambda$$

for penalized score vector and Fisher information, respectively. The resulting estimate of the update vector is

$$\hat{\boldsymbol{\gamma}}_{(l)} = \mathbf{I}_{\mathcal{A}_{(l)}} \mathbf{F}_p^{-1}(\mathbf{b}^*_{(l)})\mathbf{s}_p(\mathbf{b}^*_{(l)}). \tag{19}$$

Here the previous estimate $\hat{\mathbf{b}}_{(l-1)}$ is included in the penalized score vector and hence modifies the direction of steepest descent in contrast to the update (16).

Through some simple manipulations the update vector $\hat{\boldsymbol{\gamma}}_{(l)}$ can be written in a more tractable form.

Lemma 1. *Equation* (19) *can be written*

$$\hat{\boldsymbol{\gamma}}_{(l)} = \mathbf{I}_{\mathcal{A}_{(l)}} \left[(\mathbf{X}^\top \mathbf{W}_{l-1}\mathbf{X} + \mathbf{M}^*_\lambda)^{-1}\mathbf{X}^\top \mathbf{W}_{l-1} \left\{ \mathbf{D}^{-1}_{l-1}(\mathbf{y} - \hat{\boldsymbol{\mu}}^{(l-1)}) + \hat{\boldsymbol{\eta}}^{(l-1)} \right\} - \hat{\mathbf{b}}_{(l-1)} \right]. \tag{20}$$

The proof is given in the Appendix. Note that (20) allows for an interesting interpretation of the update. Indeed, $\hat{\boldsymbol{\gamma}}_{(l)}$ swaps the coefficients that correspond with the elements of $\mathcal{A}_{(l)}$ while all others are maintained. This makes it differ from the usual boosting algorithms such as componentwise boosting, RidgeBoost or GBlockBoost. There the coefficients are successively modified by adding current updates. Here we have a 'swap or maintain' strategy.

The resulting boosting algorithm is denoted ForwardBoost and is summarized in the following.

Algorithm Forward Boosting (ForwardBoost)

Step 1: Initialization Fit the model $\mu = h(\beta_0)$ by one step of Fisher scoring to obtain $\hat{\mathbf{b}}_{(0)} = (\hat{\beta}_0, 0, \ldots, 0)^\top$ with $\hat{\beta}_0 = g\left(\frac{1}{n}\sum_{i=1}^n y_i\right)$, and $\hat{\boldsymbol{\eta}}^{(0)} = \mathbf{X}\hat{\mathbf{b}}_{(0)}, \hat{\boldsymbol{\mu}}^{(0)} = \left\{ h(\mathbf{x}_1^\top \hat{\mathbf{b}}_{(0)}), \ldots, h(\mathbf{x}_n^\top \hat{\mathbf{b}}_{(0)}) \right\}^\top$, $\hat{\mathcal{A}}_{(0)} = \{0\}$.

Step 2: Iteration For $l = 1, 2, \ldots, l_{\max}$

- **(i) Estimation** Consider the potential update set $\mathcal{A}_{(l)} = \mathcal{A}_{(l-1)} \cup \{j\}, j \in \{0, 1, \ldots, p\}$. When fitting the model $\mu = h\left(\hat{\eta}^{(l-1)} + \mathbf{x}^\top \boldsymbol{\gamma}_{(l)}\right)$ use

$$\hat{\boldsymbol{\gamma}}_{(l)} = \mathbf{I}_{\mathcal{A}_{(l)}} \left[(\mathbf{X}^\top \mathbf{W}_{l-1}\mathbf{X} + \mathbf{M}^*_\lambda)^{-1}\mathbf{X}^\top \mathbf{W}_{l-1} \left\{ \mathbf{D}^{-1}_{l-1}(\mathbf{y} - \hat{\boldsymbol{\mu}}^{(l-1)}) + \hat{\boldsymbol{\eta}}^{(l-1)} \right\} - \hat{\mathbf{b}}_{(l-1)} \right].$$

- **(ii) Selection** Choose the potential update set that improves the fit maximally. That is

$$\hat{\mathcal{A}}_{(l)} = \arg\min_{\mathcal{A}_{(l)}} \mathrm{Dev}(\hat{\boldsymbol{\gamma}}_{(l)}).$$

(iii) Update For $\nu \in (0,1]$ update

$$\begin{aligned}\hat{\mathbf{b}}_{(l)} &= \hat{\mathbf{b}}_{(l-1)} + \nu\hat{\boldsymbol{\gamma}}_{(l)} \\ &= \hat{\mathbf{b}}_{(l-1)} + \nu\mathbf{I}_{\hat{\mathcal{A}}_{(l)}}\left[(\mathbf{X}^\top\mathbf{W}_{l-1}\mathbf{X} + \mathbf{M}_\lambda^*)^{-1}\mathbf{X}^\top\mathbf{W}_{l-1}\times\right. \\ &\quad \left.\times\left\{\mathbf{D}_{l-1}^{-1}(\mathbf{y} - \hat{\boldsymbol{\mu}}^{(l-1)}) + \hat{\boldsymbol{\eta}}^{(l-1)}\right\} - \hat{\mathbf{b}}_{(l-1)}\right],\end{aligned}$$

where $\hat{\boldsymbol{\eta}}^{(l)} = \mathbf{X}\hat{\mathbf{b}}_{(l)}$ and $\hat{\boldsymbol{\mu}}^{(l)} = \left\{h(\mathbf{x}_1^\top\hat{\mathbf{b}}_{(l)}), \ldots, h(\mathbf{x}_n^\top\hat{\mathbf{b}}_{(l)})\right\}^\top$.

Note that the active set $\mathcal{A}_{(l)}$ is monotonically increasing. As pointed out the update $\hat{\boldsymbol{\gamma}}_{(l)}$ completely interchanges the corresponding coefficients. Hence, both aspects are similar to the forward selection strategy to obtain suitable subsets of explanatory variables in regression models. For this reason we denote our method as 'Forward Boosting'.

As mentioned above, the ForwardBoost algorithm can be interpreted as gradient descent algorithm. To prevent from too big movements into a specific direction we use the additional boosting parameter $\nu \in (0,1]$. As a result, the algorithm tends to be more stable and the occurrence of divergence problems has been reduced. This parameter is not treated directly as a tuning parameter, that is it is not optimized on in a data driven way. In our experience, using $\nu = 0.1$ is quite a good compromise between increased stability and an increased number of necessary boosting iterations until convergence.

For $\nu = 1$ the coefficients corresponding to the indices in $\hat{\mathcal{A}}_{(l)}$ are currently updated by getting replaced by the corresponding elements of $\hat{\boldsymbol{\gamma}}_{(l)}$ while all other coefficients remain. For $\nu \in (0,1)$ the estimated coefficients $\hat{\mathbf{b}}_{(l)}$ are indeed a convex combination of $\hat{\mathbf{b}}_{(l-1)}$ and $\hat{\boldsymbol{\gamma}}_{(l)}$. This leads to a simple iterative version of the hat matrix.

