Submitted to Statistical Science

BOOSTING: A STATISTICAL PERSPECTIVE

By Peter Bühlmann and Torsten Hothorn

ETH Zürich and Universität Erlangen-Nürnberg

We present a statistical perspective on boosting. Special emphasis is given to estimating potentially complex parametric or nonparametric models, including generalized linear and additive models as well as regression models for survival analysis. Concepts of degrees of freedom and corresponding Akaike or Bayesian information criteria, particularly useful for regularization and variable selection in high-dimensional covariate spaces, are discussed as well.

The practical aspects of boosting procedures for fitting statistical models are illustrated by means of the dedicated open-source software package **mboost**. This package implements functions which can be used for model fitting, prediction and variable selection. It is flexible, allowing for the implementation of new boosting algorithms optimizing user-specified loss functions, and it is especially attractive for variable selection in high-dimensional generalized linear models.

1. Introduction. Freund and Schapire's AdaBoost algorithm for classification [26–28] has attracted much attention in the machine learning community [cf. 61, and the references therein] as well as in related areas in statistics [14, 15, 30]. Various versions of the AdaBoost algorithm have proven to be very competitive in terms of prediction accuracy in a variety of applications. Boosting methods have been originally proposed as ensemble methods, see Section 1.1, which rely on the principle of generating multiple predictions and majority voting (averaging) among the individual classifiers.

Later, Breiman [14, 15] made a path-breaking observation that the AdaBoost algorithm can be viewed as a gradient descent algorithm in function space, inspired by numerical optimization and statistical estimation; see also Friedman et al. [30] and Rätsch et al. [54]. This insight opened new perspectives, namely to use boosting methods in many other contexts than classification. We mention here boosting methods for regression (including generalized regression) [19, 29, 55], for density estimation [56], for survival analysis [37, 55] or for multivariate analysis [30, 46]. In quite a few of these proposals, boosting is not only a black-box prediction tool but also an estimation method for models with a specific structure such as linearity or additivity [17, 19, 37]. Boosting can then be seen as an interesting regular-

Keywords and phrases: Generalized linear models, Generalized additive models, Gradient boosting, Survival analysis, Variable selection, Software

ization scheme for estimating a model. This is what we call the "statistical perspective" which will drive the focus of our exposition of boosting.

We present here some coherent explanations and illustrations of concepts about boosting, some derivations which are novel and we aim to increase understanding of the methods and selected known results. Besides giving an overview on theoretical concepts of boosting as an algorithm for fitting statistical models, we look at the methodology from a practical point of view as well. The associated add-on package **mboost** ["model boosting", 36] to the R system for statistical computing [53] implements computational tools which enable the data analyst to compute on the theoretical concepts explained in this paper as close as possible. The theoretical ingredients of boosting algorithms, such as loss functions and its negative gradients, base learners and internal stopping criteria, find their computational counterparts in the **mboost** package. Its implementation and user-interface reflect our "statistical perspective" of boosting as a tool for estimation in structured models. For example, and extending the reference implementation of tree-based gradient boosting from the **gbm** package [57] in this respect, **mboost** allows to fit potentially high-dimensional linear or smooth additive models, and it has methods to compute degrees of freedom which in turn allow for the use of information criteria such as AIC or BIC or for estimation of variance. Moreover, for high-dimensional (generalized) linear models, our implementation is fast enough to fit models in reasonable time when the dimension of the predictor space is in the ten-thousands.

The illustrations presented throughout the paper focus on three regression problems with continuous, binary and censored response variables and a potential large number of covariates. For each example, we only present the most important steps of the analysis. Because reproducibility is a natural and essential requirement for research on new statistical methods, the **mboost** package includes a *vignette* containing the complete sources of each analysis as well as the sources of our implementations of the boosting algorithms utilized in these analyses. Access to open-source software is not only important for reproducibility of numerical results but serves as a basis for further computational implementations and research in the area of boosting. The **mboost** package is freely available from http://CRAN.R-project.org and the reader can install our package directly from the R prompt via

```
R> install.packages("mboost", dependencies = TRUE)
R> library("mboost")
```

The rendered output of the data analyses presented in this paper is available by the R-command

```
R> vignette("mboost_illustrations", package = "mboost")
```

whereas the R code for reproducibility of our analyses can be assessed by

```
R> edit(vignette("mboost_illustrations", package = "mboost"))
```

Unless stated differently, we assume that the data are realizations of random variables

$$(X_1, Y_1), \dots, (X_n, Y_n)$$

from a stationary process with p-dimensional predictor variables X_i and one-dimensional response variables Y_i; for the case of multivariate responses, some references are given in Section 9.1. In particular, the setting above includes independent, identically distributed (*i.i.d.*) observations. In the sequel, the jth component of a vector c will be denoted by $c^{(j)}$.

1.1. *Ensemble schemes: multiple prediction and aggregation.* Ensemble schemes construct many function estimates or predictions from re-weighted data and use a linear (or sometimes convex) combination thereof for producing the final, aggregated estimator or prediction.

First, we specify a *base procedure* which constructs a function estimate $\hat{g}(\cdot)$ with values in $\mathbb{R}$, based on some data $(X_1, Y_1), \dots, (X_n, Y_n)$:

$$(X_1, Y_1), \dots, (X_n, Y_n) \quad \xrightarrow{\text{base procedure}} \quad \hat{g}(\cdot).$$

For example, a very popular base procedure is a regression tree.

Then, generating an ensemble from the base procedures, i.e., an ensemble of function estimates or predictions, works generally as follows:

$$\begin{array}{ccc}
\text{re-weighted data 1} & \xrightarrow{\text{base procedure}} & \hat{g}^{[1]}(\cdot) \\
\text{re-weighted data 2} & \xrightarrow{\text{base procedure}} & \hat{g}^{[2]}(\cdot) \\
 & & \\
\cdots & & \cdots \\
\cdots & & \cdots \\
 & & \\
\text{re-weighted data } M & \xrightarrow{\text{base procedure}} & \hat{g}^{[M]}(\cdot)
\end{array}$$

$$\text{aggregation: } \hat{f}_A(\cdot) = \sum_{m=1}^{M} \alpha_m \hat{g}^{[m]}(\cdot).$$

What is termed here with "re-weighted data" means that we have assigned individual data weights to every of the n sample points. We have also implicitly assumed that the base procedure allows to do some weighted fitting, i.e., estimation is based on a weighted sample. Throughout the paper (except in

Section 1.2), we assume that a base procedure estimate $\hat{g}(\cdot)$ is real-valued (i.e. a regression procedure) making it more adequate for the "statistical perspective" on boosting, in particular for the generic FGD algorithm in Section 2.1.

The above description of an ensemble scheme is too general to be of any direct use. The specification of the data re-weighting mechanism as well as the form of the linear combination coefficients $\{\alpha_m\}_{m=1}^M$ are crucial, and various choices characterize different ensemble schemes. Most boosting methods are special kinds of *sequential* ensemble schemes, where the data weights in iteration m depend on the results from the previous iteration $m-1$ only (*memoryless* with respect to iterations $m-2, m-3, \ldots$). Examples of other ensemble schemes include bagging [13] or random forests [2, 16].

1.2. *AdaBoost.* The AdaBoost algorithm for binary classification [28] is the most well known boosting algorithm. The base procedure is a classifier (slightly different from a real-valued function estimator as assumed above), e.g. a classification tree.

AdaBoost algorithm

1. Initialize some weights for individual sample points: $w_i^{[0]} = 1/n$ for $i = 1, \ldots, n$. Set $m = 0$.
2. Increase m by 1. Fit the base procedure to the weighted data, i.e., do a weighted fitting using the weights $w_i^{[m-1]}$, yielding the classifier $\hat{g}^{[m]}(\cdot)$.
3. Compute the weighted in-sample misclassification rate

$$\mathrm{err}^{[m]} = \sum_{i=1}^n w_i^{[m-1]} I(Y_i \neq \hat{g}^{[m]}(X_i)) / \sum_{i=1}^n w_i^{[m-1]},$$
$$\alpha^{[m]} = \log\left(\frac{1 - \mathrm{err}^{[m]}}{\mathrm{err}^{[m]}}\right),$$

and up-date the weights

$$\tilde{w}_i = w_i^{[m-1]} \exp(\alpha^{[m]} I(Y_i \neq \hat{g}^{[m]}(X_i))),$$
$$w_i^{[m]} = \tilde{w}_i / \sum_{j=1}^n \tilde{w}_j.$$

4. Iterate steps 2 and 3 until $m = m_{\mathrm{stop}}$ and build the aggregated classifier by weighted majority voting:

$$\hat{f}_{\mathrm{AdaBoost}}(x) = \underset{y \in \{0,1\}}{\mathrm{argmin}} \sum_{m=1}^{m_{\mathrm{stop}}} \alpha^{[m]} I(\hat{g}^{[m]}(x) = y).$$

By using the terminology m_{stop} (instead of M as in the general description of ensemble schemes), we emphasize here and later that the iteration process should be stopped to avoid overfitting. It is a tuning parameter of AdaBoost which may be selected using some cross-validation scheme.

1.3. *Historical remarks.*

The idea of boosting as an ensemble method for improving the predictive performance of a base procedure seems to have its root in machine learning. Kearns and Valiant [43] proved that if individual classifiers perform at least slightly better than guessing at random, their predictions can be combined and averaged yielding much better predictions. Later, Schapire [60] proposed a boosting algorithm with provable polynomial run-time to construct such a better ensemble of classifiers. The AdaBoost algorithm ([26–28]) is considered as a first path-breaking step towards practically feasible boosting algorithms.

The results from Breiman [14, 15], showing that boosting can be understood as a functional gradient descent algorithm, uncover older roots of boosting. In the context of regression, there is an immediate connection to the Gauss-Southwell algorithm [63] for solving a linear system of equations (see Section 4.1) and to Tukey's [67] method of "twicing" (see Section 5.1).

2. Functional gradient descent.

Breiman [14, 15] showed that the AdaBoost algorithm can be represented as a steepest descent algorithm in function space which we call functional gradient descent (FGD). Consider the problem of estimating a real-valued function

$$f^*(\cdot) = \operatorname*{argmin}_{f(\cdot)} \mathbb{E}[\rho(Y, f(X))], \tag{2.1}$$

where $\rho(\cdot,\cdot)$ is a loss function which is typically assumed to be differentiable and convex with respect to the second argument. For example, the squared error loss $\rho(y, f) = |y - f|^2$ yields the well-known population minimizer $f^*(x) = \mathbb{E}[Y|X = x]$.

2.1. *The generic FGD or boosting algorithm.*

In the sequel, FGD and boosting are used as equivalent terminology for the same method or algorithm.

Estimation of $f^*(\cdot)$ in (2.1) with boosting can be done by considering the empirical risk $n^{-1}\sum_{i=1}^{n} \rho(Y_i, f(X_i))$ and pursuing iterative steepest descent in function space as follows.

Generic FGD algorithm

1. Initialize $\hat{f}^{[0]}$ with an offset value. Common choices are $\hat{f}^{[0]} \equiv \overline{Y}$ or $\hat{f}^{[0]} \equiv 0$. Set $m = 0$.

2. Increase m by 1. Compute the negative gradient $-\frac{\partial}{\partial f}\rho(Y,f)$ and evaluate at $\hat{f}^{[m-1]}(X_i)$:
$$U_i = -\frac{\partial}{\partial f}\rho(Y,f)|_{f=\hat{f}^{[m-1]}(X_i)},\ i=1,\ldots,n.$$
3. Fit the negative gradient vector $U_1,\ldots,U_n$ to $X_1,\ldots,X_n$ by the real-valued base procedure (e.g. regression)
$$(X_i,U_i)_{i=1}^n \overset{\text{base procedure}}{\longrightarrow} \hat{g}^{[m]}(\cdot).$$
Thus, $\hat{g}^{[m]}(\cdot)$ can be viewed as an approximation of the negative gradient vector.
4. Up-date $\hat{f}^{[m]} = \hat{f}^{[m-1]}(\cdot) + \nu\cdot\hat{g}^{[m]}(\cdot)$, where $0<\nu\le 1$ is a step-length factor (see below), i.e. proceed along an estimate of the negative gradient vector.
5. Iterate steps 2 to 4 until $m = m_{\text{stop}}$ for some stopping iteration m_{stop}.

The stopping iteration, which is the main tuning parameter, can be estimated via cross-validation or some information criterion, see Section 5.4. The choice of the step-length factor ν in step 5 is of minor importance, as long as it is "small" such as $\nu = 0.1$. A smaller value of ν typically requires a larger number of boosting iterations and thus more computing time, while the predictive accuracy has been empirically found to be potentially better and almost never worse when choosing ν "sufficiently small" (e.g. $\nu = 0.1$) [29]. Friedman [29] suggests to use an additional line search between steps 3 and 4 (in case of different loss functions $\rho(\cdot,\cdot)$ than squared error): it yields a slightly different algorithm but the additional line search seems unnecessary to pursue for achieving a good estimator $\hat{f}^{[m_{\text{stop}}]}$.

2.1.1. *Alternative formulation in function space.* In steps 2 and 3 of the generic FGD algorithm, we associated with $U_1,\ldots,U_n$ a negative gradient vector. A reason for this can be seen from the following formulation in function space which is similar to the exposition in Mason et al. [49].

Consider the empirical risk functional $C(f) = n^{-1}\sum_{i=1}^n \rho(Y_i, f(X_i))$ and the usual inner product $\langle f,g\rangle = n^{-1}\sum_{i=1}^n f(X_i)g(X_i)$. We can then calculate the negative Gâteaux derivative $dC(\cdot)$ of the functional $C(\cdot)$,
$$-dC(f)(x) = -\frac{\partial}{\partial\alpha}C(f+\alpha\delta_x)|_{\alpha=0},\quad f:\mathbb{R}^p\to\mathbb{R},\ x\in\mathbb{R}^p,$$
where δ_x denotes the delta- (or indicator-) function at $x\in\mathbb{R}^p$. In particular, when evaluating the derivative $-dC$ at $\hat{f}^{[m-1]}$ and X_i, we get
$$-dC(\hat{f}^{[m-1]})(X_i) = U_i,$$

with $U_1, ..., U_n$ exactly as in steps 2 and 3 of the generic FGD algorithm. Thus, the negative gradient vector $U_1, \ldots, U_n$ can be interpreted as a functional (Gâteaux) derivative evaluated at the data points.

3. Some loss functions and boosting algorithms.

Various boosting algorithms can now be defined by specifying different loss functions $\rho(\cdot, \cdot)$. The **mboost** package provides infrastructure for defining loss functions via *boost_family* objects, as exemplified below.

3.1. *Binary classification.*

For binary classification where $Y \in \{0, 1\}$ with $\mathbb{P}[Y = 1] = p$, we may use the negative binomial log-likelihood as loss function:

$$-\left(y \log(p) + (1-y) \log(1-p)\right).$$

We parameterize $p = \exp(f)/(\exp(f)+\exp(-f))$ so that $f = \log(p/(1-p))/2$ equals half of the log-odds ratio; the factor 1/2 is a bit unusual but it will enable that the population minimizer of the loss in (3.1) will be the same as for the exponential loss in (3.3) below. The negative log-likelihood becomes then

$$\log(1 + \exp(-2(2y-1)f)).$$

By scaling, we prefer to use the equivalent loss function

$$\rho_{\text{log-lik}}(y, f) = \log_2(1 + \exp(-2\tilde{y}f)), \quad \tilde{y} = 2y - 1 \in \{-1, +1\} \tag{3.1}$$

which then becomes an upper bound of the misclassification error, see Figure 1. In **mboost**, the negative gradient of this loss function is implemented in a pre-fabricated function

```
R> bin <- Binomial()
```

returning an object of class *boost_family* which contains the negative gradient *function* as a slot (with input response $\mathtt{y} \in \{-1, +1\}$):

```
R> bin@ngradient
```

```
function (y, f)
{
    exp2yf <- exp(-2 * y * f)
    -(-2 * y * exp2yf)/(log(2) * (1 + exp2yf))
}
<environment: 0x8456d40>
```

The population minimizer can be shown to be

$$f^*_{\text{log-lik}}(x) = \frac{1}{2}\log\left(\frac{p(x)}{1-p(x)}\right), \quad p(x) = \mathbb{P}[Y = 1|X = x].$$

We point out that $\tilde{y} = 2y - 1 \in \{-1, +1\}$ is the usual encoding for binary responses in the machine learning literature (and it is used in **mboost** as well). The loss function in (3.1) becomes then a function of $\tilde{y}f$, the so-called margin value, where the function f induces the following classifier for Y:

$$\mathcal{C}(x) = \begin{cases} 1 & \text{if } f(x) > 0 \\ 0 & \text{if } f(x) < 0 \\ \text{undetermined} & \text{if } f(x) = 0. \end{cases}$$

Therefore, a misclassification (including the undetermined case) happens if and only if $\tilde{Y}f(X) \leq 0$. Hence, the misclassification loss is

$$\rho_{\text{0-1}}(y, f) = I_{\{\tilde{y}f \leq 0\}}, \quad \tilde{y} = 2y - 1 \in \{-1, +1\} \tag{3.2}$$

whose population minimizer is equivalent to the Bayes classifier (for $\tilde{y} \in \{-1, +1\}$)

$$f^*_{\text{0-1}}(x) = \begin{cases} +1 & \text{if } p(x) > 1/2 \\ -1 & \text{if } p(x) \leq 1/2, \end{cases}$$

where $p(x) = \mathbb{P}[Y = 1|X = x]$. Note that the 0-1 loss in (3.2) cannot be used for boosting or FGD: it is non-differentiable and also non-convex as a function of the margin value $\tilde{y}f$. The negative log-likelihood loss in (3.1) can be viewed as a convex upper approximation of the (computationally intractable) non-convex 0-1 loss, see Figure 1. We will describe in Section 3.3 the BinomialBoosting algorithm (similar to LogitBoost [30]) which uses the negative log-likelihood as implementing loss function.

