Matthias Schmid & Torsten Hothorn

Boosting Additive Models using Component-wise P-Splines

Technical Report Number 002, 2007
Department of Statistics
University of Munich

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

Boosting Additive Models using Component-wise P-Splines

Matthias Schmid[1]
Institut für Medizininformatik, Biometrie und Epidemiologie
Friedrich-Alexander-Universität Erlangen-Nürnberg
Waldstraße 6, D–91054 Erlangen, Germany

Torsten Hothorn[1]
Institut für Statistik
Ludwig-Maximilians-Universität München
Ludwigstraße 33, D–80539 München, Germany

Abstract

We consider an efficient approximation of Bühlmann & Yu's L_2Boosting algorithm with component-wise smoothing splines. Smoothing spline base-learners are replaced by P-spline base-learners which yield similar prediction errors but are more advantageous from a computational point of view. In particular, we give a detailed analysis on the effect of various P-spline hyper-parameters on the boosting fit. In addition, we derive a new theoretical result on the relationship between the boosting stopping iteration and the step length factor used for shrinking the boosting estimates.

Key words: L_2Boosting, P-splines, smoothing splines, additive models, variable selection, component-wise base-learners

1 Introduction

In recent years boosting has developed into one of the most important techniques for fitting regression models in high-dimensional data settings. Originally, the technique was designed as a machine learning procedure for improving the prediction of binary outcomes (Freund and Schapire, 1996, 1997). Later, Breiman (1998, 1999) and Friedman et al. (2000) have shown that boosting can also be regarded as a gradient-descent algorithm in function space:

[1]Supported by Deutsche Forschungsgemeinschaft, grant HO 3242/1–3

In each iteration of the algorithm, a so-called base-learner (e.g., a tree or a least squares estimator) is fitted to the negative gradient of a pre-specified loss function. The current estimate of the predictor function is then updated with the actual estimate of the negative gradient, which automatically results in an additive model fit. Since the work of Friedman et al. (2000), a variety of boosting algorithms for various statistical problems and loss functions has been suggested in the literature; for an overview, we refer to Bühlmann and Hothorn (2008).

By explicitly considering least squares base-learners in combination with the L_2 loss function, Bühlmann and Yu (2003) have established boosting as a technique for fitting additive regression models ("L_2Boosting"). In particular, the authors showed how to choose an alternative base-learner such that the boosting algorithm includes a built-in variable selection technique (Bühlmann and Yu, 2003; Bühlmann, 2006). Thus, when the number of covariates p in a data set is large (and when selecting a small number of relevant covariates is desirable), boosting is usually superior to standard estimation techniques for regression models (such as stepwise linear regression, which, e.g., cannot be applied if p is larger than the number of observations n).

In this paper, the focus is on a promising application of L_2Boosting: We consider fitting additive regression models with a continuous outcome and smooth functions of the predictors. In this case, a smooth function is fitted to the negative gradient of the loss function in each iteration. The starting point of the paper is the well-established approach of Bühlmann and Yu (2003), who used smoothing splines as the base-learner. Apart from deriving a number of theoretically appealing results on smoothing spline base-learners, Bühlmann and Yu (2003) have shown that this strategy is competitive with standard estimation techniques for additive models (e.g., backfitting) and is even able to outperform them.

From a computational point of view, however, smoothing splines are clearly less efficient than other smooth base-learners. This is mainly due to the structure of their integrated squared second-order derivative penalty which is relatively time-consuming to evaluate (Eilers and Marx, 1996; Marx and Eilers, 1998). We therefore investigate whether smoothing spline base-learners can be replaced by a simpler base-learner yielding approximately the same performance results. A natural choice to approximate smoothing splines is the P-spline approach introduced by Eilers and Marx (1996). P-splines, which have become a standard tool for estimating generalized additive models, use a penalty based on higher-order differences of regression coefficients of adjacent knots. If the difference order is equal to 2, the penalty is a discrete approximation of the smoothing spline penalty. A major benefit of this approximation is that the dimensionality of the penalty is greatly reduced, so that P-spline estimates are far less time-consuming to evaluate. Thus, when used as base-learners for a large number of variables, P-splines can greatly increase the efficiency of a boosting algorithm.