As we have seen, the initial estimate of $\mathbf{b}$ is $\hat{\mathbf{b}}_{(0)} = (\hat{\beta}_0, 0, \ldots, 0)^\top$ with $\hat{\beta}_0 = g\left(1/n\sum_{i=1}^n y_i\right)$ yielding an initial hat matrix

$$\mathbf{H}_{\hat{\mathbf{b}}_{(0)}} = \mathbf{H}_{(0)} = \frac{1}{n}1_n 1_n^\top,$$

where 1_n denotes the n-dimensional vector of ones. In the l-th iteration $(l \geq 1)$ of the ForwardBoost algorithm the coefficients update is

$$\hat{\mathbf{b}}_{(l)} = \hat{\mathbf{b}}_{(l-1)} + \nu\hat{\boldsymbol{\gamma}}_{(l)}.$$

Let

$$\mathbf{H}_{(l)} = \mathbf{W}_{l-1}^{\top/2}\mathbf{X}\mathbf{I}_{\hat{\mathcal{A}}_{(l)}}(\mathbf{X}^\top\mathbf{W}_{l-1}\mathbf{X} + \mathbf{M}_\lambda^*)^{-1}\mathbf{X}^\top\mathbf{W}_{l-1}^{1/2}.$$

Due to its form the term $\mathbf{H}_{(l)}$ is easily seen to be the (generalized) hat matrix of the first summand on the right hand side of (20). Hence the hat matrix of $\hat{\mathbf{b}}_{(l)}$ is given iteratively by

$$\mathbf{H}_{\hat{\mathbf{b}}_{(l)}} = (1-\nu)\mathbf{H}_{\hat{\mathbf{b}}_{(l-1)}} + \nu\mathbf{H}_{(l)}. \tag{21}$$

We will use the trace of $\mathbf{H}_{\hat{\mathbf{b}}_{(l)}}$ as an estimate of degrees of freedom for estimating $\mathbf{b}$ in the l-th boosting iteration. Furthermore we apply the deviance to measure the performance of potential updates. However, its major problem is that it does not directly penalize for an increasing amount of model complexity when an additional regressor is included. This suggests to rather use an information criterion. Note that we would need to compute the trace of the updated hat matrix in this case. This in turn can become quite time-consuming. When dealing with metric covariates only we are usually on the safe site if we just add one regressor in a single boosting iteration. So the increasing complexity is just one additional parameter and hence can be neglected especially in the longer run when lot of coefficients are already included in the predictor.

Nevertheless, we use information criteria for stopping the algorithm, such as AIC

$$\mathrm{AIC}(\hat{\mathbf{b}}_{(l)}) = \mathrm{Dev}(\hat{\mathbf{b}}_{(l)}) + 2\,\mathrm{tr}(\mathbf{H}_{\hat{\mathbf{b}}_{(l)}}), \tag{22}$$

or BIC

$$\mathrm{BIC}(\hat{\mathbf{b}}_{(l)}) = \mathrm{Dev}(\hat{\mathbf{b}}_{(l)}) + \log(n)\,\mathrm{tr}(\mathbf{H}_{\hat{\mathbf{b}}_{(l)}}), \tag{23}$$

where $\mathrm{Dev}(\hat{\mathbf{b}}_{(l)})$ denotes the deviance of the fitted model in the l-th boosting step. Hence we stop ForwardBoost after the iteration that minimizes the stopping criterion. The trace of the corresponding hat matrix (21) can be used to estimate the unknown dispersion parameter according to (14).

After each iteration we check whether or not there is a definite update in the estimated coefficients. This is measured by

$$\frac{\|\hat{\mathbf{b}}_{(l)} - \hat{\mathbf{b}}_{(l-1)}\|}{\|\hat{\mathbf{b}}_{(l)}\|} \leq \epsilon \tag{24}$$

for some small $\epsilon > 0$. If not we stop the algorithm. It can happen then that the minimum position of the stopping criterion coincides with this iteration.

The estimated update set $\hat{\mathcal{A}}_{(l)}$ must be monotonically increasing in l, that is all the previously updated coefficients must be included in the current active set. This is especially necessary to obtain the grouping effect. For an explanation of the necessity let us initially consider the monotonically increasing active set $\hat{\mathcal{A}}_{(l)}$. Note that for $l \to \infty$ we have $\hat{\mathcal{A}}_{(l)} \to \{0, 1, \ldots, p\}$. Even if there are some pure noise regressors in the data the deviance will be improving (at least not getting worse) if some of them are additionally included in the predictor. As a result, in the long run the ForwardBoost algorithm will then tend to include all given regressors in the predictor. However, this will also occur for componentwise updates but the important point is that in the monotonic case for l 'large' the set $\hat{\mathcal{A}}_{(l)}$ will contain *all* coefficient indices.

Now we look at the resulting estimate of the ForwardBoost algorithm at convergence. We assume that the corresponding penalized Fisher scoring algorithm with identical $\mathbf{M}^*_\lambda$ converges to, say, $\hat{\mathbf{b}}_{QP}$. The condition of the ForwardBoost algorithm for convergence is that $\hat{\mathbf{b}}_{(l+1)} = \hat{\mathbf{b}}_{(l)}$ (up to small componentwise relative errors). Due to the update step 2(iii) this condition is equivalent with $\hat{\boldsymbol{\gamma}}_{(l)} = \mathbf{0}$ or $\hat{\mathbf{b}}_{(l-1)} = (\mathbf{X}^\top\mathbf{W}_{l-1}\mathbf{X} +$

$\mathbf{M}^*_\lambda)^{-1}\mathbf{X}^\top\mathbf{W}_{l-1}\left\{\mathbf{D}^{-1}_{l-1}(\mathbf{y}-\hat{\boldsymbol{\mu}}^{(l-1)})+\hat{\boldsymbol{\eta}}^{(l-1)}\right\}$ what in turn is algebraically equivalent with $\hat{\mathbf{b}}_{QP}$. Note that rather the index l but $\mathbf{W}, \mathbf{D}$ and $\boldsymbol{\mu}$ must be identical in $\hat{\mathbf{b}}_{(l-1)}$ and $\hat{\mathbf{b}}_{QP}$ in order to fulfill the convergence assumption. For this reason both estimates will coincide. Now consider the set $\hat{\mathcal{A}}_{(l)}$ at convergence again. For monotonic increases this implies that all elements of $\hat{\mathbf{b}}_{(l-1)}$ and $\hat{\mathbf{b}}_{QP}$ must coincide. If at convergence $|\hat{\mathcal{A}}_{(l)}| < p+1$ (e.g. $|\hat{\mathcal{A}}_{(l)}| = 1$ as for componentwise updates) then it is sufficient for $\hat{\mathbf{b}}_{(l-1)}$ and $\hat{\mathbf{b}}_{QP}$ to just being equal in the corresponding elements of $\hat{\mathcal{A}}_{(l)}$. Consequently, they do not have to be equal in all of their elements. Indeed, the monotonic increase of $\hat{\mathcal{A}}_{(l)}$ guarantees the ForwardBoost algorithm to converge to $\hat{\mathbf{b}}_{QP}$ if it exists. Furthermore, in our experience using a monotonic increase also fastens the speed of convergence.

We illustrate the functioning of the ForwardBoost algorithm with the following small example. Consider a GLM with Poisson distributed response. The predictor contains an intercept and three highly correlated regressors x_1, x_2, x_3. The true parameter vector is $\mathbf{b}^0 = (0.5, 1, 2, 0)^\top$, the correlation structure is $\varrho_{ij} = 0.99^{|i-j|}$ for $i, j = 1, 2, 3$. Due to this correlation structure the estimated coefficients $\hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_3$ are supposed to be nearly equal if the grouping effect will be gained by the algorithm. We use a sample of size $n = 20$. The main results from the first seven iterations are given in Table 1. Note that the algorithm converges very fast. After seven iterations the change of the estimated coefficients meets (24) where we have used $\epsilon = 10^{-8}$. The resulting coefficient build-up is visualized in the right panel of Figure 2. In its left panel we visualize the failure of GBlockBoost in this situation. As you can see the updates of x_2 on one hand and x_1 and x_3 on the other behave quite parallel after the second iteration. But the algorithm is not able to adjust the levels for the coefficients of these both groups.