Another upper convex approximation of the 0-1 loss function in (3.2) is the exponential loss

$$\rho_{\exp}(y, f) = \exp(-\tilde{y}f), \quad \tilde{y} = 2y - 1 \in \{-1, +1\} \tag{3.3}$$

implemented (again assuming `y` $\in \{-1, +1\}$) in **mboost** as

```
R> AdaExp()

         Adaboost Exponential Error

Loss function: exp(-y * f)
```

The population minimizer can be shown to be the same as for the log-likelihood loss:

$$f^*_{\exp}(x) = \frac{1}{2}\log\left(\frac{p(x)}{1-p(x)}\right), \quad p(x) = \mathbb{P}[Y=1|X=x].$$

Using functional gradient descent with different implementing loss functions yields different boosting algorithms. When using the log-likelihood loss in (3.1), we obtain LogitBoost [30] or BinomialBoosting from Section 3.3; and with the exponential loss in (3.3), we essentially get AdaBoost [27] from Section 1.2.

We interpret the boosting estimate $\hat{f}^{[m]}(x)$ as an estimate of the population minimizer $f^*(x)$. Thus, the output from AdaBoost, Logit- or BinomialBoosting are estimates of half the log odds ratio. In particular, we can obtain probability estimates via

$$\hat{p}^{[m]}(x) = \frac{\exp(\hat{f}^{[m]}(x))}{\exp(\hat{f}^{[m]}(x)) + \exp(-\hat{f}^{[m]}(x))}.$$

Also very popular in machine learning is the hinge function, the standard loss function for support vector machines:

$$\rho_{\mathrm{SVM}}(y,f) = [1-\tilde{y}f]_+, \ \tilde{y} = 2y-1 \in \{-1,+1\},$$

where $[x]_+ = xI_{\{x>0\}}$ denotes the positive part. It is also an upper convex bound of the misclassification error, see Figure 1. Its population minimizer is

$$f^*_{\mathrm{SVM}}(x) = \mathrm{sign}(p(x) - 1/2)$$

which is the Bayes classifier for $\tilde{Y} \in \{-1,+1\}$. Since $f^*_{\mathrm{SVM}}(\cdot)$ is a classifier and non-invertible function of $p(x)$ there is no direct way to obtain conditional class probability estimates.

[Fig 1 about here.]

3.2. *Regression.* For regression with response $Y \in \mathbb{R}$, we use most often the squared error loss (scaled by the factor 1/2 such that the negative gradient vector equals the residuals, see Section 3.3 below),

$$\rho_{L_2}(y,f) = \frac{1}{2}|y-f|^2 \qquad (3.4)$$

with population minimizer

$$f^*_{L_2}(x) = \mathbb{E}[Y|X=x].$$

The corresponding boosting algorithm is L_2Boosting, see Friedman [29] and Bühlmann and Yu [19]. It is described in more detail in Section 3.3. This loss function is available in **mboost** (without the irrelevant factor 1/2) as

```
R> GaussReg()

         Squared Error (Regression)

Loss function: (y - f)^2
```

Alternative loss functions which have some robustness properties (with respect to the error distribution, i.e., in "Y-space") include the L_1- and Huber-loss. The former is

$$\rho_{L_1}(y, f) = |y - f|$$

with population minimizer

$$f^*(x) = \text{median}(Y|X = x)$$

and is implemented in **mboost** as

```
R> Laplace()

         Absolute Error

Loss function: abs(y - f)
```

Although the L_1-loss is not differentiable at the point $y = f$, we can compute partial derivatives since the single point $y = f$ (usually) has probability zero to be realized by the data. A compromise between the L_1- and L_2-loss is the Huber-loss function from robust statistics:

$$\rho_{\text{Huber}}(y, f) = \begin{cases} |y - f|^2/2, & \text{if } |y - f| \leq \delta \\ \delta(|y - f| - \delta/2), & \text{if } |y - f| > \delta. \end{cases}$$

which is available in **mboost** as (here with $\delta = 2$)

```
R> Huber(d = 2)

         Huber Error (with d = 2)

Loss function: ifelse((a <- abs(y - f)) < d,
                      a^2/2, d * (a - d/2))
```

A strategy for choosing (a changing) δ adaptively has been proposed by Friedman [29]:

$$\delta_m = \text{median}(\{|Y_i - \hat{f}^{[m-1]}(X_i)|;\ i = 1, \ldots, n\}),$$

where the previous fit $\hat{f}^{[m-1]}(\cdot)$ is used. This method is available in **mboost** via `Huber(d = NULL)`.

3.2.1. *Connections to binary classification.* We point out that the L_2- or L_1-loss can also be used for binary classification. For $Y \in \{0, 1\}$, the population minimizers are then

$$f^*_{L_2}(x) = \mathbb{E}[Y|X = x] = p(x) = \mathbb{P}[Y = 1|X = x],$$
$$f^*_{L_1}(x) = \text{median}(Y|X = x) = \begin{cases} 1 & \text{if } p(x) > 1/2 \\ 0 & \text{if } p(x) \leq 1/2. \end{cases}$$

Thus, the population minimizer of the L_1-loss is the Bayes classifier.

Both L_2- and L_1-loss functions can be parameterized as functions of the margin value $\tilde{y}f$ with $\tilde{y} = 2y - 1 \in \{-1, +1\}$:

$$|\tilde{y} - f|^2 = 1 - 2\tilde{y}f + (\tilde{y}f)^2,$$
$$|\tilde{y} - f| = |1 - \tilde{y}f|. \tag{3.5}$$

The L_2-loss for classification (with `y` $\in \{-1, +1\}$) is implemented in **mboost** as

```
R> GaussClass()

         Squared Error (Classification)

Loss function: 1 - 2 * y * f + (y * f)^2
```

All loss functions mentioned for binary classification (displayed in Figure 1) can be viewed and interpreted from the perspective of proper scoring rules, cf. Buja et al. [21]. We usually prefer the negative log-likelihood loss in (3.1) because: (i) it yields probability estimates; (ii) it is a monotone loss function of the margin value $\tilde{y}f$; (iii) it grows linearly as the margin value $\tilde{y}f$ tends to $-\infty$, unlike the exponential loss in (3.3). The third point reflects a robustness aspect: it is similar to Huber's loss function which also penalizes large values linearly instead of quadratically (as with the L_2-loss).

3.2.2. *Specifying a loss function of your own choice in* **mboost**. The `Family` function in **mboost** can be used to create an object of class *boost_family* implementing the negative gradient for general loss functions. Such an object can later be fed into the fitting procedure of a linear or additive model which optimizes the corresponding empirical risk. An example is given in Section 5.2.

3.3. *Two important boosting algorithms.* Table 1 summarizes the most popular loss functions and their corresponding boosting algorithms.

[Table 1 about here.]

We describe now in detail the two algorithms appearing in the last two rows of Table 1.

3.3.1. *L_2Boosting.* L_2Boosting is the simplest and most instructive boosting algorithm. It is very useful for regression, in particular when there are very many predictor variables. Applying the general description of the FGD-algorithm from Section 2.1 to the squared error loss function $\rho_{L_2}(y, f) = |y - f|^2/2$, we obtain the following algorithm.

L_2Boosting algorithm

1. Initialize $\hat{f}^{[0]}$ with an offset value. The default value is $\hat{f}^{[0]} \equiv \overline{Y}$. Set $m = 0$.
2. Increase m by 1. Compute the residuals $U_i = Y_i - \hat{f}^{[m-1]}(X_i)$ for $i = 1, \ldots, n$.
3. Fit the residual vector $U_1, \ldots, U_n$ to $X_1, \ldots, X_n$ by the real-valued base procedure (e.g. regression)
$$(X_i, U_i)_{i=1}^n \xrightarrow{\text{base procedure}} \hat{g}^{[m]}(\cdot).$$
4. Up-date $\hat{f}^{[m]} = \hat{f}^{[m-1]}(\cdot) + \nu \cdot \hat{g}^{[m]}(\cdot)$, where $0 < \nu \le 1$ is a step-length factor (as in the general FGD-algorithm).
5. Iterate steps 2 to 4 until $m = m_{\text{stop}}$ for some stopping iteration m_{stop}.

The stopping iteration m_{stop} is the main tuning parameter which can be selected using cross-validation or some information criterion as described in Section 5.4.

The derivation from the generic FGD algorithm in Section 2.1 is straightforward. Note that the negative gradient vector becomes the residual vector. Thus, L_2Boosting amounts to refitting residuals multiple times. Tukey [67] recognized this to be useful and proposed "twicing" which is nothing else than L_2Boosting using $m_{\text{stop}} = 2$ (and $\nu = 1$).

3.3.2. *BinomialBoosting: the FGD version of LogitBoost.* We already gave some reasons at the end of Section 3.2.1 why the negative log-likelihood loss function in (3.1) is very useful for binary classification problems. Friedman et al. [30] proposed LogitBoost which is very similar to the generic FGD algorithm when using the loss from (3.1): they propose to use Newton's method involving the Hessian matrix.

For the sake of coherence with the generic functional gradient descent algorithm in Section 2.1, we describe here a version of LogitBoost: to avoid conflicts of terminology, we call it BinomialBoosting.

BinomialBoosting algorithm

Apply the generic FGD algorithm from Section 2.1 using the loss function $\rho_{\text{log-lik}}$ from (3.1). The default offset value is $\hat{f}^{[0]} = \log(\hat{p}/(1 - \hat{p}))/2$, where $\hat{p}$ is the relative frequency of $Y = 1$ (i.e. $\hat{p} = \overline{Y}$).

With BinomialBoosting, there is no need that the base procedure is able to do weighted fitting: this constitutes a slight difference to the requirement for LogitBoost [30].

3.4. *Other data structures and models.* Due to the generic nature of boosting or functional gradient descent, we can use the technique in very many other settings. For data with univariate responses and loss functions which are differentiable with respect to the second argument, the boosting approach to be followed is described in Section 2.1; see also Section 3.2.2. Survival analysis is an important area of application with censored observations: we describe how one can deal with it in Section 8.

4. Choosing the base procedure.

Every boosting algorithm requires the specification of a base procedure. This choice can be driven by the aim of optimizing the predictive capacity only or by considering some structural properties of the boosting estimate in addition. We find the latter usually more interesting as it allows for better interpretation of the resulting model.

We recall that the generic boosting estimator is a sum of base procedure estimates

$$\hat{f}^{[m]}(\cdot) = \nu \sum_{k=1}^{m} \hat{g}^{[k]}(\cdot).$$

Therefore, structural properties of the boosting function estimator are induced by a linear combination of structural characteristics of the base procedure.

The following important examples of base procedures yield useful structures for the boosting estimator $\hat{f}^{[m]}(\cdot)$. The notation is as follows: $\hat{g}(\cdot)$ is an estimate from a base procedure which is based on data $(X_1, U_1), \ldots, (X_n, U_n)$ where $(U_1, \ldots, U_n)$ denotes the current negative gradient.

4.1. *Componentwise linear least squares for linear models.* Consider the base procedure

$$\begin{aligned} &\hat{g}(x) = \hat{\beta}^{(\hat{\mathcal{S}})} x^{(\hat{\mathcal{S}})}, \\ &\hat{\beta}^{(j)} = \sum_{i=1}^{n} X_i^{(j)} U_i \Big/ \sum_{i=1}^{n} (X_i^{(j)})^2, \; \hat{\mathcal{S}} = \operatorname*{argmin}_{1 \le j \le p} \sum_{i=1}^{n} (U_i - \hat{\beta}^{(j)} X_i^{(j)})^2. \end{aligned} \tag{4.1}$$

It selects the best variable in a simple linear model in the sense of ordinary least squares fitting.

When using L_2Boosting with this base procedure, we select in every iteration one predictor variable, not necessarily a different one for each iteration, and we up-date the function linearly:

$$\hat{f}^{[m]}(x) = \hat{f}^{[m-1]}(x) + \nu \hat{\beta}^{(\hat{\mathcal{S}}_m)} x^{(\hat{\mathcal{S}}_m)},$$

where $\hat{\mathcal{S}}_m$ denotes the index of the selected predictor variable in iteration m. Alternatively, the up-date of the coefficient estimates is

$$\hat{\beta}^{[m]} = \hat{\beta}^{[m-1]} + \nu \cdot \hat{\beta}^{(\hat{\mathcal{S}}_m)}.$$

The notation should be read that only the $\hat{\mathcal{S}}_m$th component of the coefficient estimate $\hat{\beta}^{[m]}$ (in iteration m) has been up-dated. For every iteration m, we obtain a linear model fit. As m tends to infinity, $\hat{f}^{[m]}(\cdot)$ converges to a least squares solution which is unique if the design matrix has full rank $p \leq n$. The method is also known as matching pursuit in signal processing [47], weak greedy algorithm in computational mathematics [65], and it is a Gauss-Southwell algorithm [63] for solving a linear system of equations. We will discuss more properties of L_2Boosting with componentwise linear least squares in Section 5.2.

When using BinomialBoosting with componentwise linear least squares from (4.1), we obtain a fit, including variable selection, of a linear logistic regression model.

As will be discussed in more details in Section 5.2, boosting typically shrinks the (logistic) regression coefficients towards zero. Usually, we do not want to shrink the intercept: we advocate to use boosting on mean centered predictor variables $\tilde{X}_i^{(j)} = X_i^{(j)} - n^{-1}\sum_{k=1}^n X_k^{(j)}$ without an intercept term in the model. In case of a linear model, when centering also the response $\tilde{Y}_i = Y_i - n^{-1}\sum_{k=1}^n Y_k$, this becomes

$$\tilde{Y}_i = \sum_{j=1}^p \beta^{(j)} \tilde{X}_i^{(j)} + \text{ noise}_i$$

which forces the regression surface through the center $(\tilde{x}^{(1)}, \ldots, \tilde{x}^{(p)}, \tilde{y}) = (0, 0, \ldots, 0)$ as with ordinary least squares. Note that it is not necessary to center the response variables when using the default offset value $\hat{f}^{[0]} = \overline{Y}$ in L_2Boosting (for BinomialBoosting, we would center the predictor variables only but never the response).

Illustration: Prediction of Total Body Fat. Garcia et al. [31] report on the development of predictive regression equations for body fat content by means of $p = 9$ common anthropometric measurements which were obtained for $n = 71$ healthy German women. In addition, the women's body composition was measured by Dual Energy X-Ray Absorptiometry (DXA). This reference method is very accurate in measuring body fat but finds little applicability in practical environments, mainly because of high costs and the methodological efforts needed. Therefore, a simple regression equation for predicting DXA measurements of body fat is of special interest for the practitioner. Backward-elimination was applied to select important variables from the available anthropometrical measurements and Garcia et al. [31] report a final linear model utilizing hip circumference, knee breadth and a compound covariate which is defined as the sum of log chin skinfold, log triceps skinfold and log subscapular skinfold:

```
R> bf_lm <- lm(DEXfat ~ hipcirc + kneebreadth + anthro3a,
        data = bodyfat)
R> coef(bf_lm)
```

```
(Intercept)     hipcirc kneebreadth    anthro3a
  -75.23478     0.51153     1.90199     8.90964
```

Since a simple and easy to communicate regression formula, such as a linear combination of only a few covariates, is of special interest in this application, we employ the `glmboost` function from package **mboost** to fit a linear regression model by means of L_2Boosting with componentwise linear least squares. We first center the covariates and specify a formula describing the model we want to fit:

```
R> indep <- names(bodyfat)[names(bodyfat) != "DEXfat"]
R> cbodyfat <- bodyfat
R> cbodyfat[indep] <- lapply(cbodyfat[indep],
      function(x) x - mean(x))
R> bffm <- DEXfat ~ age + waistcirc + hipcirc + elbowbreadth +
        kneebreadth + anthro3a + anthro3b + anthro3c +
        anthro4
```

By default, the function `glmboost` fits a linear model (with initial $m_{\text{stop}} = 100$ and shrinkage parameter $\nu = 0.1$) by minimizing squared error (argument `family = GaussReg()` is the default):

```
R> bf_glm <- glmboost(bffm, data = cbodyfat)
```

Note that, by default, the mean of the response variable is used as an offset in the first step of the boosting algorithm. As mentioned above, the special form of the base learner, i.e., componentwise linear least squares, allows for a reformulation of the boosting fit in terms of a linear combination of the covariates which can be assessed via

```
R> coef(bf_glm)
```

```
 (Intercept)          age   waistcirc     hipcirc
    0.000000     0.013602    0.189716    0.351626
elbowbreadth  kneebreadth    anthro3a    anthro3b
   -0.384140     1.736589    3.326860    3.656524
    anthro3c      anthro4
    0.595363     0.000000
```

[Fig 2 about here.]