In the following we will show that this increase in efficiency is possible even without decreasing the predictive performance of the L_2Boosting algorithm. By means of theoretical results and an extensive simulation study we compare smoothing spline base-learners to P-spline base-learners and show that using the latter is a successful alternative to Bühlmann and Yu's approach using smoothing spline base-learners. In addition, we investigate the effect of various P-spline hyper-parameters (such as the smoothing parameter and the number of knots) on the performance of L_2Boosting.

It should be noted that P-splines have been used before in a boosting context (Tutz and Binder, 2006). However, the authors used a likelihood-based learning procedure, which is related but not equivalent to L_2Boosting. Also, Tutz and Binder (2006) did not provide a systematic investigation on the effect of P-spline hyper-parameters on the boosting fit.

The paper is organized as follows: In Section 2, we briefly outline the concept of L_2Boosting with smooth components. Moreover, we analyze the effect of various boosting tuning parameters and P-spline hyper-parameters on the behavior of the L_2Boosting algorithm. In Section 3, smoothing spline base-learners and P-spline base-learners are compared by means of a simulation study. This study is based on artificial data, as well as on a real world data set originally analyzed by Garcia et al. (2005). Section 4 deals with various computational issues involved in making L_2Boosting with P-spline base-learners a computationally feasible estimation technique. A summary of the results of this paper is given in Section 5.

2 L_2Boosting with smooth components

2.1 The FGD algorithm

Consider a set of realizations of i.i.d. random variables $(X_1, Y_1), \ldots, (X_n, Y_n)$, where $X_1, \ldots, X_n$ are p-dimensional vectors of covariates and $Y_1, \ldots, Y_n$ are one-dimensional response variables. We want to minimize the real-valued function

$$f^* := \underset{f(\cdot)}{\operatorname{argmin}} \, \mathrm{E}[\rho(Y, f(X))] \ , \tag{1}$$

where (X, Y) follows the same distribution as each of the (X_i, Y_i), $i = 1, \ldots, n$. The function ρ is a loss function that is assumed to be differentiable with respect to $f(X)$. Estimation of f^* is performed by minimizing the empirical risk $\sum_{i=1}^n \rho(Y_i, f(X_i))$ with respect to f. We consider the following boosting algorithm introduced by Friedman et al. (2000) and Friedman (2001):

1. Initialize the n-dimensional vector $\hat{f}^{[0]}$ with an offset value, e.g., $\hat{f}^{[0]} \equiv 0$. Set $m = 0$ and specify a base-learner with one dependent variable and one covariate (e.g., a simple linear regression estimator).
2. Increase m by 1. Compute the negative gradient $-\frac{\partial}{\partial f}\rho(Y, f)$ and evaluate at $\hat{f}^{[m-1]}(X_i)$, $i = 1, \ldots, n$. This yields the negative gradient vector $U^{[m-1]} = (U_i^{[m-1]})_{i=1,\ldots,n} := (-\frac{\partial}{\partial f}\rho(Y, f)|_{Y=Y_i, f=\hat{f}^{[m-1]}(X_i)})_{i=1,\ldots,n}$.

3. Fit the negative gradient $U^{[m-1]}$ to each of the p components of X (i.e., to each covariate) separately by p times using the regression estimator specified in (1). This yields p vectors, where each vector is an estimate of the negative gradient vector $U^{[m-1]}$.

4. Select the component of X which fits $U^{[m-1]}$ best according to a pre-specified goodness-of-fit criterion. Set $\hat{U}^{[m-1]}$ equal to the fitted values from the corresponding best model fitted in (3).