Another application of the ForwardBoost algorithm is shown in Figure 3. Here we use a probit model based on $n = 75$ independent observations. The predictor consists of an intercept and 11 regressors. The first nine regressors form three clusters of three highly correlated regressors, e.g. x_1, x_2, x_3 are cluster 1, x_4, x_5, x_6 are cluster 2, and x_7, x_8, x_9 are cluster 3. The clusters are not correlated among themselves. We generate the correlation ϱ_{ij} among the regressors x_i and x_j as $\varrho_{ij} = \varrho^{|i-j|}$ if i, j belong to the same cluster, and $\varrho_{ij} = 1_{\{i=j\}}$ otherwise, that is $\varrho_{ij} = 1$ if $i = j$ and $\varrho_{ij} = 0$ otherwise. We set $\varrho = 0.95$. The true parameter vector is $\mathbf{b}^0 = (0, 1, 2, 0, 1, -2, 0, 0, 0, 0, 1, 0)^\top$. Hence, the first two clusters are relevant but only the first two components of each cluster are indeed relevant to predict the response. Furthermore, x_{10} is also relevant. We have used Forward Boosting with the correlation-based penalty (9) in the left panel and with the correlation-driven penalty function (10) of the weighted fusion penalty in the right panel. Note that the latter is more adequate in extracting the grouping effect. But we have to find two, instead of one, optimal tuning parameters in this case.

Iteration:	$l=1$	$l=2$	$l=3$	$l=4$	$l=5$	$l=6$	$l=7$
	-0.00	-0.16	-0.14	-0.05	-0.00	-0.00	-0.00
$\hat{\boldsymbol{\gamma}}^0_{(l)}$	0.00	0.00	0.00	0.03	0.00	0.00	0.00
	0.00	0.11	0.12	0.03	0.00	0.00	0.00
	0.00	0.00	0.12	0.03	0.00	0.00	0.00
$\mathrm{Dev}(\hat{\gamma}^0_{(l)})$	185.81	137.38	93.70	62.47	62.43	62.42	62.42
	-0.00	-0.16	**-0.14**	-0.05	-0.00	-0.00	-0.00
$\hat{\boldsymbol{\gamma}}^1_{(l)}$	1.14	1.25	**1.37**	0.03	0.00	0.00	0.00
	0.00	0.11	**0.12**	0.03	0.00	0.00	0.00
	0.00	0.00	**0.12**	0.03	0.00	0.00	0.00
$\mathrm{Dev}(\hat{\gamma}^1_{(l)})$	142.10	99.58	**64.41**	62.47	62.43	62.42	62.42
	-0.00	-0.16	-0.14	-0.05	-0.00	-0.00	-0.00
$\hat{\boldsymbol{\gamma}}^2_{(l)}$	**0.00**	0.00	0.00	0.03	0.00	0.00	0.00
	1.14	0.11	0.12	0.03	0.00	0.00	0.00
	0.00	0.00	0.12	0.03	0.00	0.00	0.00
$\mathrm{Dev}(\hat{\gamma}^2_{(l)})$	**140.09**	137.38	93.70	62.47	62.43	62.42	62.42
	-0.00	**-0.16**	-0.14	**-0.05**	**-0.00**	**-0.00**	**-0.00**
$\hat{\boldsymbol{\gamma}}^3_{(l)}$	0.00	**0.00**	0.00	**0.03**	**0.00**	**0.00**	**0.00**
	0.00	**0.11**	0.12	**0.03**	**0.00**	**0.00**	**0.00**
	1.14	**1.25**	0.12	**0.03**	**0.00**	**0.00**	**0.00**
$\mathrm{Dev}(\hat{\gamma}^3_{(l)})$	140.49	**98.31**	93.70	**62.47**	**62.43**	**62.42**	**62.42**

Table 1: The main results for the first 7 iterations of the ForwardBoost algorithm with $\lambda = 1.5, \nu = 1$ and $P^{cb}_{\lambda}(\boldsymbol{\beta})$ as penalty when applied to the Poisson example. We denote the (potential) updates at the l-th iteration with $\hat{\boldsymbol{\gamma}}^r_{(l)}$ where r indicates the added regressor, besides the intercept. Consider the first column where $l = 1$. The intercept has been already estimated previously. So if β_0 is solely to be updated then there is only a small negative change. As you can see, the grouping effect directly works since all coefficients $\hat{\gamma}^1_{1,(1)} = \hat{\gamma}^2_{2,(1)} = \hat{\gamma}^3_{3,(1)} = 1.14$ get (nearly) identical values. Due to a minimum deviance $\hat{\boldsymbol{\gamma}}^2_{(1)}$ has been chosen for updating, so that $\hat{\beta}_{2,(1)} = 1.14$. Now consider the second column where $l = 2$. Since ForwardBoost uses monotonic updates, $\gamma_{2,(2)}$ will always be updated. As you can see, the grouping effect is also kept here. For the other regressors we get $\hat{\gamma}_{1,(2)} = \hat{\gamma}_{3,(2)} = 1.25$ which correspond with an update of $\hat{\beta}_{2,(1)}$ about 0.11. Finally, $\hat{\boldsymbol{\gamma}}^3_{(2)}$ has been chosen, so that $\hat{\beta}_{2,(2)} = \hat{\beta}_{3,(2)} = 1.25$. This principle continues in the next iterations.

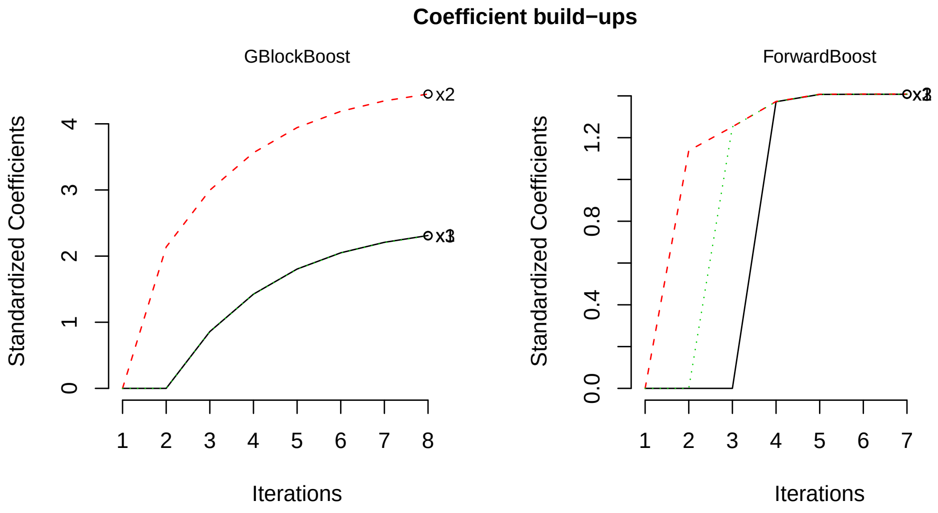


Figure 2: Coefficient build-up of the GBlockBoost algorithm (Ulbricht and Tutz, 2008) with $\lambda = 1.5$ (left panel) and of the ForwardBoost algorithm with $\lambda = 1.5, \nu = 1$ and $P_\lambda^{cb}(\boldsymbol{\beta})$ as penalty (right panel). Note that the grouping effect of the ForwardBoost algorithm can be obtained at the earliest after p iterations (here $p = 3$). This is due to its construction in the sense of forward selection.

5 Application to Simulations and Real Data

5.1 Simulations

We consider two different classes of simulation settings, a low-dimensional (that is $n \gg p$) and a high-dimensional one ($n \ll p$). In both classes the predictor consists of a number of blocked regressors and a number of independent regressors.