We notice that most covariates have been used for fitting and thus no extensive variable selection was performed in the above model. Thus, we need to investigate how many boosting iterations are appropriate. Resampling methods such as cross-validation or the bootstrap can be used to study the empirical risk for a varying number of boosting iterations. The out-of-bootstrap mean squared error for 100 bootstrap samples is depicted in the upper part of Figure 2. The plot leads to the impression that approximately $m_{\text{stop}} = 44$ would be a sufficient number of boosting iterations. In Section 5.4, a corrected version of the Akaike information criterion (AIC) is proposed for determining the optimal number of boosting iterations. This criterion attains its minimum for

```
R> mstop(aic <- AIC(bf_glm))
```

```
[1] 45
```

boosting iterations, see the bottom part of Figure 2 in addition.

The coefficients of the boosted linear model with $m_{\text{stop}} = 45$ boosting iterations are

```
R> coef(bf_glm[mstop(aic)])
```

```
 (Intercept)          age   waistcirc     hipcirc
   0.0000000    0.0023271   0.1893046   0.3488781
elbowbreadth  kneebreadth    anthro3a    anthro3b
   0.0000000    1.5217686   3.3268603   3.6051548
    anthro3c      anthro4
   0.5043133    0.0000000
```

and thus only 7 covariates have been selected for the final model (intercept equal to zero occurs here for mean centered response and predictors and hence, $n^{-1}\sum_{i=1}^{n} Y_i = 30.783$ is the intercept in the uncentered model). Note that the variables `hipcirc`, `kneebreadth` and `anthro3a`, which we have used for fitting a simple linear model at the beginning of this paragraph, have been selected by the boosting algorithm as well.

4.2. *Componentwise smoothing spline for additive models.* We may choose a nonparametric base procedure for function estimation. Suppose that

$$\hat{f}^{(j)}(\cdot) \text{ is a least squares smoothing spline estimate based on } U_1, \ldots, U_n \text{ against } X_1^{(j)}, \ldots, X_n^{(j)} \text{ with fixed degrees of freedom df.} \tag{4.2}$$

That is,

$$\hat{f}^{(j)} = \underset{f}{\operatorname{argmin}} \sum_{i=1}^{n} (U_i - f(X_i^{(j)}))^2 + \lambda \int (f''(x))^2 dx, \tag{4.3}$$

where $\lambda > 0$ is a tuning parameter such that the trace of the corresponding hat matrix equals df. For further details, we refer to Green and Silverman [33].

The base procedure is then

$$\hat{g}(x) = \hat{f}^{(\hat{\mathcal{S}})}(x^{(\hat{\mathcal{S}})}),$$
$$\hat{f}^{(j)}(\cdot) \text{ as above and } \hat{\mathcal{S}} = \underset{1 \le j \le p}{\operatorname{argmin}} \sum_{i=1}^{n} (U_i - \hat{f}^{(j)}(X_i^{(j)}))^2,$$

where the degrees of freedom df is the same for all $\hat{f}^{(j)}(\cdot)$.

L_2Boosting with the componentwise smoothing spline yields an additive model, including variable selection, i.e., a fit which is additive in the predictor variables. This can be seen immediately since L_2Boosting proceeds additively for up-dating the function $\hat{f}^{[m]}(\cdot)$, see Section 3.3. Typically, we then normalize and obtain the following additive model estimator:

$$\hat{f}^{[m]}(x) = \hat{\mu} + \sum_{j=1}^{p} \hat{f}^{[m],(j)}(x^{(j)}),$$
$$n^{-1} \sum_{i=1}^{n} \hat{f}^{[m],(j)}(X_i^{(j)}) = 0 \text{ for all } j = 1, \ldots, p.$$

As with the componentwise linear least squares base procedure, we can use componentwise smoothing splines also in BinomialBoosting, yielding an additive logistic regression fit.

The degrees of freedom in the base procedure should be chosen "small" such as df = 4. This yields low variance but typically large bias of the base procedure. The bias can then be reduced by additional boosting iterations. This choice of low variance but high bias has been analysed in Bühlmann and Yu [19], see also Section 4.4.

Componentwise smoothing splines can be generalized to pairwise smoothing splines which searches for and fits the best pairs of predictor variables such that a smooth of $U_1, \ldots, U_n$ against this pair of predictors reduces residual sum of squares most. With L_2Boosting, this yields a nonparametric model fit with first order interaction terms. This procedure has been empirically demonstrated to be often much better than fitting with MARS [20].

Illustration: Prediction of Total Body Fat (cont.). Being more flexible than the linear model which we fitted to the `bodyfat` data in Section 4.1, we estimate an additive model using the `gamboost` function from **mboost** (first with pre-specified $m_{\text{stop}} = 100$ boosting iterations, $\nu = 0.1$ and squared error loss):

```
R> bf_gam <- gamboost(bffm, data = cbodyfat)
```

The degrees of freedom for the smoothing splines, which are utilized as base learners here, can be defined by the `dfbase` argument, defaulting to 4.

We can estimate the number of boosting iterations m_{stop} using the corrected AIC criterion described in Section 5.4 (see Figure 3) via

```
R> mstop(aic <- AIC(bf_gam))
[1] 46
R> bf_gam <- bf_gam[mstop(aic)]
```

Similar to the linear regression model, the partial contributions of the covariates can be extracted from the boosting fit. For the most important variables, the partial fits are given in Figure 4 showing some slight non-linearity for `hipcirc` and `kneebreadth`.

[Fig 3 about here.]

[Fig 4 about here.]

4.3. *Trees.*

For boosting, regression trees are the most popular base procedures in machine learning. They have the advantage to be invariant under monotone transformations of predictor variables, i.e., we do not need to search for good data transformations. Unfortunately, the superb interpretation of a single tree is lost in boosting consisting of a linear combination of tree estimates. However, some weaker structural properties still hold.

When using stumps, i.e., a tree with two terminal nodes only, the boosting estimate will be an additive model in the original predictor variables, because every stump-estimate is a function of a single predictor variable only. Similarly, boosting trees with (at most) d terminal nodes results in a nonparametric model having at most interactions of order $d-2$. Therefore, if we want to constrain the degree of interactions, we can easily do this by constraining the (maximal) number of nodes in the base procedure.

Illustration: Prediction of Total Body Fat (cont.). Both the **gbm** package [57] and the **mboost** package are helpful when decision trees are to be used as base learners. In **mboost**, the function `blackboost` implements gradient boosting for fitting such classical *black box* models, e.g. to the `bodyfat` data, via

```
R> bf_black <- blackboost(bffm, data = bodyfat,
                          control = boost_control(mstop = 500))
```

Conditional inference trees [39] as available from the **party** package [38] are utilized as base learners. Here, the function `boost_control` defines hyper parameters, for example the number of boosting iterations `mstop`.

Alternatively, we can use the function `gbm` from the **gbm** package as follows:

```
R> bf_gbm <- gbm(bffm, data = bodyfat, distribution = "gaussian",
          interaction.depth = 2, n.trees = 500, shrinkage = 0.1,
          bag.fraction = 1, train.fraction = 1, verbose = FALSE)
```

which yields roughly the same fit as can be seen from Figure 5.

[Fig 5 about here.]

4.4. *The bias-variance trade-off.* We have seen above that the structural properties of a boosting estimate are determined by the choice of a base procedure. In our opinion, the structure specification should come first. After having made a choice, the question becomes how "complex" the base procedure should be. For example, how should we choose the degrees of freedom for the componentwise smoothing spline in (4.2)? A general answer is: choose the base procedure (having the desired structure) with low variance at the price of larger estimation bias. For the componentwise smoothing splines, this would imply a low number of degrees of freedom, e.g. df $= 4$.

We give some reasons for the low-variance principle in Section 5.1 (Replica 1). Moreover, it has been demonstrated in Friedman [29] that a small step-size factor ν can be often beneficial and almost never yields substantially worse predictive performance of boosting estimates. Note that a small step-size factor can be seen as a shrinkage of the base procedure by the factor ν, implying low variance but potentially large estimation bias.

5. L_2Boosting.

L_2Boosting is functional gradient descent using the squared error loss which amounts to repeated fitting of ordinary residuals, as mentioned already in Section 3.3.1. Here, we aim at increasing the understanding of the simple L_2Boosting algorithm. We first start with a toy problem of curve estimation, and we will then illustrate concepts and results which are especially useful for high-dimensional data. These can serve as heuristics for boosting algorithms with other convex loss functions for problems in e.g. classification or survival analysis.

5.1. *Nonparametric curve estimation: from basics to asymptotic optimality.*

Consider the toy problem of estimating a regression function $\mathbb{E}[Y|X = x]$ with one-dimensional predictor $X \in \mathbb{R}$ and a continuous response $Y \in \mathbb{R}$. We illustrate L_2Boosting using a smoothing spline with pre-specified degrees of freedom df (cf. formula (4.3)), and we review here some parts of the arguments and results from Bühlmann and Yu [19].

Consider the case with a linear base procedure having a hat matrix $\mathcal{H} : \mathbb{R}^n \to \mathbb{R}^n$, mapping the response variables $\mathbf{Y} = (Y_1, \ldots, Y_n)^\top$ to their fitted values $(\hat{f}(X_1), \ldots, \hat{f}(X_n))^\top$. Examples include nonparametric kernel smoothers or smoothing splines. It is easy to show that the hat matrix of the L_2Boosting fit (for simplicity, with $\hat{f}^{[0]} \equiv 0$ and $\nu = 1$) in iteration m equals:

$$\mathcal{B}_m = \mathcal{B}_{m-1} + \mathcal{H}(I - \mathcal{B}_{m-1}) = I - (I - \mathcal{H})^m. \tag{5.1}$$

Formula (5.1) allows for several insights. First, if the base procedure satisfies $\|I - \mathcal{H}\| < 1$ for a suitable norm, i.e., has a "learning capacity" such that the residual vector is shorter than the input-response vector, we see that $\mathcal{B}_m$ converges to the identity I as $m \to \infty$, and $\mathcal{B}_m\mathbf{Y}$ converges to the fully saturated model $\mathbf{Y}$, interpolating the response variables exactly. Thus, we see here explicitly that we have to stop early with the boosting iterations in order to prevent over-fitting.

When specifying to the case of a smoothing spline base procedure, it is useful to invoke some eigen-analysis. The spectral representation is

$$\mathcal{H} = UDU^\top, \;\; U^\top U = UU^\top = I, \;\; D = \mathrm{diag}(\lambda_1, \ldots, \lambda_n),$$

where $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_n$ denote the (ordered) eigenvalues of $\mathcal{H}$. It then follows with (5.1) that

$$\begin{aligned} &\mathcal{B}_m = UD_mU^\top, \\ &D_m = \mathrm{diag}(d_{1,m}, \ldots, d_{n,m}), \quad d_{i,m} = 1 - (1 - \lambda_i)^m. \end{aligned}$$

It is well known that a smoothing spline satisfies:

$$\lambda_1 = \lambda_2 = 1, \quad 0 < \lambda_i < 1 \ (i = 3, \dots, n).$$

Therefore, the eigenvalues of the boosting hat operator (matrix) in iteration m satisfy:

$$\begin{aligned}&d_{1,m} \equiv d_{2,m} \equiv 1 \text{ for all m},\\ &0 < d_{i,m} = 1 - (1 - \lambda_i)^m < 1 \ (i = 3, \dots, n), \ d_{i,m} \to 1 \ (m \to \infty).\end{aligned}$$

When comparing the spectrum, i.e., the set of eigenvalues, of a smoothing spline with its boosted version, we see the following. For both cases, the largest two eigenvalues are equal to 1. Moreover, all other eigenvalues can be changed by either varying the degrees of freedom $\mathrm{df} = \sum_{i=1}^n \lambda_i$ in a single smoothing spline, or by varying the boosting iteration m with some fixed (low-variance) smoothing spline base procedure having fixed (low) values λ_i. In Figure 6 we demonstrate the difference between the two approaches for changing "complexity" of the estimated curve fit by means of a toy example first shown in [19]. Both methods have about the same minimum mean squared error but L_2Boosting overfits much more slowly than a single smoothing spline.

[Fig 6 about here.]

By careful inspection of the eigen-analysis for this simple case of boosting a smoothing spline, Bühlmann and Yu [19] proved an asymptotic minimax rate result:

Replica 1. *[19] When stopping the boosting iterations appropriately, i.e. $m = m_n = O(n^{4/(2\xi+1)})$, $m_n \to \infty$ $(n \to \infty)$ with $\xi \geq 2$ as below, L_2Boosting with cubic smoothing splines having* fixed *degrees of freedom achieves the minimax convergence rate over Sobolev function classes of smoothness degree $\xi \geq 2$, as $n \to \infty$.*

Two items are interesting. First, minimax rates are achieved by using a base procedure with fixed degrees of freedom which means low variance from an asymptotic perspective. Secondly, L_2Boosting with cubic smoothing splines has the capability to *adapt* to higher order smoothness of the true underlying function: thus, with the stopping iteration as the one and only tuning parameter, we can nevertheless adapt to any higher-order degree of smoothness (without the need of choosing a higher order spline base procedure).

Recently, asymptotic convergence and minimax rate results have been established for early-stopped boosting in more general settings [9, 75].

5.1.1. *L_2Boosting using kernel estimators.* As we have pointed out in Replica 1, L_2Boosting of smoothing splines can achieve faster mean squared error convergence rates than the classical $O(n^{-4/5})$, assuming that the true underlying function is sufficiently smooth. We illustrate here a related phenomenon with kernel density estimators.

We consider fixed, univariate design points $x_i = i/n$ $(i = 1, \ldots, n)$ and the Nadaraya-Watson kernel estimator for the nonparametric regression function $\mathbb{E}[Y|X = x]$:

$$\hat{g}(x; h) = (nh)^{-1} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) Y_i = n^{-1} \sum_{i=1}^{n} K_h(x - x_i) Y_i,$$

where $h > 0$ is the bandwidth, $K(\cdot)$ a kernel in the form of a probability density which is symmetric around zero and $K_h(x) = h^{-1}K(x/h)$. It is straightforward to derive the form of L_2Boosting using $m = 2$ iterations (with $\hat{f}^{[0]} \equiv 0$ and $\nu = 1$), i.e. twicing [67], with the Nadaraya-Watson kernel estimator:

$$\hat{f}^{[2]}(x) = (nh)^{-1} \sum_{i=1}^{n} K_h^{tw}(x - x_i) Y_i,\ K_h^{tw}(u) = 2K_h(u) - K_h * K_h(u),$$

$$\text{where } K_h * K_h(u) = n^{-1} \sum_{r=1}^{n} K_h(u - x_r) K_h(x_r).$$

For fixed design points $x_i = i/n$, the kernel $K_h^{tw}(\cdot)$ is asymptotically equivalent to a higher-order kernel (which can take negative values) yielding a squared bias term of order $O(h^8)$, assuming that the true regression function is four times continuously differentiable. Thus, twicing or L_2Boosting with $m = 2$ iterations amounts to be a Nadaraya-Watson kernel estimator with a higher-order kernel. This explains from another angle why boosting is able to improve the mean squared error rate of the base procedure. More details including also non-equispaced designs are given in DiMarzio and Taylor [24].

5.2. *L_2Boosting for high-dimensional linear models.* Consider a potentially high-dimensional linear model

$$Y_i = \beta_0 + \sum_{j=1}^{p} \beta^{(j)} X_i^{(j)} + \varepsilon_i,\ i = 1, \ldots, n, \tag{5.2}$$

where $\varepsilon_1, \ldots, \varepsilon_n$ are i.i.d. with $\mathbb{E}[\varepsilon_i] = 0$ and independent from all X_i's. We allow for the number of predictors p to be much larger than the sample size n.