5. Update $\hat{f}^{[m]} = \hat{f}^{[m-1]} + \nu \, \hat{U}^{[m-1]}$, where $0 < \nu \leq 1$ is a real-valued step length factor.

6. Iterate Steps 2–5 until $m = m_{\text{stop}}$ for some stopping iteration m_{stop}.

The above algorithm can be interpreted as a negative gradient descent algorithm in function space: In each step, an estimate of the true negative gradient of the loss function is added to the current estimate of f^*. Thus, a structural (regression) relationship between Y and the covariate vector X is established. It can also be seen that the above algorithm carries out variable selection in each iteration, as only one component of X is updated in Step (5). This property makes the algorithm applicable even if $p > n$. Due to the additive structure in (5), the final boosting estimate at iteration m_{stop} can be interpreted as an additive model fit but will typically depend on only a subset of the p components of X. We refer to the above algorithm as "component-wise functional gradient descent (FGD) boosting".

Choosing an appropriate value of m_{stop}, i.e., stopping the algorithm before convergence, is necessary to prevent component-wise FGD from overfitting the data. The tuning parameter ν can be regarded as the step length of the gradient descent algorithm, or, alternatively, as a shrinkage factor of the regression coefficient estimates. For details on how to find appropriate values of m_{stop} and ν we refer to Section 2.2.

As we want to consider additive models with a continuous outcome, we assume $Y_1, \ldots, Y_n$ to be continuous and set ρ equal to the squared error loss $(Y - f(X))^2$. In this case, FGD corresponds to a stagewise re-fitting of the residuals $\hat{U}_i^{[m-1]} = Y_i - \hat{f}_i^{[m-1]}$, $i = 1, \ldots, n$. A natural choice for the goodness-of-fit criterion in Step (4) is the fraction of explained variance R^2. To incorporate smooth functions of the covariates into the model fit, we use *smoothing splines* (Green and Silverman, 1994) and *P-splines* (Eilers and Marx, 1996) as base-learners. The final boosting estimate $\hat{f}^{[m_{\text{stop}}]}$ then becomes a sum of spline functions and can be interpreted as the fit of an additive model with smooth functions of the predictors.

Using cubic *smoothing spline* base-learners for boosting additive models has been suggested by Bühlmann and Yu (2003). In this case, the knot positions of the splines coincide with the data values. The roughness penalty is an integral of the squared second derivative of the corresponding spline – see Green and Silverman (1994) or Hastie and Tibshirani (1990) for details. *P-splines* are a special form of penalized regression splines. In this paper, we pursue the strategy of Eilers and Marx (1996), who suggested to use the numerically advantageous B-spline basis, along with a large number of equidistant knots and a roughness penalty based on higher-order squared differences of the coefficients of adjacent basis functions. Eilers and Marx (1996) showed that setting the difference order to $k = 2$ corresponds to an approximation of the integrated squared second derivative penalty used by smoothing splines. Therefore, as we want to compare P-spline base-learners with smoothing spline base-learners, we choose cubic B-spline basis functions and set the difference order equal to $k = 2$.

2.2 Reflections on boosting tuning parameters and hyper-parameters of smoothing spline and P-spline base-learners

Boosting with smoothing spline base-learners and cubic P-spline base-learners (with difference order $k = 2$) generally involves the following parameters:

1. the stopping iteration m_{stop},
2. the step length factor / shrinkage parameter ν,
3. the degrees of freedom df, which are equal to the trace of the smoothing spline / P-spline hat matrix. Small values of df correspond to a large amount of smoothing, i.e., to a relatively large bias and small variance of the respective spline.
4. the number of equidistant knots P (only when P-spline base-learners are used; in case of smoothing spline base-learners, the knots are not equidistant and their number is always equal to n).

First consider the *stopping iteration* m_{stop}, which should be chosen such that overfitting of the data is prevented. In the literature, two strategies for determining m_{stop} have been suggested: cross-validation and AIC-type stopping criteria (Bühlmann, 2006). In this paper, we use the AIC approach, which is computationally far less intensive than cross-validation and which has been shown to work well in a boosting context.