In the low-dimensional settings we use $p = 40$ variables, $n_{\text{train}} = 100$ training data to fit the model, $n_{\text{vali}} = 50$ validation data to select optimal tuning parameters and $n_{\text{test}} = 50$ test data to evaluate the model performances. The true parameters are given by

$$\boldsymbol{\beta}^0 = (\underbrace{0,\dots,0}_{\text{10 times}}, \underbrace{0.5,\dots,0.5}_{\text{10 times}}, \underbrace{0,\dots,0}_{\text{10 times}}, \underbrace{0.25,\dots,0.25}_{\text{10 times}})$$

and the true intercept as $\beta_0^0 = 0$ yielding $\mathbf{b}^0 = (\beta_0^0, \boldsymbol{\beta}^0)^\top$. The corresponding covariates are simulated as multivariate normal random variables with unit variances and the following correlation structure: The first 20 covariates constitute two independent blocks of 10 variables each with correlation structure between x_i and x_j

$$\varrho_{ij} = \begin{cases} \varrho^{|i-j|}, & \text{if } i, j \text{ are in the same block,} \\ 1_{\{i=j\}}, & \text{otherwise,} \end{cases} \tag{25}$$

where $1_{\{i=j\}} = 1$ if $i = j$ and $1_{\{i=j\}} = 0$ otherwise. We vary the correlation parameter as $\varrho \in \{0.99, 0.95, 0.5\}$ to investigate performances under different amounts of grouping

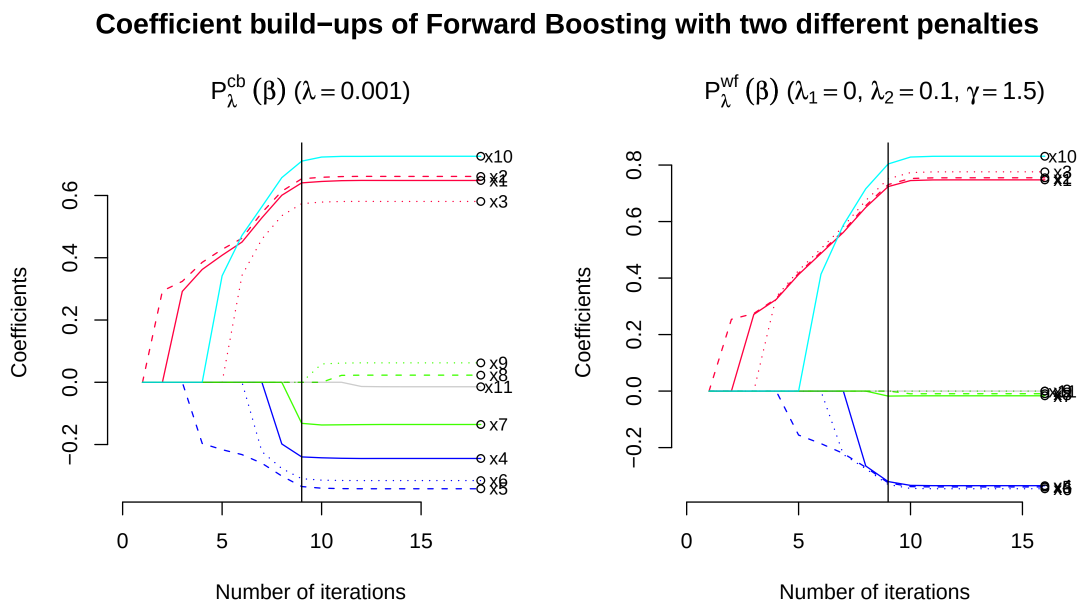


Figure 3: Application of the ForwardBoost algorithm to the simulated probit model. In the left panel we use the correlation-based penalty (9) with $\lambda = 0.001$. In the right panel we apply the correlation-driven penalty function (10) of the weighted fusion penalty (Daye and Jeng, 2009) with $\lambda_2 = 0.1$ and $\gamma = 1.5$. Note that $\lambda_1 = 0$ indicates the irrelevance of the lasso penalty term, so that variable selection is solely driven by the ForwardBoost algorithm. When the AIC criterion (22) is used to stop then $l = 9$ iterations are optimal for both penalties.

strength. Note that $0.99^{10} \approx 0.9, 0.95^{10} \approx 0.6$, and $0.5^{10} \approx 0.001$ so that we capture all three situations of high correlation, mean correlation and nearly no correlation throughout the blocks, respectively. The last 20 covariates are drawn independently from the standard normal distribution. Note that just the second block and the last 10 independent variables are influential.

In the high-dimensional settings we consider $p = 100$ regressors and $n_{\text{train}} = 40$ training data for model fitting, $n_{\text{vali}} = 20$ validation data to determine optimal tuning parameters, and $n_{\text{test}} = 20$ test data for model evaluation. The true parameters are now given by

$$\boldsymbol{\beta}^0 = (\underbrace{0,\ldots,0}_{\text{20 times}}, \underbrace{0.5,\ldots,0.5}_{\text{10 times}}, \underbrace{1,\ldots,1}_{\text{10 times}}, \underbrace{0.5,\ldots,0.5}_{\text{10 times}}, \underbrace{0.25,\ldots,0.25}_{\text{10 times}}, \underbrace{0,\ldots,0}_{\text{40 times}})$$

and again $\beta_0^0 = 0$ so that $\mathbf{b}^0 = (\beta_0^0, \boldsymbol{\beta}^0)^\top$ as before. Also the correlation structure of the covariates is similar to the low-dimensional setting. Now the first 50 covariates are blocked into 5 blocks of 10 correlated variables each. The correlation structure is again given by (25). The last 50 covariates are independently drawn from the standard normal distribution. Now just the last three blocks are truly influential as well as the first 10 independent regressors.

For the simulation studies we create 100 replications which are based on the following model assumptions

(i) Logit model with binary dependent observations, i.e. $y_i \sim B(1, p_i)$, where $p_i = (1 + \exp\{-\eta_i\})^{-1}$;

(ii) a loglinear Poisson model, i.e. $y_i \sim Pois(\lambda_i)$ with $\lambda_i = \exp\{-\eta_i/4\}$;

In the high-dimensional settings we modify the parameter of model (ii) to $\lambda_i = \exp\{-\eta_i/24\}$.

We use the deviance based on the test data as a measure for goodness of fit. For ease of description we will refer to it as *deviance loss*. To consider the explanation ability of our fitted models, we use

$$MSE_b(\hat{\mathbf{b}}, \mathbf{b}^0) := \|\hat{\mathbf{b}} - \mathbf{b}^0\|_2^2 \tag{26}$$

that measures the squared deviation between the estimated coefficients $\hat{\mathbf{b}}$ and the true parameter vector $\mathbf{b}^0$. As the test deviance and MSE_b are based on loss functions, the smaller their values the better the performance.