The model encompasses the representation of a noisy signal by an expansion with an over-complete dictionary of functions $\{g^{(j)}(\cdot) : j = 1, \ldots, p\}$; e.g. for surface modeling with design points in $Z_i \in \mathbb{R}^2$,

$$Y_i = f(Z_i) + \varepsilon_i, \quad f(z) = \sum_j \beta^{(j)} g^{(j)}(z) \ (z \in \mathbb{R}^2).$$

Fitting the model (5.2) can be done using L_2Boosting with the componentwise linear least squares base procedure from Section 4.1 which fits in every iteration the best predictor variable reducing the residual sum of squares most. This method has the following basic properties:

1. As the number m of boosting iterations increases, the L_2Boosting estimate $\hat{f}^{[m]}(\cdot)$ converges to a least squares solution. This solution is unique if the design matrix has full rank $p \leq n$.
2. When stopping early which is usually needed to avoid over-fitting, the L_2Boosting method often does variable selection.
3. The coefficient estimates $\hat{\beta}^{[m]}$ are (typically) shrunken versions of a ordinary least squares estimate $\hat{\beta}_{\text{OLS}}$ which are "related" to the Lasso, see Section 5.2.1.

Illustration: Breast Cancer Subtypes. Variable selection is especially important in high-dimensional situations. As an example, we study a binary classification problem involving $p = 7129$ gene expression levels in $n = 49$ breast cancer tumor samples [data taken from 74]. For each sample, a binary response variable describing the lymph node status of the patient is to be explained by the expression profiles.

The data are stored in form of an *exprSet* object [see 32] and we first extract the matrix of expression levels and the response variable, and center the expression levels for each gene around zero

```
R> x <- exprs(westbc)
R> x <- t(x - rowMeans(x))
R> y <- pData(westbc)$nodal.y
R> dim(x)
```

```
[1]   49 7129
```

```
R> table(y)
```

```
y
 0  1
25 24
```

We aim at using L_2Boosting for classification, see Section 3.2.1, with classical AIC based on the binomial log-likelihood. Thus, we first transform the factor y to a numeric variable with 0/1 coding (different from the $-1/+1$-encoding in Section 3.1):

```
R> yfit <- as.numeric(y) - 1
```

The general framework implemented in **mboost** allows us to specify the negative gradient corresponding to the implementing loss function (the `ngradient` argument), here the squared error loss, and a different evaluating loss function (the `loss` argument), here the negative log-likelihood, with the `Family` function as follows:

```
R> L2fm <- Family(ngradient = function(y, f) y - f,
                  loss = function(y, f, w = 1) {
                      p <- pmax(pmin(1 - 1e-05, f), 1e-05)
                      -y * log(p) - (1 - y) * log(1 - p)
                  },
                  offset = function(y, w)
                      weighted.mean(y, w))
```

The resulting object can now be passed to the `glmboost` function for boosting with componentwise linear least squares (here initial $m_{\text{stop}} = 200$ iterations are used):

```
R> west_glm <- glmboost(x, yfit, family = L2fm,
                        control = boost_control(mstop = 200))
```

Fitting such a linear model to $p = 7129$ covariates for $n = 49$ observations takes about 2 seconds on a medium scale desktop computer (Intel Pentium 4, 2.8GHz). Thus, this form of estimation and variable selection is computationally very efficient. As a comparison, computing all Lasso solutions, using the **lars** in R (with `use.gram=FALSE`) [25], is about a factor three slower.

The question how to choose m_{stop} remains open and can be addressed by the classical AIC criterion (see Section 5.4) as follows

```
R> aic <- AIC(west_glm, method = "classical")
R> mstop(aic)
[1] 100
```

where the AIC is computed as -2(log-likelihood) + 2(degrees of freedom) = 2 (evaluating loss) + 2(degrees of freedom). The notion of degrees of freedom is discussed in Section 5.3.

Figure 7 shows the AIC curve depending on the number of boosting iterations. When we stop after $m_{\text{stop}} = 100$ boosting iterations, we obtain 33 genes with non-zero regression coefficients whose standardized values $\hat{\beta}^{(j)}\sqrt{\widehat{\text{Var}}(X^{(j)})}$ are depicted in the left panel of Figure 7.

[Fig 7 about here.]

5.2.1. *Connections to the Lasso.* There is an intriguing connection between L_2Boosting with componentwise linear least squares and the Lasso [66] which is the following ℓ^1-penalty method:

$$\hat{\beta}(\lambda) = \underset{\beta}{\operatorname{argmin}}\, n^{-1}\sum_{i=1}^{n}\left(Y_i - \beta_0 - \sum_{j=1}^{p}\beta^{(j)}X_i^{(j)}\right)^2 + \lambda\sum_{j=1}^{p}|\beta^{(j)}|. \tag{5.3}$$

Efron et al. [25] consider a version of L_2Boosting, called forward stagewise linear regression (FSLR), and they show that FSLR with infinitesimally small step-sizes (i.e., the value ν in step 4 of the L_2Boosting algorithm in Section 3.3.1) produces a set of solutions which is approximately equivalent to the set of Lasso solutions when varying the regularization parameter λ in Lasso (see (5.3) above). The approximate equivalence is derived by representing FSLR and Lasso as two different modifications of the computationally efficient least angle regression (LARS) algorithm from Efron et al. [25]. The latter is very similar to the algorithm proposed earlier by Osborne et al. [52]. In special cases where the design matrix satisfies a "positive cone condition", FSLR, Lasso and LARS all coincide [25, p.425]. For more general situations, when adding some backward steps to boosting, such modified L_2Boosting coincides with the Lasso (Zhao and Yu [77]).

Despite the fact that L_2Boosting and Lasso are not equivalent methods in general, it may be useful to interpret boosting as being "related" to ℓ^1-penalty based methods.

5.2.2. *Asymptotic consistency in high dimensions.* We review here a result establishing asymptotic consistency for very high-dimensional but sparse linear models as in (5.2). To capture the notion of high-dimensionality, we equip the model with a dimensionality $p = p_n$ which is allowed to grow with sample size n; moreover, the coefficients $\beta^{(j)} = \beta_n^{(j)}$ are now potentially depending on n and the regression function is denoted by $f_n(\cdot)$.

REPLICA 2. *[17] Consider the linear model in (5.2). Assume that $p_n = O(\exp(n^{1-\xi}))$ for some $0 < \xi \le 1$ (high-dimensionality) and $\sup_{n\in\mathbb{N}}\sum_{j=1}^{p_n}|\beta_n^{(j)}| < \infty$ (sparseness of the true regression function w.r.t. the ℓ^1-norm); moreover, the variables $X_i^{(j)}$ are bounded and $\mathbb{E}[|\varepsilon_i|^{4/\xi}] < \infty$. Then: when stopping the boosting iterations appropriately, i.e. $m = m_n \to \infty$ $(n \to \infty)$ sufficiently slowly, L_2Boosting with componentwise linear least squares satisfies*

$$\mathbb{E}_{X_{new}}[(\hat{f}_n^{[m_n]}(X_{new}) - f_n(X_{new}))^2] \to 0 \textit{ in probability } (n \to \infty),$$

where X_{new} denotes new predictor variables, independent of and with the same distribution as the X-component of the data (X_i, Y_i) $(i = 1, \dots, n)$.

The result holds for almost arbitrary designs and no assumptions about collinearity or correlations are required. Replica 2 identifies boosting as a method which is able to consistently estimate a very high-dimensional but sparse linear model; for the Lasso in (5.3), a similar result holds as well [34]. In terms of empirical performance, there seems to be no overall superiority of L_2Boosting over Lasso or vice-versa.

5.2.3. *Transforming predictor variables.* In view of Replica 2, we may enrich the design matrix in model (5.2) with many transformed predictors: if the true regression function can be represented as a sparse linear combination of original or transformed predictors, consistency is still guaranteed. It should be noted though that the inclusion of non-effective variables in the design matrix does degrade the finite-sample performance.

As a standard option, we propose the use of fractional polynomials [59] for transforming each predictor variable and its log-transform. First, any of the p predictor variables, denoted for simplicity just by X, is transformed into some appropriate interval (for example $[1, 2]$), denoted by $\tilde{X}$. We then consider the following transformed values as new additional predictor variables in the model:

$$\begin{aligned}&\log(\tilde{X}), \log(\tilde{X})^2\\ &\tilde{X}^r, \tilde{X}^r \log(X),\ r \in \{-2, -1, -0.5, 0.5, 1, 2, 3\}.\end{aligned}$$

Thus, each covariate is associated with 16 transformations which additionally enter into the design matrix of the corresponding linear model.

Illustration: Prediction of Total Body Fat (cont.). Such transformation defined in terms of fractional polynomials for estimating a linear model can be used with the `glmboost` function [an implementation of the original fractional polynomials approach is available in package **mfp**, see 1, 58], where the model formula performs the computations of all transformations by means of the `FP` (fractional polynomials) function. First, we scale each covariate to the interval $[1, 2]$ and then fit the complex linear model by using the `glmboost` function with initial $m_{\text{stop}} = 3000$ boosting iterations:

```
R> tbodyfat <- bodyfat
R> tbodyfat[indep] <- lapply(bodyfat[indep], function(x) {
         x <- x - min(x)
         x/max(x) + 1
```

```
     })
R> fpfm <- as.formula(paste("DEXfat ~ ", paste("FP(",
         indep, ")", collapse = "+")))
R> fpfm
```

```
DEXfat ~ FP(age) + FP(waistcirc) + FP(hipcirc) +
    FP(elbowbreadth) + FP(kneebreadth) + FP(anthro3a) +
    FP(anthro3b) + FP(anthro3c) + FP(anthro4)
```

```
R> bf_fp <- glmboost(fpfm, data = tbodyfat,
                     control = boost_control(mstop = 3000))
R> mstop(aic <- AIC(bf_fp))
```

```
[1] 2480
```

The corrected AIC criterion (see Section 5.4) suggests to stop after $m_{\text{stop}} = 2480$ boosting iterations and the final model selects 21 (transformed) predictor variables. Again, the partial contributions of each of the 9 original covariates can be computed easily and are shown (for the same variables as in Figure 4) in Figure 8. Note that the depicted functional relationship derived from the multivariate fractional polynomial model (Figure 8) is qualitatively the same as the one derived from the additive model (Figure 3).

[Fig 8 about here.]

5.3. *Degrees of freedom for L_2Boosting.* A notion of degrees of freedom will be useful for estimating the stopping iteration of boosting (Section 5.4) or for a sparse version of boosting (Section 6.2).

5.3.1. *Componentwise linear least squares.* We consider L_2Boosting with componentwise linear least squares. Denote by

$$\mathcal{H}^{(j)} = \mathbf{X}^{(j)}(\mathbf{X}^{(j)})^\top / \|\mathbf{X}^{(j)}\|^2, \ j = 1, \ldots, p,$$

the $n \times n$ hat matrix for the linear least squares fitting operator using the jth predictor variable $\mathbf{X}^{(j)} = (X_1^{(j)}, \ldots, X_n^{(j)})^\top$ only; $\|x\|^2 = x^\top x$ denotes the Euclidean norm for a vector $x \in \mathbb{R}^n$. The hat matrix of the componentwise linear least squares base procedure (see (4.1)) is then

$$\mathcal{H}^{(\hat{\mathcal{S}})} : \ (U_1, \ldots, U_n) \mapsto \hat{U}_1, \ldots, \hat{U}_n,$$

where $\hat{\mathcal{S}}$ is as in (4.1). Similarly to (5.1), we then obtain the hat matrix of L_2Boosting in iteration m:

$$\begin{aligned} \mathcal{B}_m &= \mathcal{B}_{m-1} + \nu \cdot \mathcal{H}^{(\hat{\mathcal{S}}_m)}(I - \mathcal{B}_{m-1}) \\ &= I - (I - \nu\mathcal{H}^{(\hat{\mathcal{S}}_m)})(I - \nu\mathcal{H}^{(\hat{\mathcal{S}}_{m-1})}) \cdots (I - \nu\mathcal{H}^{(\hat{\mathcal{S}}_1)}), \end{aligned} \tag{5.4}$$

where $\hat{\mathcal{S}}_r \in \{1, \ldots, p\}$ denotes the component which is selected in the componentwise least squares base procedure in the rth boosting iteration. We then define the degrees of freedom of the boosting fit in iteration m as

$$\mathrm{df}(m) = \mathrm{trace}(\mathcal{B}_m).$$

Even with $\nu = 1$, $\mathrm{df}(m)$ is very different from counting the number of variables which have been selected until iteration m. Moreover, the sequential order of the variables entering the model in the process of boosting matters as well.

Having some notion of degrees of freedom at hand, we can estimate the error variance $\sigma_\varepsilon^2 = \mathbb{E}[\varepsilon_i^2]$ in the linear model (5.2) by

$$\hat{\sigma}_\varepsilon^2 = \frac{1}{n - \mathrm{df}(m_{\mathrm{stop}})} \sum_{i=1}^{n} (Y_i - \hat{f}^{[m_{\mathrm{stop}}]}(X_i))^2.$$

We can represent

$$\mathcal{B}_m = \sum_{j=1}^{p} \mathcal{B}_m^{(j)}, \tag{5.5}$$

where $\mathcal{B}_m^{(j)}$ is the hat matrix which yields the fitted values for the jth predictor, i.e. $\mathcal{B}_m^{(j)}\mathbf{Y} = \mathbf{X}^{(j)}\hat{\beta}_j^{[m]}$. Note that the $\mathcal{B}_m^{(j)}$'s can be easily computed in an iterative way by up-dating as follows:

$$\mathcal{B}_m^{(\hat{\mathcal{S}}_m)} = \mathcal{B}_{m-1}^{(\hat{\mathcal{S}}_m)} + \nu \cdot \mathcal{H}^{(\hat{\mathcal{S}}_m)}(I - \mathcal{B}_{m-1}),$$
$$\mathcal{B}_m^{(j)} = \mathcal{B}_{m-1}^{(j)} \text{ for all } j \neq \hat{\mathcal{S}}_m.$$

Thus, we have a decomposition of the total degrees of freedom into p terms:

$$\mathrm{df}(m) = \sum_{j=1}^{p} \mathrm{df}^{(j)}(m),$$
$$\mathrm{df}^{(j)}(m) = \mathrm{trace}(\mathcal{B}_m^{(j)}).$$

The individual degrees of freedom $\mathrm{df}^{(j)}(m)$ are a useful measure to quantify the "complexity" of the coefficient estimate $\hat{\beta}_j^{[m]}$.

5.3.2. *Other base procedures.* Most of the reasonable base procedure do some variable selection. We denote the selected predictor variable index (or indices) by $\hat{\mathcal{S}} \subset \{1, \ldots, p\}$, where $\hat{\mathcal{S}}$ has been estimated from a specified set Γ of subsets of variables. For example, for componentwise linear least squares

(Section 4.1), componentwise splines (Section 4.2) or stumps (Section 4.3), $\hat{\mathcal{S}} \in \Gamma = \{1, \ldots, p\}$ (one variable is selected); for pairwise splines (end of Section 4.2), $\hat{\mathcal{S}} \in \Gamma = \{1, \ldots, p\}^2$ (a pair of variables is selected).

Assuming that the base procedure is linear for a selected set of variables, denote by $\mathcal{H}^{(\mathcal{S})}$ its hat matrices ($\mathcal{S} \in \Gamma$). Denoting by $\hat{\mathcal{S}}_r$ the set of variables which has been selected in the rth round of boosting, all the concepts and formulae from Section 5.3.1 hold (the sum in the decomposition of $\mathrm{df}(m)$ is now over the indices $j \in \Gamma$).

For example, with componentwise smoothing splines: the hat matrices $\mathcal{H}^{(j)}$ ($j = 1, \ldots p$) are known explicitly [33], and $\mathrm{df}^{(j)}(m)$ denotes then the degrees of freedom for the boosting function estimate of the jth additive term.

5.4. *Internal stopping criteria for L_2Boosting.* Having some degrees of freedom at hand, we can now use internal information criteria for estimating a good stopping iteration, without pursuing some sort of cross-validation.

We can use the corrected AIC [40]:

$$\mathrm{AIC}_c(m) = \log(\hat{\sigma}^2) + \frac{1 + \mathrm{df}(m)/n}{1 - \mathrm{df}(m) + 2)/n},$$
$$\hat{\sigma}^2 = n^{-1} \sum_{i=1}^{n} (Y_i - (\mathcal{B}_m \mathbf{Y})_i)^2.$$

In **mboost**, the corrected AIC criterion can be computed via `AIC(x, method = "corrected")` (with `x` being an object returned by `glmboost` or `gamboost` called with `family = GaussReg()`). Alternatively, we may employ the gMDL criterion Hansen and Yu [35]:

$$\mathrm{gMDL}(m) = \log(S) + \frac{\mathrm{df}(m)}{n} \log(F),$$
$$S = \frac{n\hat{\sigma}^2}{n - \mathrm{df}(m)}, \quad F = \frac{\sum_{i=1}^{n} Y_i^2 - n\hat{\sigma}^2}{\mathrm{df}(m) S}.$$

The gMDL criterion bridges the AIC and BIC in a data-driven way: it is an attempt to adaptively select the better among the two.