AIC-based stopping works as follows: Define $\mathcal{H}^j$ to be the $n \times n$ hat matrix of the (smoothing or P-spline) fitting operator using the jth component of X. It can then be shown (cf. Bühlmann and Yu, 2003) that the hat matrix of L_2Boosting at iteration m is equal to

$$\mathcal{B}_m = I - (I - \nu\mathcal{H}^{\mathcal{S}_m}) \cdot \ldots \cdot (I - \nu\mathcal{H}^{\mathcal{S}_1}) \ , \tag{2}$$

where $\mathcal{S}_r \in \{1, \ldots, p\}$ denotes the index of the covariate which is selected by the component-wise base-learner in the rth boosting iteration. In the special case where there is only one covariate (X one-dimensional), (2) reduces to

$$\mathcal{B}_m = I - (I - \nu\mathcal{H}^1)^m \ . \tag{3}$$

Thus, for each iteration, one obtains a corrected AIC criterion (Hurvich et al., 1998) given by

$$\text{AIC}(m) = \log(\hat{\sigma}_m^2) + \frac{1 + \text{df}(m)/n}{(1 - \text{df}(m) + 2)/n} \ , \tag{4}$$

where $\hat{\sigma}^2_m := n^{-1} \sum_{i=1}^n (Y_i - (\mathcal{B}_m Y)_i)^2$ is an estimate of the residual variance of the additive model and $\mathrm{df}(m)$ is the trace of $\mathcal{B}_m$, i.e., the degrees of freedom at iteration m. The series of corrected AIC values ($m = 1, \dots$) yields the stopping iteration m_{stop}, which is the iteration with lowest corresponding AIC value.

Next consider the value of the *step length* ν. In the spirit of functional gradient descent methods, Friedman (2001) has suggested to estimate $\nu = \nu(m)$ in each boosting iteration by performing a line search. Other authors (see Bühlmann and Hothorn, 2008) have argued that this (rather time-consuming) procedure is of relatively low importance when it comes to the predictive ability of FGD. They favored to use a constant value of the step length ν, with the only requirement that ν should be small (in order to obtain a stagewise adaption of the true predictor function f^*). In this paper, we use constant values of ν, e.g., $\nu = 0.01$ or $\nu = 0.1$.

Bühlmann and Hothorn (2008) also noted that the number m_{stop} of boosting operations usually increases with decreasing value of ν, so there seems to be a dependence between m_{stop} and ν. We now give an argument that when using the corrected AIC criterion (4) for stopping, this dependence is approximately *linear*, so that m_{stop} is approximately inversely proportional to ν. To derive this relationship, we assume that there is only one covariate, such that the hat matrix $\mathcal{B}_m$ is of the form (3).

Theorem 1. *Suppose that L_2Boosting is applied to an i.i.d. data set with a one-dimensional response variable Y and a one-dimensional covariate X. Consider a constant step length value $0 < \nu < 1$ and a proportionality factor $\kappa > 1$. Define $\mathcal{H}$ to be the hat matrix of the the base-learner and let $\mathcal{H} = \Gamma \Lambda \Gamma^\top$ be its spectral decomposition (i.e., Γ is an orthogonal matrix and Λ is a diagonal matrix containing the eigenvalues of $\mathcal{H}$ on its main diagonal). Then the following relationship holds:*

$$\left(I - \frac{\nu}{\kappa}\mathcal{H}\right)^\kappa = I - \nu\mathcal{H} + \mathcal{R} \ , \tag{5}$$

where $\mathcal{R} := \frac{\nu^2}{2}\frac{\kappa-1}{\kappa}\Gamma\Lambda^2(I - \frac{\nu^*}{\kappa}\Lambda)^{\kappa-1}\Gamma^\top$ and $0 \leq \nu^* \leq \nu$.