Additionally, we use the criteria *hits* and *false positives* to evaluate the identification of relevant regressors. For $j = 0, 1, \ldots, p$ they are defined as

$$hits(\hat{\mathbf{b}}, \mathbf{b}^0) := \left| \left\{ \{j : \hat{\beta}_j \neq 0\} \cap \{j : \beta_j^0 \neq 0\} \right\} \right|, \tag{27}$$

and

$$fps(\hat{\mathbf{b}}, \mathbf{b}^0) := \left| \left\{ \{j : \hat{\beta}_j \neq 0\} \cap \{j : \beta_j^0 = 0\} \right\} \right|, \tag{28}$$

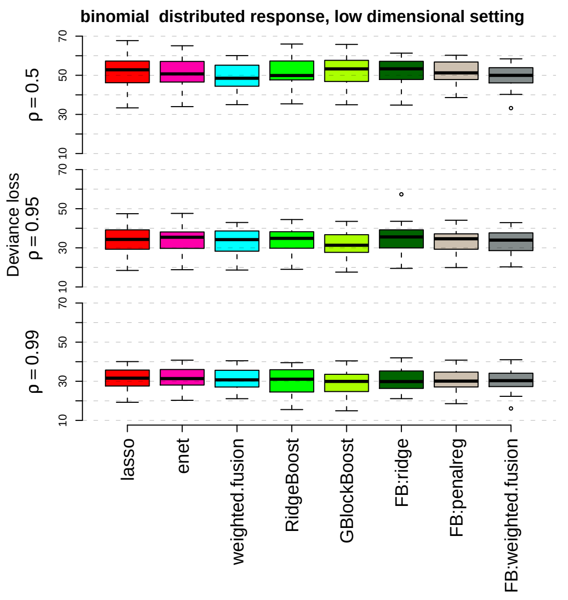
respectively. Hence, *hits* refers to the number of correctly identified coefficients, false positives (fps) is the number of non-influential regressors (including the intercept if appropriate) dubbed influential.

We consider the performance of Forward Boosting based on the ridge penalty (FB:ridge), the correlation-based penalty (FB:penalreg), and the weighted-fusion penalty (FB:weighted.fusion), respectively. We compare their results with the performances of lasso, elastic net, RidgeBoost, and GBlockBoost (Ulbricht and Tutz, 2008). For Forward Boosting, the weighted-fusion penalty is reduced to the correlation-driven penalty (10). As proposed by Daye and Jeng (2009) we use

$$\omega_{ij} = \frac{|\varrho_{ij}|^\gamma}{1 - |\varrho_{ij}|}, \tag{29}$$

where $\gamma > 0$ is an additional tuning parameter to compute (10). Note that all considered methods are invariant towards the order of regressors.

The result for the test deviances and the MSE_b are given in Figures 4 to 8. All considered methods show similar results for the test deviances in all low and high dimensional simulation settings. The level of deviance increases when the correlation among regressors decreases. However, Forward Boosting shows the best performance in terms of MSE_b. The methods that do not explicitly incorporate the grouping effect, for example lasso,



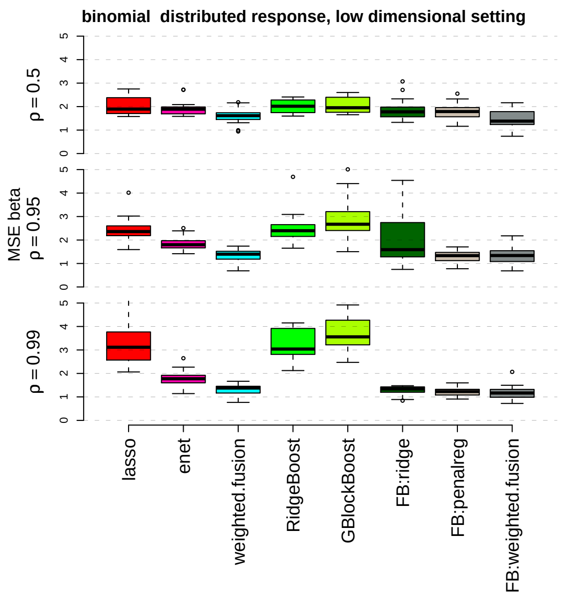


Figure 4: Boxplots of the low dimensional setting with binomial distributed response.

RidgeBoost and GBlockBoost, show poor performance in particular when correlation is high. For small correlation the difference between methods is smaller. As is seen from Figures 6 and 9, the ForwardBoost algorithm tends to find the most hits. In the low dimensional settings all methods, besides GBlockBoost, behave quite similar in the number of false positives. In the high dimensional settings this aspect slightly changes as the ForwardBoost tends to select some more false positives especially when $\varrho = 0.5$. However, this primarily refers to the inclusion of the grouping effect.

5.2 Predicting Fat Content From Spectrometric Wavelengths

Chemometrics deals with the data-driven extraction of information from chemical systems. One important field is signal regression where the outcomes are scalars and the regressors are one-dimensional signals that have been measured by some (near infrared) spectroscopy analysis. A nice overview on statistical tools for signal regression has been given by Frank and Friedman (1993). Applications in signal regression are usually characterized by high correlations between neighboring regressors and an $n \ll p$ data situation. Signal regression is often related to functional data analysis (Ramsay and Silverman, 2005). The latter interprets the regressors as smooth function.

If the main concern of data analysis is prediction, then smoothing of regressors might be appropriate. On the other hand, regressor selection is of interest from the viewpoint of interpretability. One wants to know which covariates effect upon the response. For spectroscopy data we are primarily interested in finding relevant areas of wavelengths (Tutz and Gertheiss, 2010). As common techniques of functional regression analysis are not capable of this we will focus on shrinkage and boosting methods in the following.

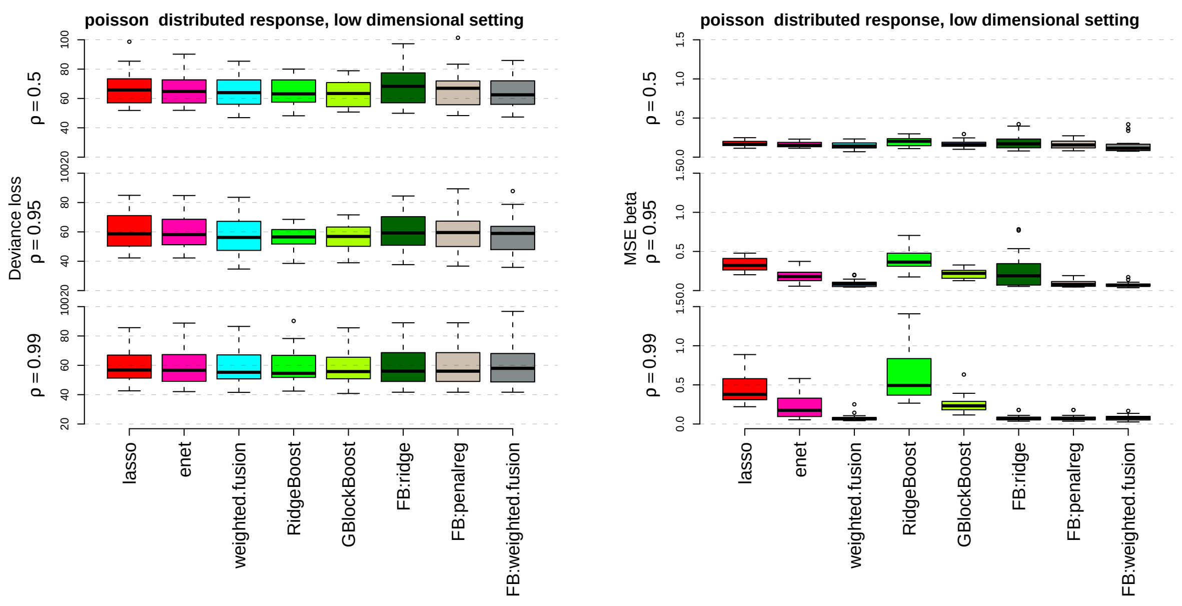


Figure 5: Boxplots of the low dimensional setting with Poisson distributed response.