When using L_2Boosting for binary classification (see also the end of Section 3.2 and end of Section 5.2), we prefer to work with the binomial log-likelihood in AIC,

$$\mathrm{AIC}(m) = -2 \sum_{i=1}^{n} Y_i \log((\mathcal{B}_m \mathbf{Y})_i) + (1 - Y_i) \log(1 - (\mathcal{B}_m \mathbf{Y})_i) + 2\mathrm{df}(m),$$

or for $\mathrm{BIC}(m)$ with the penalty term $\log(n)\mathrm{df}(m)$. (If $(\mathcal{B}_m \mathbf{Y})_i \notin [0, 1]$, we truncate by $\max(\min((\mathcal{B}_m \mathbf{Y})_i, 1 - \delta), \delta)$ for some small $\delta > 0$. e.g. $\delta = 10^{-5}$).

6. Boosting for variable selection.

We address here the question whether boosting is a good variable selection scheme. For problems with many predictor variables, boosting is computationally much more efficient than classical all subset selection schemes. The mathematical properties of boosting for variable selection are still open questions, e.g. whether it leads to a consistent model selection method.

6.1. *L_2Boosting.*

When borrowing from the analogy of L_2Boosting with the Lasso (see Section 5.2.1), the following holds. Consider a linear model as in (5.2), assuming Gaussian errors and predictor variables and allowing for $p \gg n$ but being sparse. Then, there is a necessary and almost sufficient condition such that for some suitable penalty parameter λ in (5.3), the Lasso finds the true underlying sub-model (the predictor variables with corresponding regression coefficients $\neq 0$) with probability tending quickly to 1 as $n \to \infty$ [50]. It is important to note the role of the sufficient and necessary condition of the Lasso for model selection: Zhao and Yu [78] call it the "irrepresentable condition" which has (mainly) implications on the design (predictor variables), and they give examples where it holds and where it fails to be true. A further complication is the fact that when tuning the Lasso for prediction optimality, i.e., choosing the penalty parameter λ in (5.3) such that the MSE is minimal, the probability for estimating the true sub-model converges to a number which is less than one or even zero if the problem is high-dimensional [50]. In fact, the prediction optimal tuned Lasso selects asymptotically too large models. However, the Lasso seems to be a very useful method for variable filtering: for many cases, the prediction optimal tuned Lasso selects a sub-model which contains the true model with high probability.

We do not expect that boosting is free from the difficulties which occur when using the Lasso for variable selection. The hope is though, that also boosting would produce an interesting set of sub-models when varying the number of iterations. As indicated above, we often would like to construct estimators which are less biased than the Lasso. It is instructive to look at regression with orthonormal design, i.e., the model (5.2) with $\sum_{i=1}^{n} X_i^{(j)} X_i^{(k)} = \delta_{jk}$. Then, the Lasso and also L_2Boosting with componentwise linear least squares and using very small ν (in step 4 of L_2Boosting, see Section 3.3.1) yield the soft-threshold estimator [20, 25], see Figure 9. Often, we would like to have an estimator which exhibits small bias if the observation is very large. The hard-threshold estimator but also the non-negative garrote estimator [12] have this property, see also Figure 9.

[Fig 9 about here.]

For linear models with non-orthogonal design, the former corresponds to all subset selection which is often computationally infeasible and in addition, it may exhibit a too large variance ("unstable"). The latter, i.e., the non-negative garrote estimator, is a nice proposal between hard- and soft-thresholding, with almost no bias for large observed values, being computationally tractable also for non-orthogonal design matrices with $p \leq n$ (in practice, p should be substantially smaller than n) and having reasonable variance (i.e. fairly "stable").

6.2. *SparseL_2Boosting.* A simple modification of L_2Boosting yields solutions in the spirit of Breiman's non-negative garrote estimator. In every iteration of L_2Boosting, we select the variable or component $\mathcal{S} \in \Gamma$ (e.g. $\Gamma = \{1, \ldots, p\}$ for componentwise linear least squares or componentwise smoothing splines) such that the residual sum of squares is minimized. For SparseL_2Boosting, the proposal is to use a final prediction error criterion [20]:

$$\begin{aligned} \hat{\mathcal{S}} = \operatorname*{argmin}_{\mathcal{S} \in \Gamma} \quad & n^{-1} \sum_{i=1}^{n} (U_i - (\mathcal{H}^{(\mathcal{S})}\mathbf{U})_i)^2 \\ & + \quad \gamma \cdot \mathrm{trace}(\mathcal{B}_{m-1} + \mathcal{H}^{(\mathcal{S})}(I - \mathcal{B}_{m-1})), \end{aligned} \tag{6.1}$$

where $\mathbf{U} = (U_1, \ldots, U_n)^\top = (I - \mathcal{B}_{m-1})\mathbf{Y}$ denotes the vector of current residuals (in iteration $m-1$). Having the estimate $\hat{\mathcal{S}}$, we proceed exactly as for L_2Boosting in Section 3.3.1. We stop the iterations with

$$m_{\mathrm{stop}} = \operatorname*{argmin}_{m} \; n^{-1} \sum_{i=1}^{n} (U_i - (\mathcal{B}_m \mathbf{Y})_i)^2 + \gamma \cdot \mathrm{trace}(\mathcal{B}_m).$$

The procedure is called SparseL_2Boosting: more explanation why it yields sparser solutions is given below.

REPLICA 3. *[20] In a linear model with orthonormal predictor variables, SparseL_2Boosting with componentwise linear least squares and using very small ν yields Breiman's non-negative garrote estimator: the parameter γ in (6.1) corresponds to a threshold in the non-negative garrote estimator .*

Replica 3 gives an explanation why SparseL_2Boosting yields sparse models in terms of the number of selected variables. Suppose that the true model is very sparse with say one effective predictor variable having a large regression coefficient and many $p-1$ non-effective predictors. Ideally, we would use the effective predictor variable only and employ very little shrinkage (because of the large corresponding regression coefficient). This can be approximately achieved with the non-negative garrote, choosing a relatively large threshold such that only the effective variable is chosen; the shrinkage effect is still small because the regression coefficient is large. On the other hand, with soft-thresholding, we can also choose a large threshold, resulting in a fit which uses the effective predictor only but employs strong shrinkage; or we choose a smaller threshold which employs less shrinkage but allows for more variables in the model. Typically, the latter has lower mean squared error and hence, the soft-threshold estimator is less sparse than the non-negative garrote (in terms of the number of selected variables). This examples generalizes to the case, where we have a few effective predictors, all of them having substantial regression coefficients, and many non-effective variables.

The final prediction error criterion function in (6.1) requires the choice of a parameter γ. In principle, we could tune this parameter using some cross-validation scheme. As an alternative, we propose the AIC$_c$ or gMDL criterion from Section 5.4. For gMDL, we replace (6.1) by

$$\hat{\mathcal{S}} = \operatorname*{argmin}_{\mathcal{S}\in\Gamma} \left(\log(T(\mathcal{S})) + \frac{k(\mathcal{S})}{n} \log(F(\mathcal{S})) \right),$$

$$T(\mathcal{S}) = \frac{\mathrm{RSS}(\mathcal{S})}{n - k(\mathcal{S})},\ \ F(\mathcal{S}) = \frac{\sum_{i=1}^{n} Y_i^2 - \mathrm{RSS}(\mathcal{S})}{k(\mathcal{S})T(\mathcal{S})},$$

$$\mathrm{RSS}(\mathcal{S}) = \sum_{i=1}^{n}(U_i - (\mathcal{H}^{(\mathcal{S})}\mathbf{U})_i)^2,\ \ k(\mathcal{S}) = \mathrm{trace}(\mathcal{B}_{m-1} + \mathcal{H}^{(\mathcal{S})}(I - \mathcal{B}_{m-1})).$$

Moreover, stooping can be done by using AIC$_c$ or gMDL as in Section 5.4.

7. Boosting for exponential family models.

7.1. *BinomialBoosting.* For binary classification with $Y \in \{0,1\}$, BinomialBoosting uses the negative binomial log-likelihood from (3.1) as loss function. The algorithm is described in Section 3.3.2. Since the population minimizer is $f^*(x) = \log[p(x)/(1-p(x))]/2$, estimates from BinomialBoosting are on half of the logit-scale: the componentwise linear least squares base procedure yields a logistic linear model fit while using componentwise smoothing splines fits a logistic additive model. Many of the concepts and facts from Section 5 about L_2Boosting become useful heuristics for BinomialBoosting.

One principal difference is the derivation of the boosting hat matrix. Instead of (5.4), a linearization argument leads to the following recursion (assuming $\hat{f}^{[0]} \equiv 0$) for an *approximate* hat matrix $\mathcal{B}_m$:

$$\begin{aligned}
&\mathcal{B}_1 = \nu 4 W^{[0]} \mathcal{H}^{\hat{\mathcal{S}}_1},\\
&\mathcal{B}_m = \mathcal{B}_{m-1} + 4\nu W^{[m-1]} \mathcal{H}^{\hat{\mathcal{S}}_m} (I - \mathcal{B}_{m-1}) \quad (m \geq 2),\\
&W^{[m]} = \mathrm{diag}(\hat{p}^{[m]}(X_i)(1 - \hat{p}^{[m]}(X_i);\ 1 \leq i \leq n).
\end{aligned} \tag{7.1}$$

A derivation is given in the appendix. Degrees of freedom are then defined as in Section 5.3,

$$\mathrm{df}(m) = \mathrm{trace}(\mathcal{B}_m),$$

and they can be used for information criteria, e.g.

$$\mathrm{AIC}(m) = -2 \sum_{i=1}^{n} [Y_i \log(\hat{p}^{[m]}(X_i)) + (1 - Y_i) \log(1 - \hat{p}^{[m]}(X_i))] + 2\mathrm{df}(m),$$

or for $\mathrm{BIC}(m)$ with the penalty term $\log(n)\mathrm{df}(m)$. In **mboost**, this AIC criterion can be computed via `AIC(x, method = "classical")` (with `x` being an object returned by `glmboost` or `gamboost` called with `family = Binomial()`).

Illustration: Wisconsin Prognostic Breast Cancer. Prediction models for recurrence events in breast cancer patients based on covariates which have been computed from a digitized image of a fine needle aspirate of breast tissue (those measurements describe characteristics of the cell nuclei present in the image) have been studied by Street et al. [64] [the data is part of the UCI repository 10].

We first analyse this data as a binary prediction problem (recurrence vs. non-recurrence) and later in Section 8 by means of survival models. Again, we are faced with lots of potential covariates ($p = 32$) for a limited number of observations without missing values ($n = 194$) and variable selection is an issue. We can choose a classical logistic regression model via AIC in a stepwise algorithm as follows (after centering the covariates)

```
R> wpbc2 <- wpbc[complete.cases(wpbc), colnames(wpbc) != "time"]
R> indep <- names(wpbc2)[names(wpbc2) != "status"]
R> wpbc2[indep] <- lapply(wpbc2[indep],
       function(x) x - mean(x))
R> wpbc_step <- step(glm(status ~ ., data = wpbc2,
         family = binomial()), trace = 0)
```

The final model consists of 16 parameters with

```
R> logLik(wpbc_step)
```

```
'log Lik.' -80.13 (df=16)
```

```
R> AIC(wpbc_step)
```

```
[1] 192.26
```

and we want to compare this model to a logistic regression model fitted via gradient boosting. We simply select the `Binomial` family (with default offset of $1/2 \log(p/(1-p))$, where p is the proportion of recurrences) and start with initial $m_{\text{stop}} = 500$ boosting iterations

```
R> wpbc_glm <- glmboost(status ~ ., data = wpbc2,
         family = Binomial(),
         control = boost_control(mstop = 500))
```

The negative binomial log-likelihood is

```
R> logLik(wpbc_glm)
```

```
[1] -89.964
```

and the classical AIC criterion suggests to stop after

```
R> aic <- AIC(wpbc_glm, "classical")
R> aic
```

```
[1] 198.44
Optimal number of boosting iterations: 260
Degrees of freedom (for mstop = 260): 7.032
```

boosting iterations. We now restrict the number of boosting iterations to $m_{\text{stop}} = 260$ via

```
R> wpbc_glm <- wpbc_glm[mstop(aic)]
R> logLik(wpbc_glm)
```

```
[1] -92.189
```

```
R> coef(wpbc_glm)[abs(coef(wpbc_glm)) > 0]
```

```
     (Intercept)      mean_texture     mean_symmetry
     -3.0110e-02       -2.4215e-02       -3.3878e+00
 mean_fractaldim        SE_texture      SE_perimeter
     -2.0321e+01       -2.6603e-02        4.0908e-02
   SE_compactness      SE_concavity SE_concavepoints
      7.0280e+00       -4.6303e+00       -1.5737e+01
     SE_symmetry      worst_radius   worst_perimeter
      2.8601e+00        1.7777e-02        1.2639e-03
      worst_area worst_smoothness             tsize
      1.5854e-04        8.8372e+00        3.1014e-02
          pnodes
      2.5981e-02
```

(because of using the offset-value $\hat{f}^{[0]}$, we have to add the value $\hat{f}^{[0]}$ to the reported intercept estimate above for the logistic regression model) and we then can extract the fitted conditional probabilities

```
R> f <- fitted(wpbc_glm)
R> p <- exp(f)/(exp(f) + exp(-f))
```

which are depicted by a conditional density plot in Figure 10.

[Fig 10 about here.]

A generalized additive model adds more flexibility to the regression function but is still interpretable. We fit a logistic additive model to the `wpbc` data as follows:

```
R> wpbc_gam <- gamboost(status ~ ., data = wpbc2,
         family = Binomial())
R> mopt <- mstop(aic <- AIC(wpbc_gam, "classical"))
R> aic
```

```
[1] 196.33
Optimal number of boosting iterations: 84
Degrees of freedom (for mstop = 84): 13.754
```

This model selected 16 out of 32 covariates. The partial contributions of the four most important variables are depicted in Figure 11 indicating a remarkable degree of non-linearity.

[Fig 11 about here.]

7.2. *PoissonBoosting.* For count data with $Y \in \{0, 1, 2, \ldots\}$, we can use Poisson regression: we assume that $Y|X = x$ has a Poisson$(\lambda(x))$ distribution and the goal is to estimate the function $f(x) = \log(\lambda(x))$. Then, the negative log-likelihood yields the loss function

$$\rho(y, f) = -yf + \exp(f), \quad f = \log(\lambda),$$

which can be used in the functional gradient descent algorithm in Section 2.1 and is implemented as

```
R> Poisson()
```

```
         Poisson

Loss function: -y * f + exp(f)
```

Similarly to (7.1), the *approximate* boosting hat matrix is computed by the following recursion

$$\begin{aligned}
&\mathcal{B}_1 = \nu W^{[0]}\mathcal{H}^{\hat{\mathcal{S}}_1},\\
&\mathcal{B}_m = \mathcal{B}_{m-1} + \nu W^{[m-1]}\mathcal{H}^{\hat{\mathcal{S}}_m}(I - \mathcal{B}_{m-1}) \quad (m \geq 2),\\
&W^{[m]} = \mathrm{diag}(\hat{\lambda}^{[m]}(X_i);\ 1 \leq i \leq n).
\end{aligned} \tag{7.2}$$

8. Survival analysis.

For survival analysis with censored data it is possible, under some circumstances, to construct a boosting algorithm based on weighted least squares fitting of a base procedure. The approach is described in detail in Hothorn et al. [37].

We assume complete data of the following form: survival times $T_i \in \mathbb{R}^+$ (some of them right-censored) and predictors $X_i \in \mathbb{R}^p$, $i = 1, \ldots, n$. We transform the survival times to the log-scale, but this step is not crucial for what follows: $Y_i = \log(T_i)$. What we observe is

$$O_i = (\tilde{Y}_i, X_i, \Delta_i),\ \tilde{Y}_i = \log(\tilde{T}_i),\ \tilde{T}_i = \min(T_i, C_i),$$

where $\Delta_i = I(T_i \leq C_i)$ is a censoring indicator and C_i the censoring time. Here, we make a restrictive assumption that C_i is conditionally independent of T_i given X_i (and we assume independence among different indices i): this implies that the coarsening at random assumption holds [73].

We consider the squared error loss for the complete data, $\rho(y, f) = (y - f)^2$ (without the irrelevant factor 1/2). For the observed data, the following weighted version turns out to be useful:

$$\begin{aligned}
&\rho_{\mathrm{obs}}(o, f) = (\tilde{y} - f)^2 \Delta \frac{1}{G(\tilde{t}|x)},\\
&G(c|x) = \mathbb{P}[C > c | X = x].
\end{aligned}$$

Thus, the observed data loss function is weighted by the inverse probability for censoring $\Delta G(\tilde{t}|x)^{-1}$ (the inverse probability of censoring weights, IPC). Under the coarsening at random assumption, it then holds that

$$\mathbb{E}_{Y,X}[(Y - f(X))^2] = \mathbb{E}_O[\rho_{\mathrm{obs}}(O, f(X))],$$

see van der Laan and Robins [73].