Proof. Since $\mathcal{H} = \Gamma\Lambda\Gamma^\top$, we obtain $(I - \frac{\nu}{\kappa}\mathcal{H})^\kappa = \Gamma(I - \frac{\nu}{\kappa}\Lambda)^\kappa\Gamma^\top$. Let $\eta_1, \dots, \eta_n$ be the eigenvalues of $\mathcal{H}$, i.e., the diagonal elements of Λ. Define

$$f_i(\nu) := \left(1 - \frac{\nu}{\kappa}\eta_i\right)^\kappa \ , \ i = 1, \dots, n \ . \tag{6}$$

Theorem 1 then follows from a Taylor series expansion of the functions f_i at $\nu_0 = 0$. □

The interpretation of Theorem 1 is as follows: Since (a) the eigenvalues η_i, $i = 1, \dots, n$, of the smoothing and P-spline hat matrices $\mathcal{H}$ are contained in $[0, 1]$ and (b) the value of ν is contained in $(0, 1)$ by assumption, the elements of $\mathcal{R}$ typically become small. Consequently, $(I - \frac{\nu}{\kappa}\mathcal{H})^\kappa$ can be approximated by $I - \nu\mathcal{H}$, and the hat matrix of the boosting fit at iteration

m can be approximated by

$$\mathcal{B}_m = I - (I - \nu\mathcal{H})^m \approx I - \left(I - \frac{\nu}{\kappa}\mathcal{H}\right)^{\kappa m} \ . \tag{7}$$

We thus see that the boosting hat matrix $\mathcal{B}_m$ with step size ν is approximately the same as the boosting hat matrix $\mathcal{B}_{\kappa m}$ with step size ν/κ. Since the corrected AIC criterion (4) is a function of the boosting hat matrix $\mathcal{B}_m$ only, the stopping iteration m_{stop} corresponding to step size ν/κ will also be approximately κ times the stopping iteration corresponding to step size ν. The choice of ν therefore is of rather low importance if the corrected AIC criterion (4) is used for stopping: The stopping iteration m_{stop} automatically adapts to the value of ν, yielding approximately the same boosting fit. Note that we have only shown this relationship for FGD with one covariate. Empirically, we will see in Section 3 that the same relationship also holds true for the component-wise FGD algorithm.

Next consider the value of the *degrees of freedom* df. Important results on this parameter have been derived by Bühlmann and Yu (2003), who argued that smoothing spline base-learners should have a large bias but low variance (”weak learners”). Only in this case a stagewise adaption of the true predictor function f^* (and thus a good predictive performance of FGD) is possible. When spline functions are used as base-learners, a large bias and low variance can be obtained by making the degrees of freedom df $:= \text{tr}(\mathcal{H}^1)$ small. Bühlmann and Yu (2003) showed that for a sufficiently small value of df, the overall bias of the boosting estimate can nevertheless be made small by carrying out an appropriate number of boosting iterations m_{stop}. Moreover, boosting with weak smoothing spline base-learners is even able to adapt to higher order degree smoothness in the data (e.g., spline functions with a degree larger than 3). Bühlmann and Yu (2003) therefore recommended to chose a small value of df for the base-learner (e.g., df = 4), and to keep this number fixed in each boosting iteration. Clearly, small values of df correspond to large smoothing parameters λ of a smoothing spline (and also of a P-spline).

We now give an additional justification for measuring the weakness of a smoothing spline or P-spline base-learner by its degrees of freedom. Consider the boosting hat matrix (3) with only one covariate. As noted by Bühlmann and Hothorn (2008), (3) suggests that if $0 < ||I - \nu\mathcal{H}^1|| < 1$ for a suitable norm, the boosting algorithm has a ”learning capacity”, in the sense that the residual vector is ”smaller” than the response vector Y. If $||I - \nu\mathcal{H}^1||$ is close to 1, the base-learner is weak, and a large value m_{stop} is required for obtaining the ”optimal” fit. Consequently, the weakness of a base-learner can be measured by $||I - \nu\mathcal{H}^1||$. The following theorem states that if using a scaled version of the Frobenius matrix norm, the weakness of a smoothing spline or P-spline base-learner can be directly related to its degrees of freedom df $= \text{tr}(\mathcal{H}^1)$.