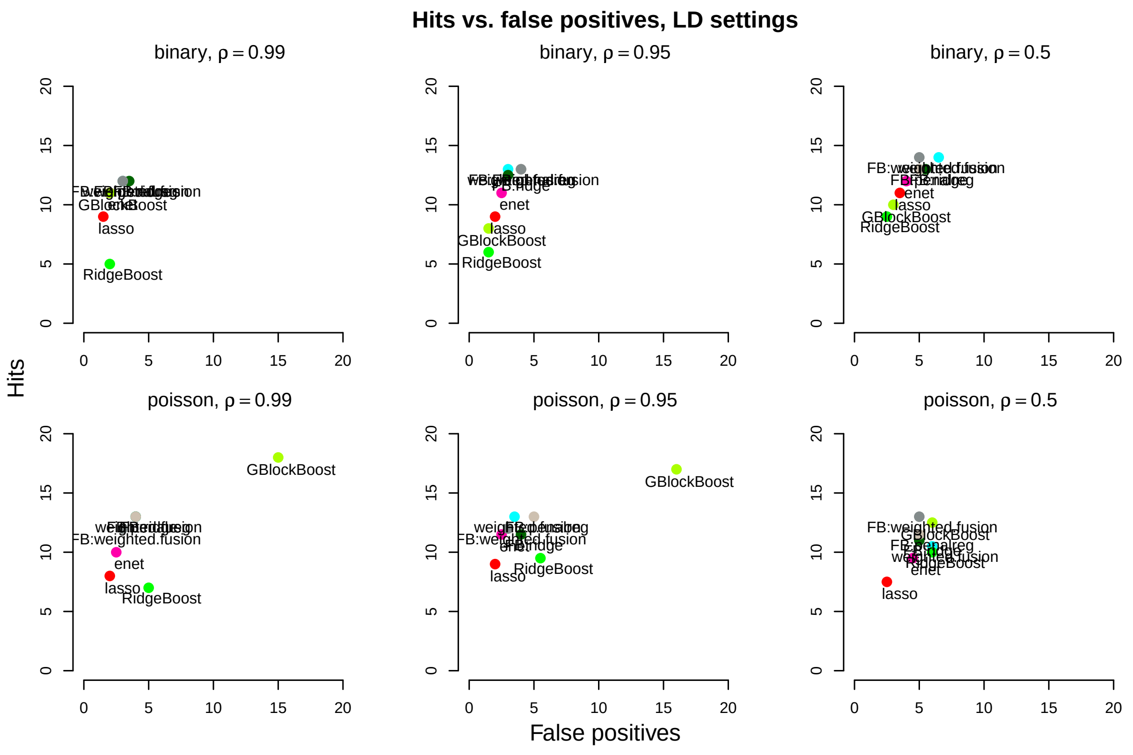


Figure 6: Hits versus false positives, low dimensional (LD) settings

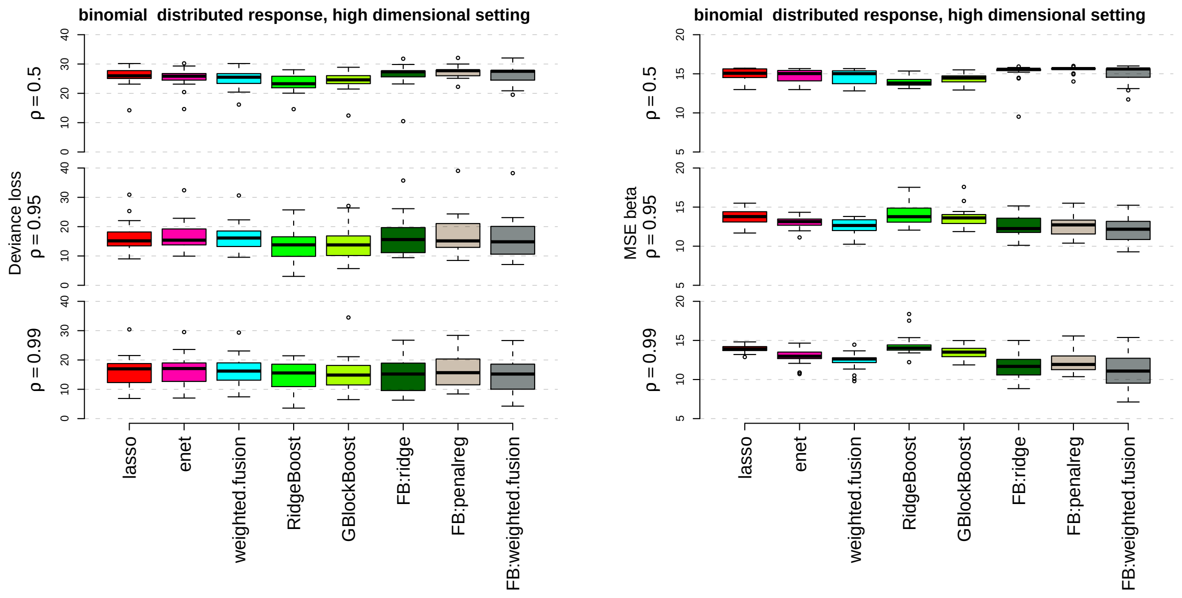


Figure 7: Boxplots of the high dimensional setting with binomial distributed response.

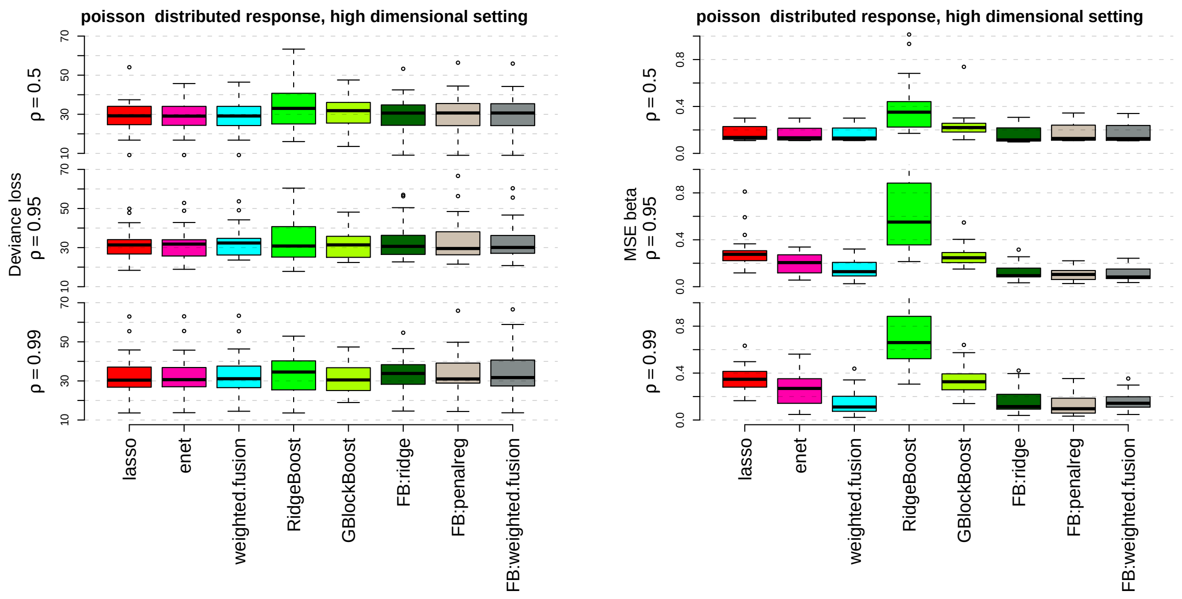


Figure 8: Boxplots of the high dimensional setting with Poisson distributed response.