The strategy is then to estimate $G(\cdot|x)$, e.g. by the Kaplan-Meier estimator, and do weighted L_2Boosting using the weighted squared error loss:

$$\sum_{i=1}^{n} \Delta_i \frac{1}{\hat{G}(\tilde{T}_i|X_i)}(\tilde{Y}_i - f(X_i))^2,$$

where the weights are of the form $\Delta_i \hat{G}(\tilde{T}_i|X_i)^{-1}$. As demonstrated in the previous sections, we can use various base procedures as long as they allow for weighted least squares fitting. Furthermore, the concepts of degrees of freedom and information criteria are analogous to Sections 5.3 and 5.4. Details are given in [37].

Illustration: Wisconsin Prognostic Breast Cancer (cont.). Instead of the binary response variable describing the recurrence status we make use of the additionally available time information for modeling the time to recurrence, i.e., all observations with non-recurrence are censored. First, we calculate IPC weights and center the covariates

```
R> iw <- IPCweights(Surv(wpbc$time, wpbc$status == "R"))
R> wpbc3 <- wpbc[, colnames(wpbc) != "status"]
R> indep <- names(wpbc3)[names(wpbc3) != "time"]
R> wpbc3[indep] <- lapply(wpbc3[indep],
       function(x) x - mean(x, na.rm = TRUE))
```

and fit a weighted linear model by boosting with componentwise linear weighted least squares as base procedure:

```
R> wpbc_surv <- glmboost(log(time) ~ ., data = wpbc3,
         control = boost_control(mstop = 500), weights = iw)
R> mstop(aic <- AIC(wpbc_surv))
```

```
[1] 122
```

```
R> wpbc_surv <- wpbc_surv[mstop(aic)]
```

The following variables have been selected for fitting

```
R> names(coef(wpbc_surv)[abs(coef(wpbc_surv)) > 0])
```

```
 [1] "mean_radius"          "mean_texture"
 [3] "mean_perimeter"       "mean_smoothness"
 [5] "mean_symmetry"        "SE_texture"
 [7] "SE_smoothness"        "SE_concavepoints"
 [9] "SE_symmetry"          "worst_concavepoints"
```

and the fitted values are depicted in Figure 12, showing a reasonable model fit.

[Fig 12 about here.]

9. Other works.

We briefly summarize here some other works which have not been mentioned in the earlier sections. A very different exposition than ours is the overview of boosting by Meir and Rätsch [51].

9.1. *Methodology and applications .* Boosting methodology has been used for various other statistical models than what we have discussed in the previous sections. Models for multivariate responses are studied in [18, 46]; some multi-class boosting methods are discussed in [30],[79]. Other works deal with boosting approaches for generalized linear and nonparametric models [69],[71],[68], for flexible semiparametric mixed models [72] or for nonparametric models with quality constraints [70],[44].

There are numerous applications of boosting methods to real data problems. We mention here classification of tumor types from gene expressions ([23],[22]), multivariate financial time series ([5], [3],[4]), text classification [62], document routing [41] or survival analysis [8] (different from the approach in Section 8).

9.2. *Asymptotic theory.* The asymptotic analysis of boosting algorithms include consistency and minimax rate results. The first consistency result for AdaBoost has been given by Jiang [42]. Later, Zhang and Yu [76] refined the results for a functional gradient descent with an additional relaxation scheme, and their theory covers also more general loss functions than the exponential loss in AdaBoost. For L_2Boosting, the first minimax rate result has been established by Bühlmann and Yu [19]. This has then be generalized to much more general settings by Yao et al. [75] and Bissantz et al. [9].

In the machine learning community, there has been a substantial focus on estimation in the convex hull of function classes (cf. [6], [45],[7]). For example, one may want to estimate a regression or probability function by using

$$\sum_{k=1}^{\infty} \hat{w}_k \hat{g}^{[k]}(\cdot),\ \hat{w}_k \geq 0,\ \sum_{k=1}^{\infty} \hat{w}_k = 1,$$

where the $\hat{g}^{[k]}(\cdot)$'s belong to a function class such as stumps or trees with a fixed number of terminal nodes. The estimator above is a convex combination of individual functions, in contrast to boosting which pursues a linear combination of individual functions. By scaling, which is necessary in practice and theory (cf. [45]), one can actually look at this as a linear combination of functions whose coefficients satisfy $\sum_k w_k = \lambda$. This then represents an ℓ^1-constraint as in Lasso, a relation which we have already seen from another perspective in section 5.2.1. Consistency of such convex combination or ℓ^1-regularized "boosting" methods have been given by Lugosi and Vayatis [45]. Mannor et al. [48] and Blanchard et al. [11] derived results for rates of convergence of (versions of) convex combination schemes.

APPENDIX A: APPENDIX SECTION

Derivation of formula (7.1). The negative gradient is

$$-\frac{\partial}{\partial f}\rho(y,f) = 2(y-p), \quad p = \frac{\exp(f)}{\exp(f)+\exp(-f)}.$$

Next, we linearize $\hat{p}^{[m]}$: we denote by $\hat{p}^{[m]} = (\hat{p}^{[m]}(X_1),\dots,\hat{p}^{[m]}(X_n))^\top$ and analogously for $\hat{f}^{[m]}$. Then,

$$\begin{aligned}\hat{p}^{[m]} &\approx \hat{p}^{[m-1]} + \frac{\partial p}{\partial f}|_{f=\hat{f}^{m-1}}(\hat{f}^{[m]} - \hat{f}^{[m-1]}\\ &= \hat{p}^{[m-1]} + 2W^{[m-1]}\nu\mathcal{H}^{\hat{\mathcal{S}}_m}2(\mathbf{Y} - \hat{p}^{[m-1]}),\end{aligned} \tag{A.1}$$

where $W^{[m]} = \mathrm{diag}(\hat{p}(X_i)(1-\hat{p}(X_i)); 1 \le i \le n)$. Since for the hat matrix, $\mathcal{B}_m\mathbf{Y} = \hat{p}^{[m]}$, we obtain from (A.1)

$$\begin{aligned}&\mathcal{B}_1 \approx \nu 4W^{[0]}\mathcal{H}^{\hat{\mathcal{S}}_1},\\ &\mathcal{B}_m \approx \mathcal{B}_{m-1} + \nu 4W^{[m-1]}\mathcal{H}^{\hat{\mathcal{S}}_m}(I - \mathcal{B}_{m-1}) \quad (m \ge 2),\end{aligned}$$

which shows that (7.1) is approximately true. □

Derivation of formula (7.2). The arguments are analogous as above for the binomial case. Here, the negative gradient is

$$-\frac{\partial}{\partial f}\rho(y,f) = y - \lambda, \quad \lambda = \exp(f).$$

When linearinzing $\hat{\lambda}^{[m]} = (\hat{\lambda}^{[m]}(X_1),\dots,\hat{\lambda}^{[m]}(X_n))^\top$ we get, analogously to (A.1),

$$\begin{aligned}\hat{\lambda}^{[m]} &\approx \hat{\lambda}^{[m-1]} + \frac{\partial \lambda}{\partial f}|_{f=\hat{f}^{m-1}}(\hat{f}^{[m]} - \hat{f}^{[m-1]}\\ &= \hat{\lambda}^{[m-1]} + W^{[m-1]}\nu\mathcal{H}^{\hat{\mathcal{S}}_m}2(\mathbf{Y} - \hat{\lambda}^{[m-1]}),\end{aligned}$$

where $W^{[m]} = \mathrm{diag}(\hat{\lambda}(X_i)(1-\hat{p}(X_i)); 1 \le i \le n)$. We then complete the derivation of (7.2) as in the binomial case above. □

ACKNOWLEDGEMENTS

We would like to thank Axel Benner, Roman Lutz and Lukas Meier for discussions and detailed comments. The work of T. Hothorn was supported by Deutsche Forschungsgemeinschaft (DFG) under grant HO 3242/1-1.

REFERENCES

[1] Ambler, G. and Benner, A. (2005). *mfp: Multivariable Fractional Polynomials*. R package version 1.3.2.
URL http://CRAN.R-project.org

[2] Amit, Y. and Geman, D. (1997). Shape quantization and recognition with randomized trees. *Neural Computation* **9** 1545–1588.

[3] Audrino, F. and Barone-Adesi, G. (2005). Functional gradient descent for financial time series with an application to the measurement of market risk. *Journal of Banking and Finance* **29** 959–977.

[4] Audrino, F. and Barone-Adesi, G. (2005). A multivariate FGD technique to improve VaR computation in equity markets. *Computational Management Science* **2** 87–106.

[5] Audrino, F. and Bühlmann, P. (2003). Volatility estimation with functional gradient descent for very high-dimensional financial time series. *Journal of Computational Finance* **6** 65–89.

[6] Bartlett, P. (2003). Prediction algorithms: complexity, concentration and convexity. In *Proceedings of the 13th IFAC Symposium on System Identification*.

[7] Bartlett, P., M.I. Jordan, M. and McAuliffe, J. (2006). Convexity, classification, and risk bounds. *Journal of the American Statistical Association* **101** 138–156.

[8] Benner, A. (2002). Application of "aggregated classifiers" in survival time studies. In *Proceedings in Computational Statistics (COMPSTAT)* (W. H. W. and B. Rönz, eds.). Physica-Verlag, Heidelberg.

[9] Bissantz, N., Hohage, T., Munk, A. and Ruymgaart, F. (2005). Convergence rates of general regularization methods for statistical inverse problems and applications. Tech. rep., Universität Göttingen.
URL http://www.stochastik.math.uni-goettingen.de/preprints/bissantz_hohage_munk_ruymgaart.pdf

[10] Blake, C. L. and Merz, C. J. (1998). UCI repository of machine learning databases.
URL http://www.ics.uci.edu/~mlearn/MLRepository.html

[11] Blanchard, G., Lugosi, G. and Vayatis, N. (2003). On the rate of convergence of regularized boosting classifiers. *Journal of Machine Learning Research* **4** 861–894.

[12] Breiman, L. (1995). Better subset regression using the nonnegative garrote. *Technometrics* **37** 373–384.

[13] Breiman, L. (1996). Bagging predictors. *Machine Learning* **24** 123–140.

[14] Breiman, L. (1998). Arcing classifiers (with discussion). *The Annals of Statistics* **26** 801–849.

[15] Breiman, L. (1999). Prediction games & arcing algorithms. *Neural Computation* **11** 1493–1517.

[16] Breiman, L. (2001). Random forests. *Machine Learning* **45** 5–32.

[17] Bühlmann, P. (2006). Boosting for high-dimensional linear models. *The Annals of Statistics* **34** 559–583.

[18] Bühlmann, P. and Lutz, R. (2006). Boosting algorithms: with an application to bootstrapping multivariate time series. In *The Frontiers in Statistics* (J. Fan and H. Koul, eds.). Imperial College Press.

[19] Bühlmann, P. and Yu, B. (2003). Boosting with the L_2 loss: Regression and classification. *Journal of the American Statistical Association* **98** 324–339.

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

[21] Buja, A., Stuetzle, W. and Shen, Y. (2005). Loss functions for binary class prob-

ability estimation: structure and applications. Tech. rep., University of Washington. URL http://www.stat.washington.edu/wxs/Learning-papers/paper-proper-scoring.pdf

[22] Dettling, M. (2004). Bagboosting for tumor classification with gene expression data. *Bioinformatics* **20** 3583–3593.

[23] Dettling, M. and Bühlmann, P. (2003). Boosting for tumor classification with gene expression data. *Bioinformatics* **19** 1061–1069.

[24] DiMarzio, M. and Taylor, C. (2005). Multistep kernel regression smoothing by boosting. Tech. rep., University of Leeds.

[25] Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. (2004). Least angle regression (with discussion). *The Annals of Statistics* **32** 407–451.

[26] Freund, Y. and Schapire, R. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*. Lecture Notes in Computer Science, Springer.

[27] Freund, Y. and Schapire, R. (1996). Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA.

[28] Freund, Y. and Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* **55** 119–139.

[29] Friedman, J. (2001). Greedy function approximation: a gradient boosting machine. *The Annals of Statistics* **29** 1189–1232.

[30] Friedman, J., Hastie, T. and Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting (with discussion). *The Annals of Statistics* **28** 337–407.

[31] Garcia, A. L., Wagner, K., Hothorn, T., Koebnick, C., Zunft, H. J. and Trippo, U. (2005). Improved prediction of body fat by measuring skinfold thickness, circumferences, and bone breadths. *Obesity Research* **13** 626–634.

[32] Gentleman, R. C., Carey, V. J., Bates, D. M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J., Hornik, K., Hothorn, T., Huber, M., Iacus, S., Irizarry, R., Leisch, F., Li, C., Maechler, M., Rossini, A. J., Sawitzki, G., Smith, C., Smyth, G., Tierney, L., Yang, J. Y. and Zhang, J. (2004). Bioconductor: open software development for computational biology and bioinformatics. *Genome Biology* **5** R80.

[33] Green, P. and Silverman, B. (1994). *Nonparametric Regression and Generalized Linear Models: A Roughness Penalty Approach*. Chapman & Hall, New York.

[34] Greenshtein, E. and Ritov, Y. (2004). Persistence in high-dimensional predictor selection and the virtue of over-parametrization. *Bernoulli* **10** 971–988.

[35] Hansen, M. and Yu, B. (2001). Model selection and minimum description length principle. *Journal of the American Statistical Association* **96** 746–774.

[36] Hothorn, T. and Bühlmann, P. (2006). *mboost: Gradient Boosting for Fitting Generalized Linear, Additive and Interaction Models*. R package version 0.4-8. URL http://CRAN.R-project.org/

[37] Hothorn, T., Bühlmann, P., Dudoit, S., Molinaro, A. and van der Laan, M. (2006). Survival ensembles. *Biostatistics* **7** 355–373.

[38] Hothorn, T., Hornik, K. and Zeileis, A. (2006). *party: A Laboratory for Recursive Part(y)itioning*. R package version 0.8-6, http://CRAN.R-project.org/.

[39] Hothorn, T., Hornik, K. and Zeileis, A. (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics* In press.

[40] Hurvich, C., Simonoff, J. and Tsai, C.-L. (1998). Smoothing parameter selection in nonparametric regression using an improved Akaike information criterion. *Journal of the Royal Statististical Society, Series B* **60** 271–293.

[41] Iyer, R., Lewis, D., Schapire, R., Singer, Y. and Singhal, A. (2000). Boosting for document routing. In *Proceedings of CIKM-00, 9th ACM Int. Conf. on Information and Knowledge Management* (A. Agah, J. Callan and E. Rundensteiner, eds.). ACM Press.

[42] Jiang, W. (2004). Process consistency for AdaBoost (with discussion). *The Annals of Statistics* **32** 13–29 (disc. pp. 85–134).

[43] Kearns, M. and Valiant, L. (1994). Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the Association for Computing Machinery* **41** 67–95.

[44] Leitenstorfer, F. and Tutz, G. (2006). Smoothing with curvature constraints based on boosting techniques. Tech. rep., SFB 386, Munich.
URL http://www.statistik.lmu.de/sfb386/papers/dsp/paper467.pdf

[45] Lugosi, G. and Vayatis, N. (2004). On the Bayes-risk consistency of regularized boosting methods (with discussion). *The Annals of Statistics* **32** 30–55 (disc. pp. 85–134).

[46] Lutz, R. and Bühlmann, P. (2006). Boosting for high-multivariate responses in high-dimensional linear regression. *Statistica Sinica* **16** 471–494.

[47] Mallat, S. and Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing* **41** 3397–3415.

[48] Mannor, S., Meir, R. and Zhang, T. (2003). Greedy algorithms for classification–consistency, convergence rates, and adaptivity. *Journal of Machine Learning Research* **4** 713–741.

[49] Mason, L., Baxter, J., Bartlett, P. and Frean, M. (2000). Functional gradient techniques for combining hypotheses. In *Advances in Large Margin Classifiers* (A. A. Smola, P. Bartlett, B. B. Schölkopf and D. Schuurmans, eds.). MIT Press, Cambridge.

[50] Meinshausen, N. and Bühlmann, P. (2006). High-dimensional graphs and variable selection with the lasso. *The Annals of Statistics* **34**. In press.

[51] Meir, R. and Rätsch, G. (2003). An introduction to boosting and leveraging. In *Advanced Lectures on Machine Learning* (S. Mendelson and A. Smola, eds.). Lecture Notes in Computer Science, Springer.

[52] Osborne, M., Presnell, B. and Turlach, B. (2000). A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis* **20** 389–403.

[53] R Development Core Team (2006). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.
URL http://www.R-project.org

[54] Rätsch, G., Onoda, T. and Müller, K. (2001). Soft margins for AdaBoost. *Machine Learning* **42** 287–320.

[55] Ridgeway, G. (1999). The state of boosting. *Computing Science and Statistics* **31** 172–181.

[56] Ridgeway, G. (2002). Looking for lumps: Boosting and bagging for density estimation. *Computational Statistics & Data Analysis* **38** 379–392.