Theorem 2. *For fixed n, consider a smoothing spline or P-spline with smoothing parameter $\lambda > 0$ and hat matrix $\mathcal{H} = \mathcal{H}(\lambda)$. Further consider the scaled squared Frobenius norm $||I - \nu\mathcal{H}|| := n^{-1}\text{tr}((I - \nu\mathcal{H})^\top(I - \nu\mathcal{H}))$ of the matrix $I - \nu\mathcal{H}$, which measures the weakness of the corresponding smoothing spline or P-spline base-learner. If $0 < \nu < 1$, the following* *results hold:*

a) $||I - \nu\mathcal{H}||$ *is strictly increasing in* λ.

b) $||I - \nu\mathcal{H}||$ *converges to 1 as* $\lambda \to \infty$.

Proof. It is well known from the literature that the trace of the hat matrix $\mathcal{H}$ of a smoothing spline or P-spline is equal to

$$\mathrm{tr}(\mathcal{H}) = \mathrm{tr}\big((I + \lambda K)^{-1}\big) \ , \tag{8}$$

where K is a positive semi-definite matrix with eigenvalues $d_1, \dots, d_n \geq 0$. Consequently, $\mathrm{tr}(\mathcal{H}) = \sum_{i=1}^n 1/(1+\lambda d_i)$ and $\mathrm{tr}(\mathcal{H}^2) = \sum_{i=1}^n 1/(1+\lambda d_i)^2$. Now consider the scaled squared Frobenius norm of $I - \nu\mathcal{H}$. We obtain

$$\begin{aligned} ||I - \nu\mathcal{H}|| &= \frac{1}{n}\mathrm{tr}\Big((I - \nu\mathcal{H})^2\Big) \\ &= \frac{1}{n}\Big[\mathrm{tr}(I) - 2\nu\mathrm{tr}(\mathcal{H}) + \nu^2\mathrm{tr}(\mathcal{H}^2)\Big] \\ &= 1 - \frac{2\nu}{n}\sum_{i=1}^n \frac{1}{(1+\lambda d_i)} + \frac{\nu^2}{n}\sum_{i=1}^n \frac{1}{(1+\lambda d_i)^2} \ . \end{aligned} \tag{9}$$

The derivative of (9) with respect to λ is

$$\frac{\partial}{\partial\lambda}||I - \nu\mathcal{H}|| = \frac{2\nu}{n}\sum_{i=1}^n d_i \left[\frac{1}{(1+\lambda d_i)^2} - \nu\frac{1}{(1+\lambda d_i)^3}\right] \ , \tag{10}$$

which is greater than 0 since $0 < \nu < 1$. This proves a). From (9) it is also clear that $||I - \nu\mathcal{H}||$ converges to 1 as $\lambda \to \infty$. This proves b). □

Although the choice of the Frobenius norm is somewhat arbitrary, we see from Theorem 2 that as the smoothing parameter λ increases, the weakness of a smoothing spline or P-spline base-learner increases as well. An increase in λ clearly implies a decrease in the degrees of freedom df of the smoothing spline or P-spline. Thus, Theorem 2 backs the results of Bühlmann and Yu (2003), who suggested to measure the weakness of smoothing spline base-learners by their degrees of freedom. In the following, we will use the same value of df for all components of X. Note that Theorem 2 also backs the argument that ν should be small: In fact, if λ is fixed and ν becomes too large, (9) becomes greater than 1, so that the boosting algorithm loses its learning capacity.