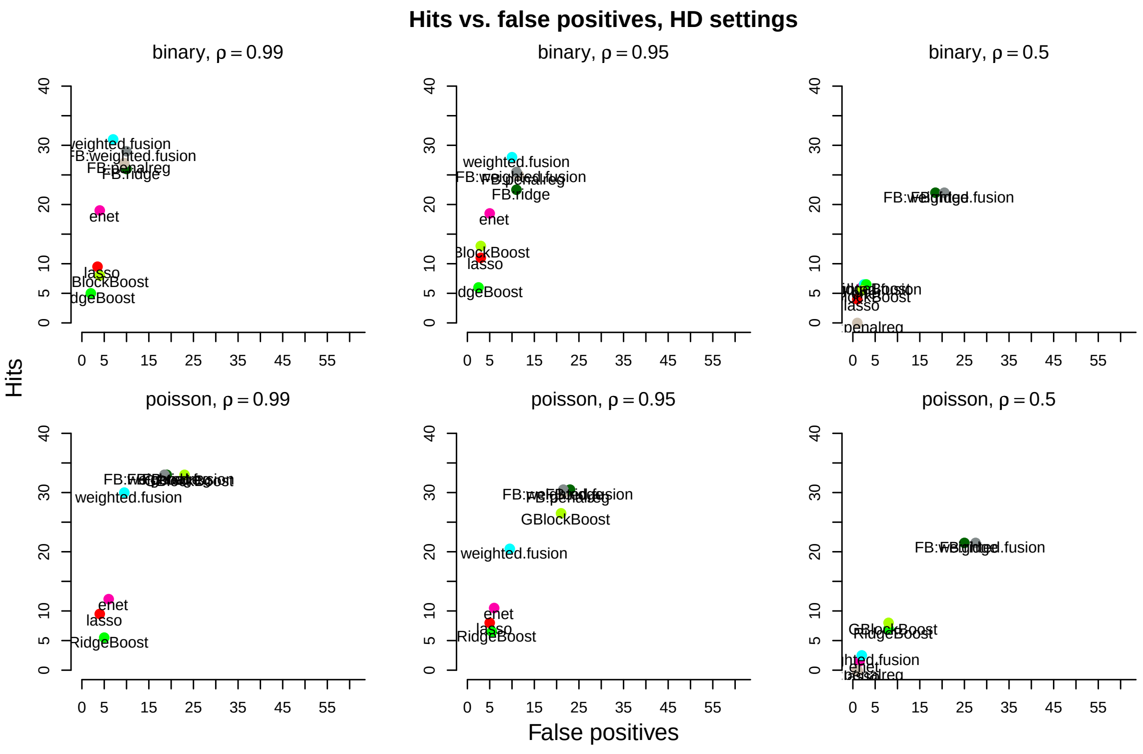


Figure 9: Hits versus false positives, high dimensional (HD) settings

The original data we want to analyze come from a quality control problem in food industry and has been used in Ferraty and Vieu (2006). This data set concerns a sample of $n = 215$ pieces of finely chopped meat. The response is the content of fat, the regressors consist of a channel spectrum of absorbances that has been discretized to $p = 100$ equidistant points. These data have been recorded on a Tecator Infratec Food and Feed Analyzer working in the wavelength range from 850 to 1050 nm by the Near Infrared Transmission (NIT) principle. Given a new set of spectrometric wavelengths the task is to predict the corresponding fat content. Indeed, it is less expensive (in terms of time and costs) to obtain the spectrometric wavelengths than to determine the percentage of fat. Hence, it is an important challenge to predict the fat content from the spectrometric data.

We randomly split the data set into $n_{\text{train}} = 129$ training data to fit the model, $n_{\text{vali}} = 43$ validation data to determine the tuning parameters with the help of the AIC criterion, and $n_{\text{test}} = 43$ test data to evaluate model performance. We repeat the random splitting 20 times. In order to apply a GLM we need to specify the exponential family of the response and a suitable link function in a first step. The model specification that has clearly ruled out the other competing ones was the inverse Gaussian distribution with log-link. As there is a natural order between the wavelengths we additionally consider the fused lasso penalty (Tibshirani et al., 2005) in the following.

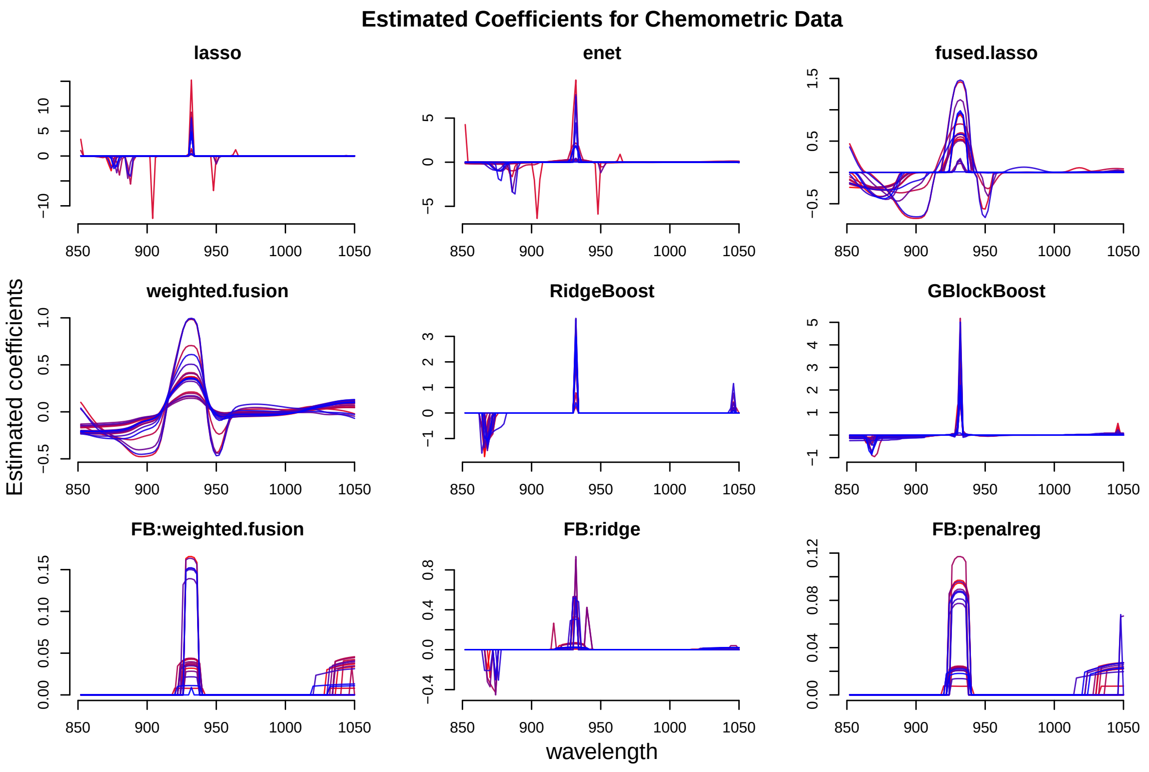


Figure 10: Estimated regressor coefficients for the chemometric data set. To help you identify the single replications we use a color spectrum from red (first replication) to blue (last replication).

Figure 10 shows the estimated regressor coefficients of all 20 replications. Forward Boosting shows the best results with regard to the combination of sparsity and smoothness. While lasso, elastic net, RidgeBoost and GBlockBoost tend to favor regression models which are too sparse, fused lasso and weighted fusion select models with high complexity. Furthermore the two latter methods show quite instable results through the 20 replications. Variable selection for Forward Boosting is very stable. The ridge penalty again tends to models which are too sparse not considering any groupings.

6 Conclusion

We proposed a new boosting technique that combines quadratic penalization and explicit variable selection in GLMs. As the monotonically increasing set of active regressors and the structure of the weak learner are quite similar to forward selection in classical regression models we denote our method as Forward Boosting. It has turned out to be highly competitive in both simulation studies and application to real data, especially when the focus is more on identifying the true model than on gaining perfect prediction. Hence, our new method is primarily intended when to study the association structure between regressors and their relations to the response.