[57] Ridgeway, G. (2006). *gbm: Generalized Boosted Regression Models.* R package version 1.5-7.
URL http://www.i-pensieri.com/gregr/gbm.shtml

[58] Sauerbrei, W., Meier-Hirmer, C., Benner, A. and Royston, P. (2006). Mul-

tivariable regression model building by using fractional polynomials: Description of SAS, STATA and R programs. *Computational Statistics & Data Analysis* In press.

[59] Sauerbrei, W. and Royston, P. (1999). Building multivariable prognostic and diagnostic models: Transformation of the predictors by using fractional polynomials. *Journal of the Royal Statistical Society, Series A* **162** 71–94.

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

[61] Schapire, R. (2002). The boosting approach to machine learning: an overview. In *MSRI Workshop on Nonlinear Estimation and Classification* (D. Denison, M. Hansen, C. Holmes, B. Mallick and B. Yu, eds.). Springer.

[62] Schapire, R. and Singer, Y. (2000). Boostexter: a boosting-based system for text categorization. *Machine Learning* **39** 135–168.

[63] Southwell, R. (1946). *Relaxation Methods in Theoretical Physics*. Oxford Univeristy Press.

[64] Street, W. N., Mangasarian, O. L., and Wolberg, W. H. (1995). An inductive learning approach to prognostic prediction. In *Proceedings of the Twelfth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA.

[65] Temlyakov, V. (2000). Weak greedy algorithms. *Advances in Computational Mathematics* **12** 213–227.

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

[67] Tukey, J. (1977). *Exploratory Data Analysis*. Addison-Wesley, Reading, MA.

[68] Tutz, G. and Binder, H. (2005). Boosting Ridge regression. Tech. rep., SFB 386, LMU München.
URL http://www.stat.uni-muenchen.de/sfb386/papers/dsp/paper418.pdf

[69] Tutz, G. and Hechenbichler, K. (2005). Aggregating classifiers with ordinal response structure. *Journal Statistical Computation and Simulation* **75** 391–408.

[70] Tutz, G. and Leitenstorfer, F. (2005). Generalized smooth monotonic regression. Tech. rep., SFB 386, LMU München.
URL http://www.statistik.lmu.de/sfb386/papers/dsp/paper417.pdf

[71] Tutz, G. and Leitenstorfer, F. (2006). Response shrinkage estimators in binary regression. *Computational Statistics & Data Analysis* In press.

[72] Tutz, G. and Reithinger, F. (2005). Flexible semiparametric mixed models. Tech. rep., SFB 386, LMU München.
URL http://www.stat.uni-muenchen.de/sfb386/papers/dsp/paper448.pdf

[73] van der Laan, M. and Robins, J. (2003). *Unified Methods for Censored Longitudinal Data and Causality*. Springer.

[74] West, M., Blanchette, C., Dressman, H., Huang, E., Ishida, S., Spang, R., Zuzan, H., Olson, J., Marks, J. and Nevins, J. (2001). Predicting the clinical status of human breast cancer by using gene expression profiles. *Proceedings of the National Academy of Sciences (USA)* **98** 11462–11467.

[75] Yao, Y., Rosasco, L. and Caponneto, A. (2005). On early stopping in gradient descent learning. Tech. rep., University of California, Berkeley.
URL http://math.berkeley.edu/~yao/publications/earlystop.pdf

[76] Zhang, T. and Yu, B. (2005). Boosting with early stopping: convergence and consistency. *The Annals of Statistics* **33** 1538–1579.

[77] Zhao, P. and Yu, B. (2005). Boosted Lasso. Tech. rep., University of California, Berkeley.

[78] Zhao, P. and Yu, B. (2006). On model selection consistency of Lasso. Tech. rep.,

University of California, Berkeley.

[79] ZHU, J., ROSSET, S., ZOU, H. and HASTIE, T. (2005). Multiclass Adaboost. Tech. rep., Stanford University.
URL `http://www-stat.stanford.edu/~hastie/pub.htm`

SEMINAR FÜR STATISTIK
ETH ZÜRICH
CH-8092 ZÜRICH
SWITZERLAND
E-MAIL: buhlmann@stat.math.ethz.ch

INSTITUT FÜR MEDIZININFORMATIK,
BIOMETRIE UND EPIDEMIOLOGIE
FRIEDRICH-ALEXANDER-UNIVERSITÄT
ERLANGEN-NÜRNBERG
WALDSTRASSE 6, D-91054 ERLANGEN, GERMANY
E-MAIL: torsten.hothorn@rzmail.uni-erlangen.de

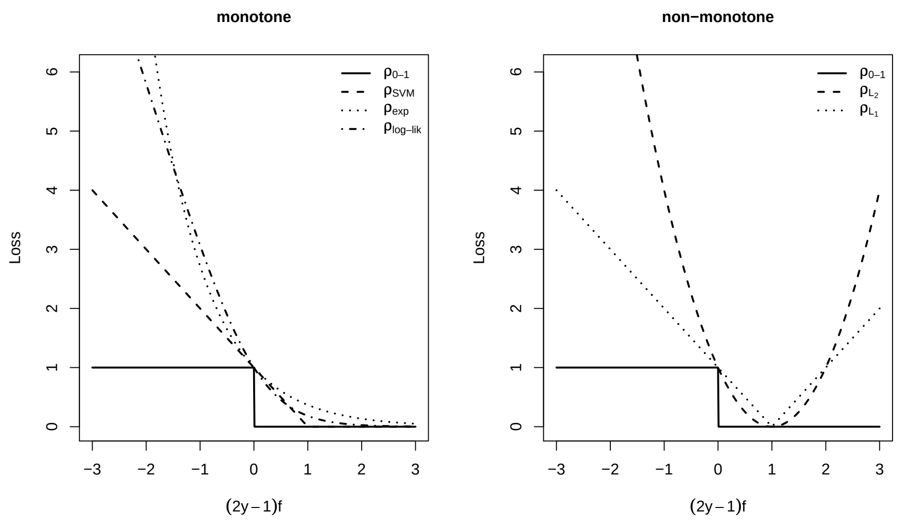

Fig 1. *Losses, as a function of the margin $\tilde{y}f = (2y-1)f$, for binary classification. Left panel with monotone loss functions: 0-1 loss, exponential loss, negative log-likelihood, hinge loss (SVM); right panel with non-monotone loss functions: squared error (L_2) and absolute error (L_1) as in (3.5).*

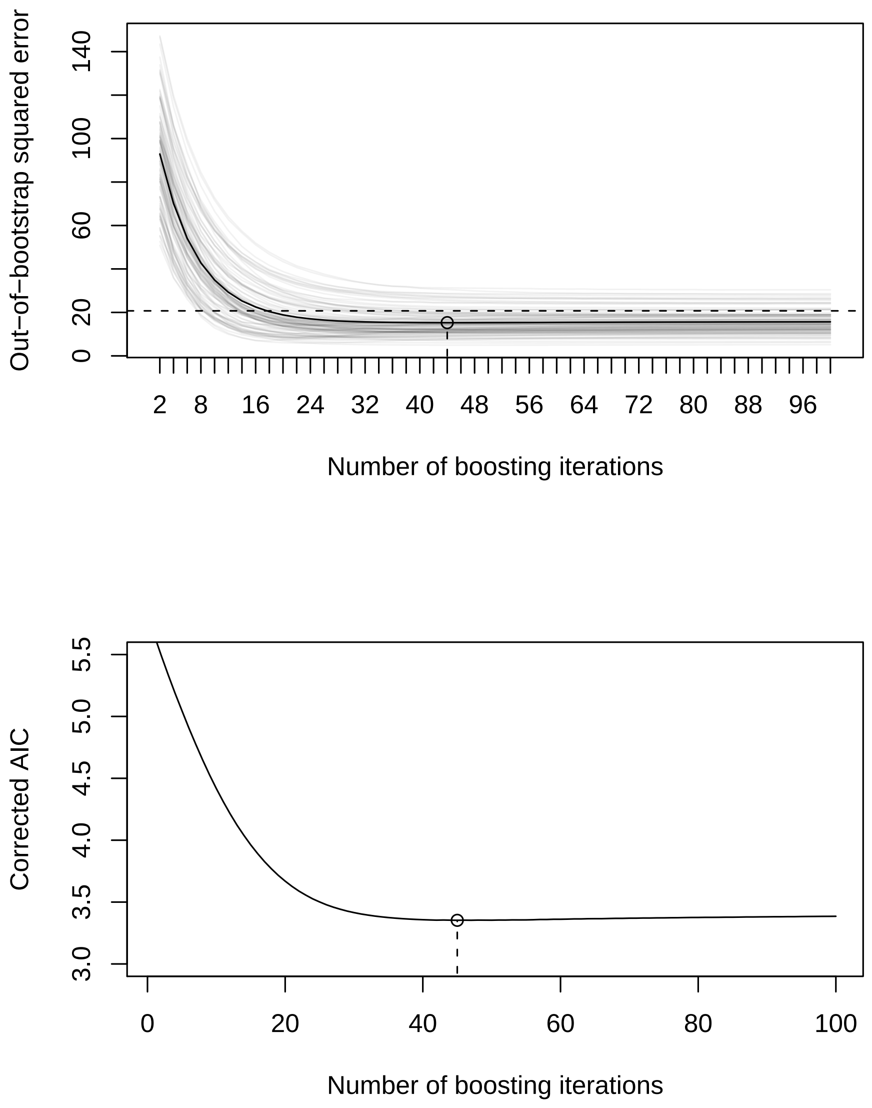

FIG 2. *`bodyfat` data: Out-of-bootstrap squared error for varying number of boosting iterations m_{stop} (top). The dashed horizontal line depicts the out-of-bootstrap error of the linear model for the pre-selected variables `hipcirc`, `kneebreadth` and `anthro3a` fitted via ordinary least squares. The lower part show the corrected AIC criterion.*

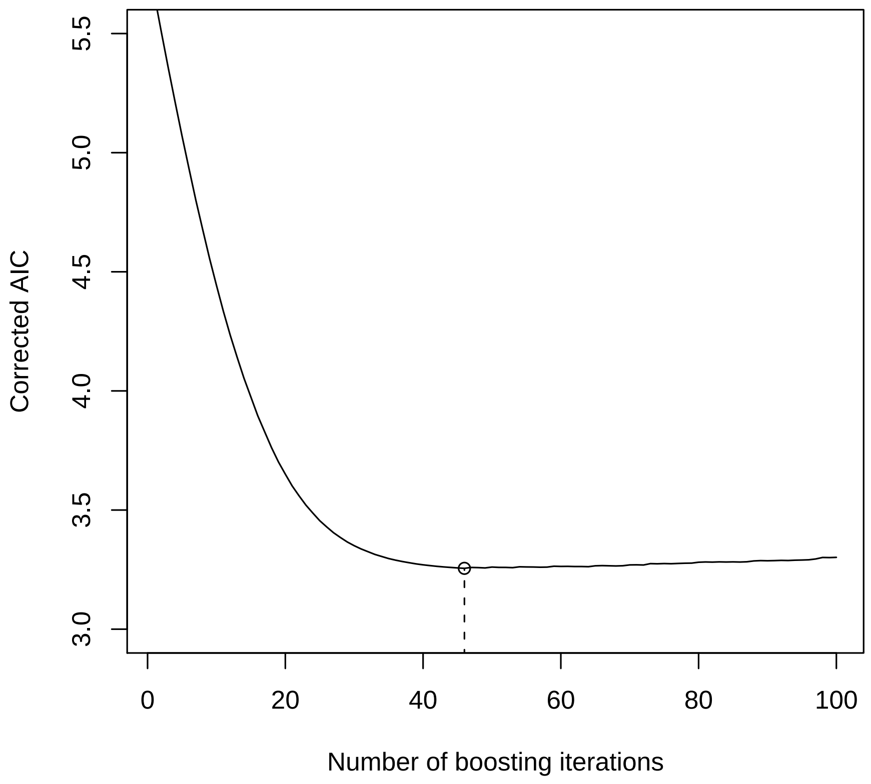

FIG 3. *`bodyfat` data: Corrected AIC as a function of the number of boosting iterations m_{stop} for fitting an additive model via `gamboost`.*

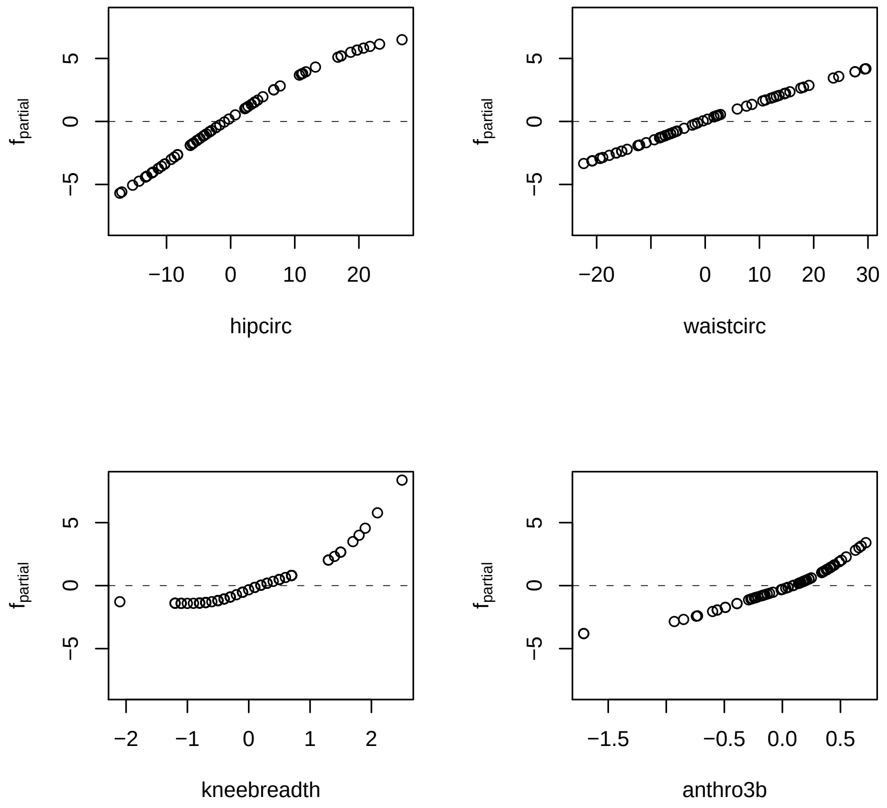

FIG 4. *bodyfat data: Partial contributions of four covariates in an additive model.*

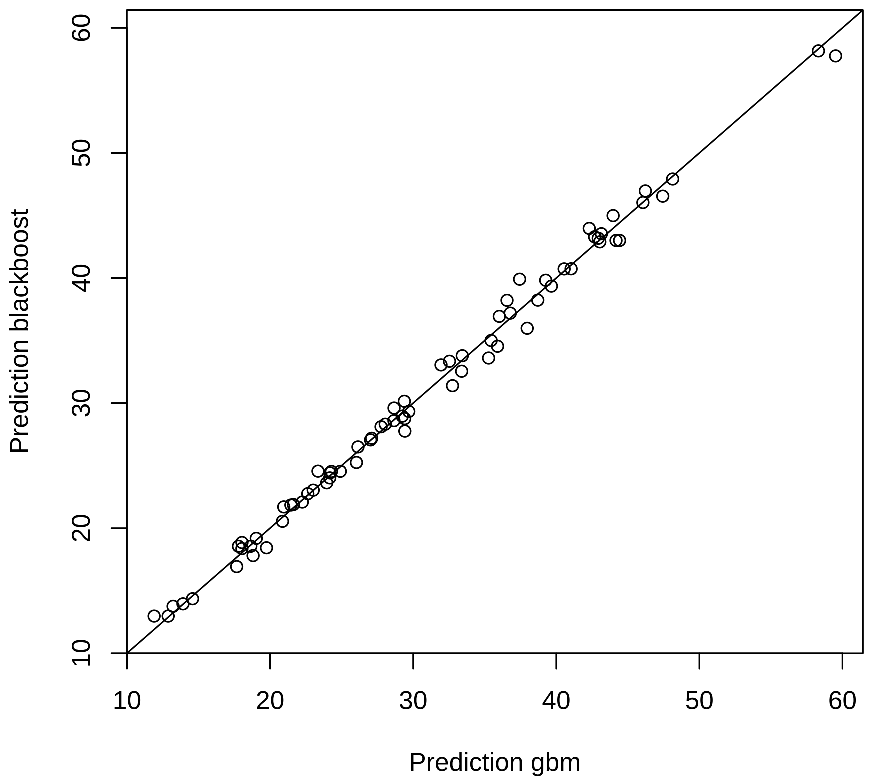

FIG 5. *`bodyfat` data: Fitted values of both the* **gbm** *and* **mboost** *implementations of L_2Boosting with different regression trees as base learners.*