Finally consider the *number of equidistant knots* P when P-spline base-learners are used for boosting. Generally, in a non-boosting context, the number of knots used for a P-spline is not likely to have a big effect on the respective spline regression fit. Ruppert (2002) has shown that there is a minimum necessary number of knots which has to be reached. Further increasing P beyond this number will typically not improve the regression fit. In practice, it is common to use 20–50 knots, which is sufficient for most data situations. In Section 3 we investigate whether the same results hold true when P-splines are used as base-learners for component-wise FGD.

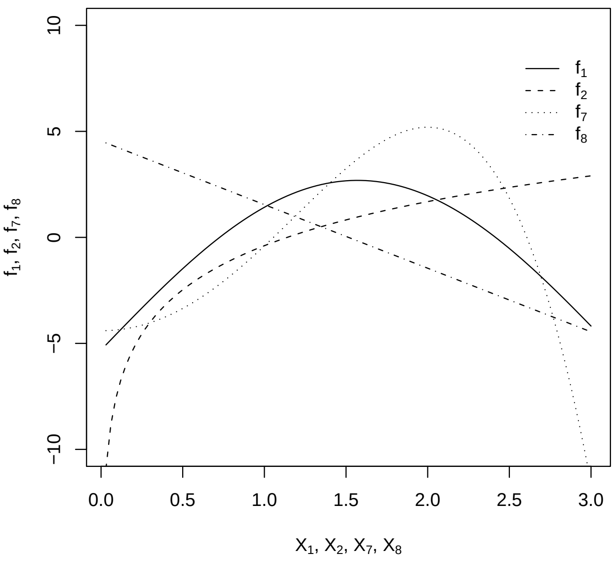

Figure 1: The smooth predictor functions f_1, f_2, f_7, and f_8 used for simulation study 1.

3 Simulations and real world data example

We carried out two simulation studies on the issues discussed in Section 2. Simulation study 1 was based on artificial data with nine covariates $X_1, \ldots, X_9$ and $n = 100$ i.i.d. observations. Each of the $p = 9$ covariates took values on a grid from 0–3 but were randomly permuted so that $X_1, \ldots, X_9$ became independent. The values of the outcome variable Y were generated by means of the model

$$Y = 1 + 8 \cdot \sin(X_1) + 3 \cdot \log(X_2) - 0.8 \cdot (X_7^4 - X_7^3 - 5 \cdot X_7^2) - 3 \cdot X_8 + \epsilon \ , \qquad (11)$$

where $\epsilon \sim \mathcal{N}(0, 3^2)$. Thus, only 4 of the 9 covariates had an effect on Y. The functions $f_1(X_1) := 8 \cdot \sin(X_1)$, $f_2(X_2) := 3 \cdot \log(X_2)$, $f_7(X_7) := -0.8 \cdot (X_7^4 - X_7^3 - 5 \cdot X_7^2)$, and $f_8(X_8) := -3 \cdot X_8$ are smooth functions of these four covariates (see Fig. 1).

Simulation study 2 was based on a real world data set originally analyzed by Garcia et al. (2005). In this study, the aim was to predict the body fat content of $n = 71$ healthy German women by means of $p = 9$ common anthropometric measurements (such as circumference or knee breadth). The dependent variable Y was obtained by Dual X-Ray Absorptiometry (DXA). DXA is an accurate method for measuring the body fat content. However, it is also very expensive, so that, in order to reduce costs, a regression equation for predicting DXA measurements by means of anthropometric measurements is desirable.

The issues to be investigated in both simulation studies were as follows:

1. Does the choice of the base-learner (smoothing splines vs. P-splines) have an effect on the performance of component-wise FGD?
2. Do the simulations studies confirm the theoretical results of Theorems 1 and 2?
3. Which combination of the hyper-parameters discussed in Section 2 yields the best prediction values?
4. Does the component-wise FGD algorithm correctly select the relevant variables X_1, X_2, X_7, X_8 in simulation study 1?