The ForwardBoost algorithm is quite competitive even from computational complexity. The monotonicity of the active regressor set encourages an increase in speed of convergence. This still holds for $\nu < 1$. In our experience, the algorithm behaves very stable. We have seen that the choice of a concrete quadratic penalty does not matter much in performance. Consequently, quadratic penalties with a single tuning parameter, such as the correlation-based penalty, might be favored, especially when another focus is on the incorporation of grouping effects.

Appendix

Proof of Lemma 1:

To simplify notation we use $\mathbf{W}_{l-1} = \mathbf{W}(\hat{\boldsymbol{\eta}}^{(l-1)}), \boldsymbol{\Sigma}_{l-1} = \boldsymbol{\Sigma}(\hat{\boldsymbol{\eta}}^{(l-1)})$ and $\mathbf{D}_{l-1} = \mathbf{D}(\hat{\boldsymbol{\eta}}^{(l-1)})$. It holds that

$$\begin{aligned}(\mathbf{X}^\top\mathbf{W}_{l-1}\mathbf{X}+\mathbf{M}_\lambda^*)^{-1}\mathbf{X}^\top\mathbf{W}_{l-1}\mathbf{X}-\mathbf{I} &= (\mathbf{X}^\top\mathbf{W}_{l-1}\mathbf{X}+\mathbf{M}_\lambda^*)^{-1}\mathbf{X}^\top\mathbf{W}_{l-1}\mathbf{X}\\ &\quad -(\mathbf{X}^\top\mathbf{W}_{l-1}\mathbf{X}+\mathbf{M}_\lambda^*)^{-1}(\mathbf{X}^\top\mathbf{W}_{l-1}\mathbf{X}+\mathbf{M}_\lambda^*)\\ &= -(\mathbf{X}^\top\mathbf{W}_{l-1}\mathbf{X}+\mathbf{M}_\lambda^*)^{-1}\mathbf{M}_\lambda^*, \qquad (30)\end{aligned}$$

so we could write

$$\begin{aligned}\hat{\boldsymbol{\gamma}}_{(l)} &= \mathbf{I}_{\mathcal{A}_l}(\mathbf{X}^\top\mathbf{W}_{l-1}\mathbf{X}+\mathbf{M}_\lambda^*)^{-1}\left\{\mathbf{X}^\top\mathbf{D}_{l-1}\boldsymbol{\Sigma}_{l-1}^{-1}(\mathbf{y}-\hat{\boldsymbol{\mu}}^{(l-1)})-\mathbf{M}_\lambda^*\hat{\mathbf{b}}_{(l-1)}\right\}\\ &= \mathbf{I}_{\mathcal{A}_l}\Big\{(\mathbf{X}^\top\mathbf{W}_{l-1}\mathbf{X}+\mathbf{M}_\lambda^*)^{-1}\mathbf{X}^\top\mathbf{D}_{l-1}\boldsymbol{\Sigma}_{l-1}^{-1}(\mathbf{y}-\hat{\boldsymbol{\mu}}^{(l-1)})-\\ &\quad -(\mathbf{X}^\top\mathbf{W}_{l-1}\mathbf{X}+\mathbf{M}_\lambda^*)^{-1}\mathbf{M}_\lambda^*\hat{\mathbf{b}}_{(l-1)}\Big\}\\ &\overset{(30)}{=} \mathbf{I}_{\mathcal{A}_l}\Big\{(\mathbf{X}^\top\mathbf{W}_{l-1}\mathbf{X}+\mathbf{M}_\lambda^*)^{-1}\mathbf{X}^\top\mathbf{D}_{l-1}\boldsymbol{\Sigma}_{l-1}^{-1}(\mathbf{y}-\hat{\boldsymbol{\mu}}^{(l-1)})+\\ &\quad +(\mathbf{X}^\top\mathbf{W}_{l-1}\mathbf{X}+\mathbf{M}_\lambda^*)^{-1}\mathbf{X}^\top\mathbf{W}_{l-1}\mathbf{X}\hat{\mathbf{b}}_{(l-1)}-\hat{\mathbf{b}}_{(l-1)}\Big\}\\ &= \mathbf{I}_{\mathcal{A}_l}\Big[(\mathbf{X}^\top\mathbf{W}_{l-1}\mathbf{X}+\mathbf{M}_\lambda^*)^{-1}\Big\{\mathbf{X}^\top\mathbf{D}_{l-1}\boldsymbol{\Sigma}_{l-1}^{-1}(\mathbf{y}-\hat{\boldsymbol{\mu}}^{(l-1)})+\\ &\quad +\mathbf{X}^\top\mathbf{W}_{l-1}\mathbf{X}\hat{\mathbf{b}}_{(l-1)}\Big\}-\hat{\mathbf{b}}_{(l-1)}\Big].\end{aligned}$$

Since $\mathbf{D}_{l-1}\boldsymbol{\Sigma}_{l-1}^{-1} = \mathbf{W}_{l-1}\mathbf{D}_{l-1}^{-1}$ and $\mathbf{X}\hat{\mathbf{b}}_{(l-1)} = \hat{\boldsymbol{\eta}}^{(l-1)}$ the proposed result follows.

References

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

Bühlmann, P. and B. Yu (2006). Sparse boosting. *Journal of Machine Learning Research 7*, 1001–1024.

Daye, Z. J. and X. J. Jeng (2009). Shrinkage and model selection with correlated variables via weighted fusion. *Computational Statistics and Data Analysis 53*, 1284–1298.

Fahrmeir, L. and G. Tutz (2001). *Multivariate Statistical Modelling based on Generalized Linear Models* (2nd ed.). New York: Springer.

Ferraty, F. and P. Vieu (2006). *Nonparametric Functional Data Analysis: Theory and Practice*. New York: Springer.

Frank, I. E. and J. H. Friedman (1993). A statistical view of some chemometrics regression tools (with discussion). *Technometrics 35*, 109–148.

Friedman, J., T. Hastie, and R. Tibshirani (2008). Regularization paths for generalized linear models via coordinate descent. Technical report, Department of Statistics, Stanford University, Stanford.

Nelder, J. A. and R. W. M. Wedderburn (1972). Generalized linear models. *Journal of the Royal Statistical Society, Series A 135*, 370–384.

Ramsay, J. O. and B. W. Silverman (2005). *Functional Data Analysis* (2nd ed.). New York: Springer.

Slawski, M., W. zu Castell, and G. Tutz (2009). Feature selection guided by structural information. Technical Report 051, Department of Statistics, University of Munich.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B 58*, 267–288.

Tibshirani, R., M. Saunders, S. Rosset, J. Zhu, and K. Knight (2005). Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society B 67*, 91–108.

Tutz, G. and H. Binder (2007). Boosting ridge regression. *Computational Statistics & Data Analysis 51*, 6044–6059.

Tutz, G. and J. Gertheiss (2010). Feature extraction in signal regression: A boosting technique for functional data regression. *Journal of Computational and Graphical Statistics 19*, 154–174.

Tutz, G. and J. Ulbricht (2009). Penalized regression with correlation based penalty. *Statistics and Computing 19*, 239–253.

Ulbricht, J. and G. Tutz (2008). Boosting correlation based penalization in generalized linear models. In Shalabh and C. Heumann (Eds.), *Recent Advances in Linear Models and Related Areas*. Heidelberg: Springer.

Wood, S. N. (2006). *Generalized Additive Models: An Introduction with R*. Boca Raton: Chapman & Hall/CRC.

Yuan, M. and Y. Lin (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society B 68*, 49–67.

Zou, H. and T. Hastie (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society B 67*, 301–320.