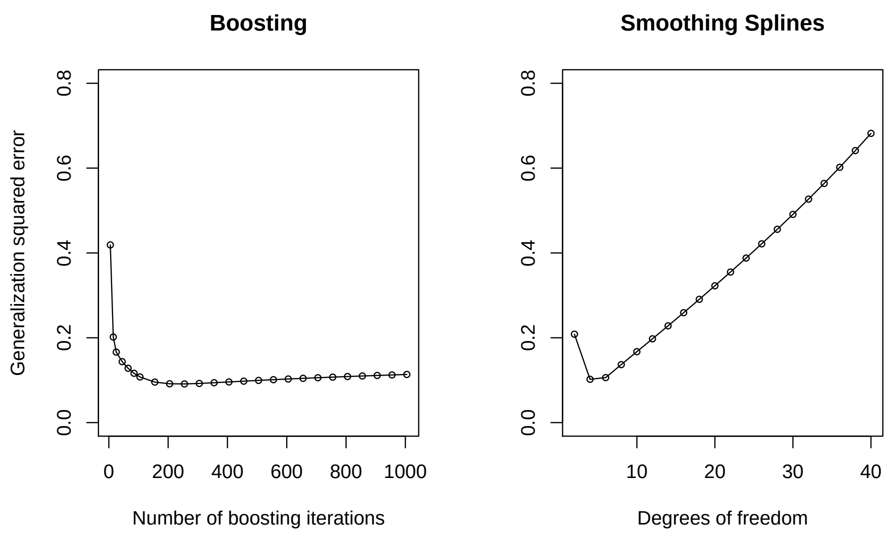

Fig 6. *Mean squared prediction error* $\mathbb{E}[(f(X) - \hat{f}(X))^2]$ *for the regression model* $Y_i = 0.8X_i + \sin(6X_i) + \varepsilon_i \ (i = 1, \ldots, n = 100)$, *with* $\varepsilon \sim \mathcal{N}(0, 2), X_i \sim \mathcal{U}(-1/2, 1/2)$, *averaged over* 100 *simulation runs. Left:* L_2*Boosting with smoothing spline base procedure (having fixed degrees of freedom df = 4) and using* $\nu = 0.1$, *for varying number of boosting iterations. Right: single smoothing spline with varying degrees of freedom.*

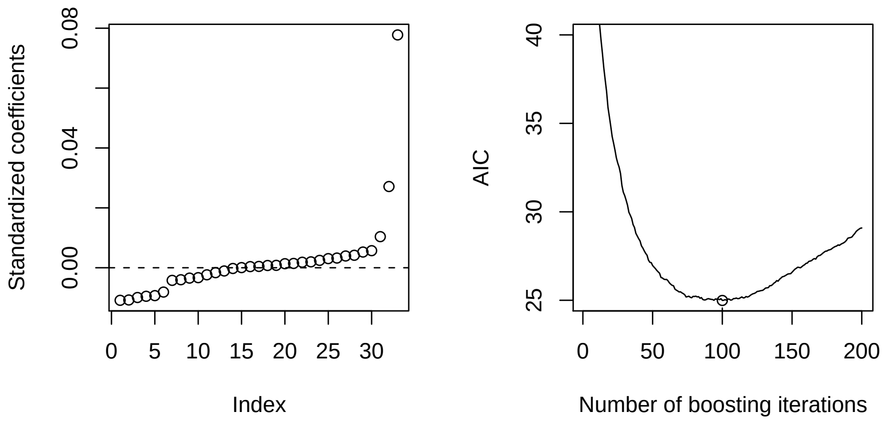

FIG 7. *`westbc` data: Standardized regression coefficients* $\hat{\beta}^{(j)}\sqrt{\widehat{Var}(X^{(j)})}$ *(left panel) for* $m_{stop} = 100$ *determined from the classical AIC criterion shown in the right panel.*

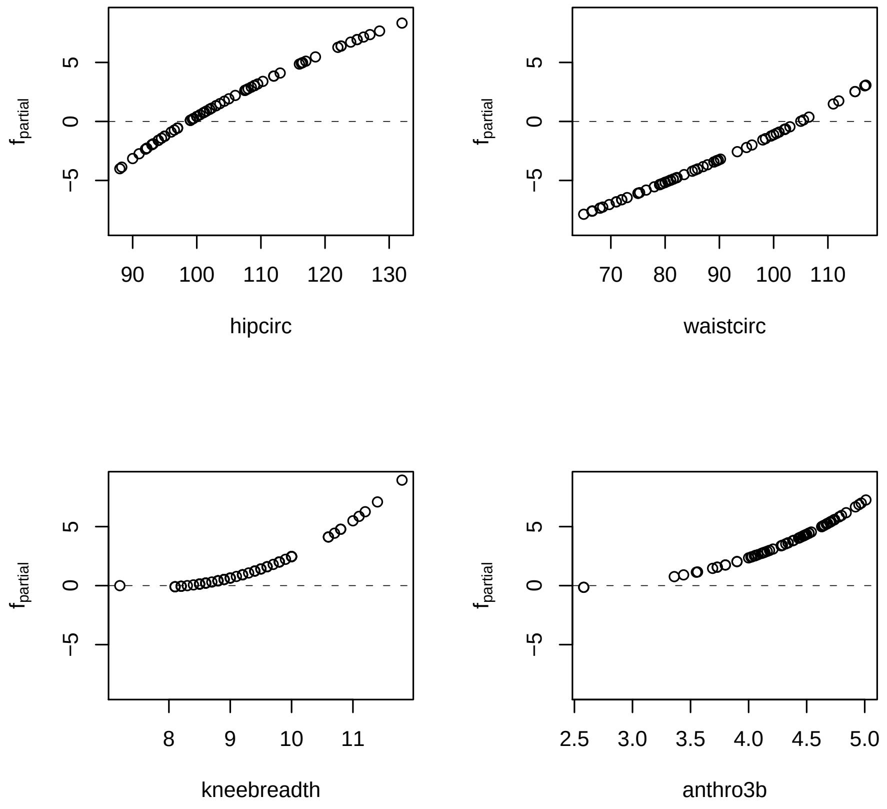

FIG 8. *`bodyfat` data: Partial fits for a linear model including fractional polynomials.*

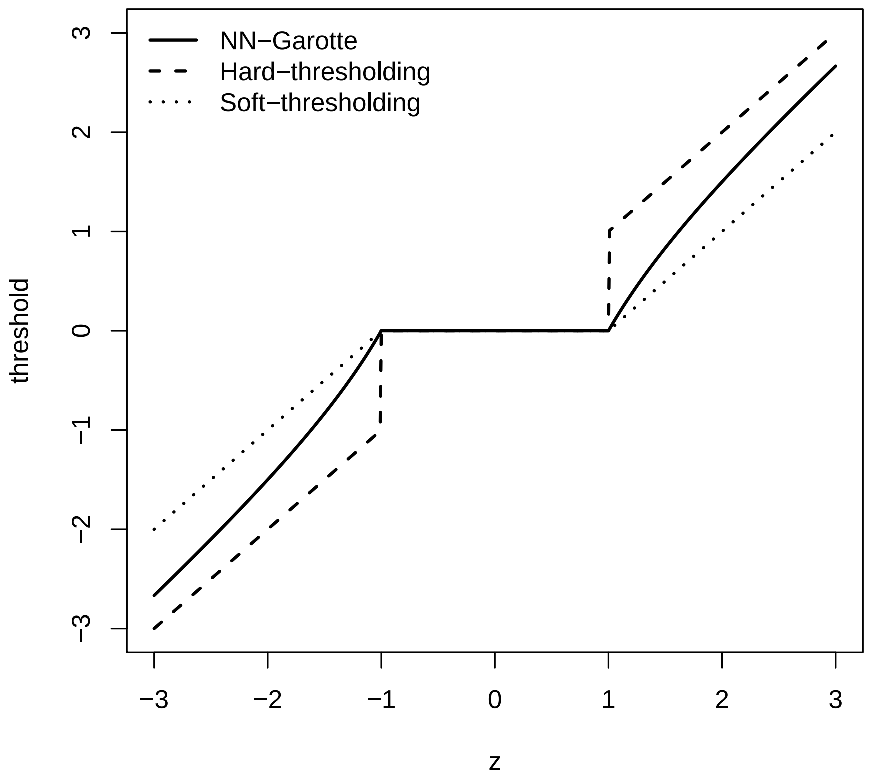

Fig 9. *Hard-threshold (dotted-dashed), soft-threshold (dotted) and non-negative garrote (solid) estimator in a linear model with orthonormal design. The value on the x-abscissa, denoted by z, is a single component of* $\mathbf{X}^\top \mathbf{Y}$.

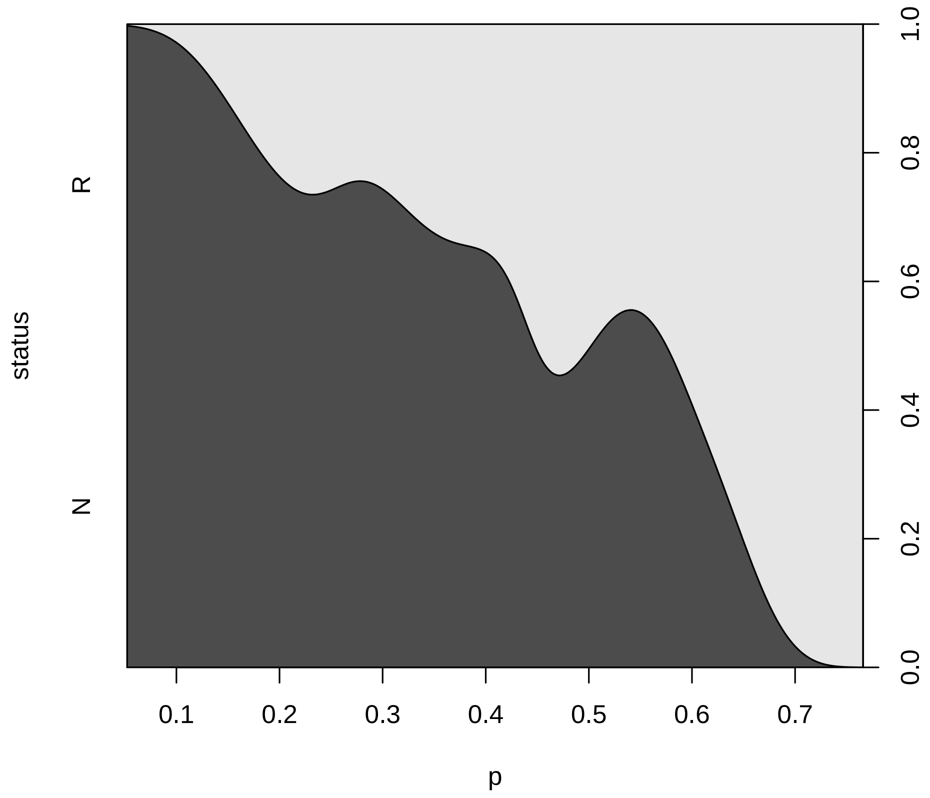

FIG 10. *`wpbc` data: Conditional density plot of the fitted probabilities of recurrence / non-recurrence.*

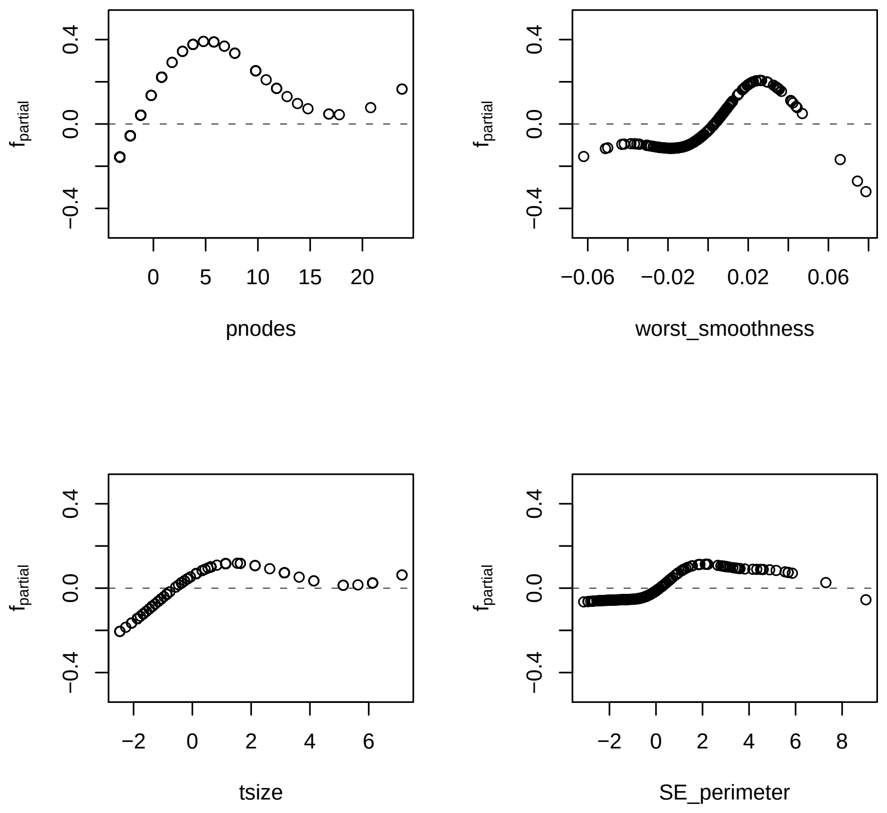

FIG 11. *wpbc data: Partial contributions of four selected covariates in an additive logistic model.*

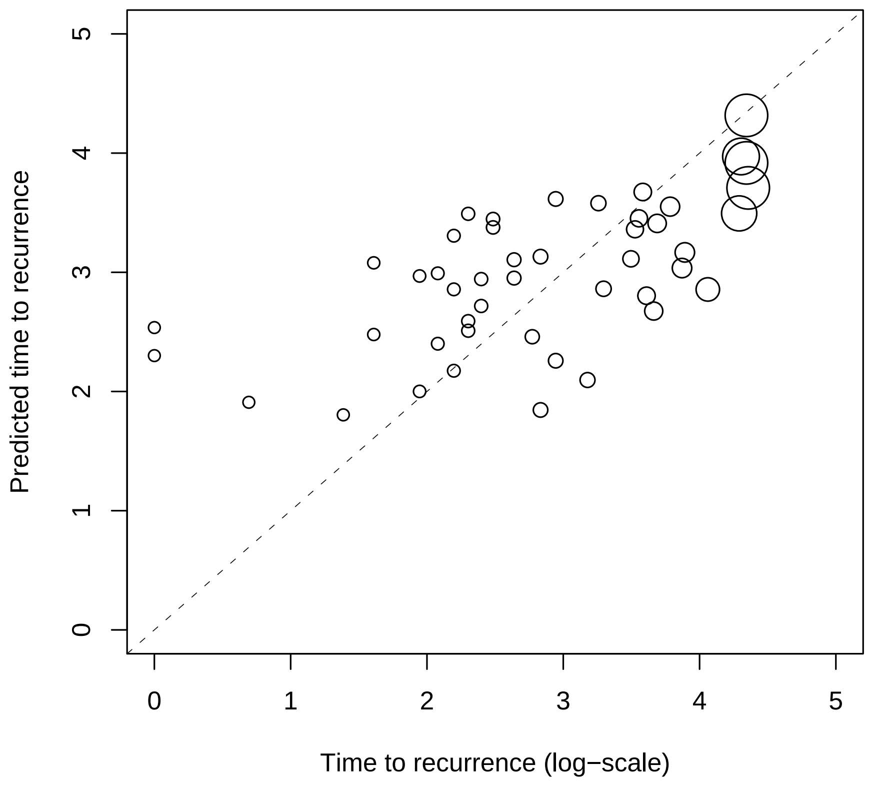

Fig 12. *`wpbc` data: Fitted values of a weighted linear model taking both time to recurrence and censoring information into account. The radius of the circles is proportional to the IPC weight of the corresponding observation, censored observations with IPC weight zero are not plotted.*

range spaces	$\rho(y, f)$	$f^*(x)$	algorithm
$y \in \{0, 1\}, f \in \mathbb{R}$	$\exp(-(2y-1)f)$	$\frac{1}{2}\log\left(\frac{p(x)}{1-p(x)}\right)$	AdaBoost
$y \in \{0, 1\}, f \in \mathbb{R}$	$\log_2(1 + e^{-2(2y-1)f})$	$\frac{1}{2}\log\left(\frac{p(x)}{1-p(x)}\right)$	LogitBoost / BinomialBoosting
$y \in \mathbb{R}, f \in \mathbb{R}$	$\frac{1}{2}\lvert y-f\rvert^2$	$\mathbb{E}[Y \lvert X = x]$	L_2Boosting

TABLE 1
Various loss functions $\rho(y, f)$, population minimizers $f^*(x)$ and names of corresponding boosting algorithms; $p(x) = \mathbb{P}[Y = 1|X = x]$.