For both simulation studies, the quality of a prediction was measured by the mean squared predictive error (MSE). To obtain unbiased prediction errors, we used training and test data sets generated in the following way: In simulation study 1, we first generated $n_{\text{training}} = 100$ observations for each covariate. Next, 100 samples of Y (of size $n_{\text{training}} = 100$ each) were generated from model (11) by using the covariate values and by repeatedly drawing the values of ϵ from a $\mathcal{N}(0, 3^2)$ distribution. These 100 samples, along with the covariate values, constituted the training data sets. The corresponding 100 test data sets (of size $n_{\text{test}} = 100$ each) were generated in the same way. For each training data set, component-wise FGD was applied, and an estimate of the mean squared prediction error was obtained from the respective test data set. In simulation study 2, we drew 100 bootstrap samples of size $n = 71$ out of the 71 observations. These bootstrap samples constituted the 100 training data sets. Estimates of the mean squared prediction error were obtained from the respective out-of-bootstrap observations.

All simulations were conducted with the R system for statistical computing (version 2.5.0, R Development Core Team, 2007) using the **mboost** add-on package (version 0.6-2, Bühlmann and Hothorn, 2008; Hothorn et al., 2007). The function `gamboost()` in **mboost** allows for running component-wise FGD with either smoothing spline or P-spline base-learners. For P-spline base-learners, it allows the user to specify the number of knots P, the difference order k, and the degrees of freedom df. For smoothing spline base-learners, the degrees of freedom df can be specified.

3.1 Results of simulation study 1

We analyzed the component-wise FGD algorithm with two kinds of base-learners: Smoothing splines (denoted by SSP) and P-Splines with a cubic B-spline basis (de Boor, 1978) and difference order $k = 2$ (denoted by PSP).

Figure 2 shows the mean squared prediction errors (MSE) obtained from the 100 test samples at iteration m_{stop} ($P = 50$, $\nu = 0.1$). Obviously, P-spline base-learners are a very good approximation to smoothing spline base-learners in terms of MSE. This seems to be true for all values of df. Moreover, the corresponding graphs for all other tested values of P ($P = 20, 30, \ldots, 80$, figures not shown here) are very similar to Figure 2. This implies that the number of knots P does not seem to have a big effect on the predictive power of

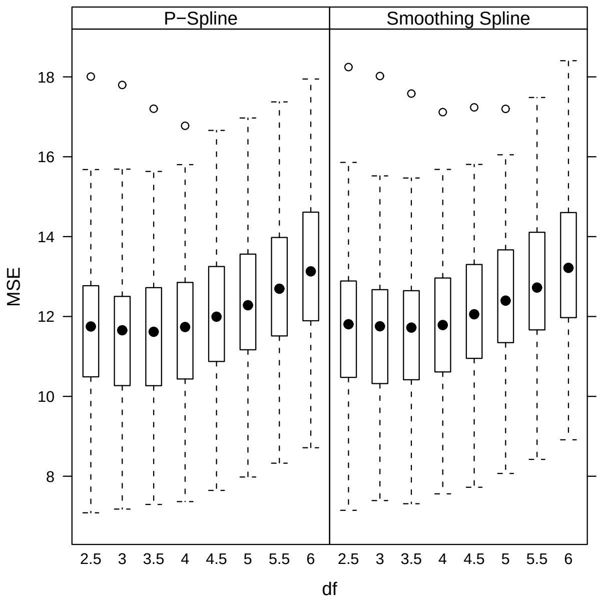

Figure 2: Simulation study 1 - mean squared predictive errors obtained from component–wise FDG with P–spline and smoothing spline base learners ($P = 50$, $\nu = 0.1$).

component-wise FGD. Concerning the degrees of freedom df, we see that small values of df (corresponding to "weak" base-learners) result in smaller prediction errors: The medians of the boxplots shown in Figure 2 take their minima at df $\approx 3 - 4$. If df is larger than four, the medians are slightly increasing in df.

We next illustrate the inversely proportional relationship between ν and m_{stop} (derived in Theorem 1 for FGD with one covariate). The above simulation study with P-spline base-learners was repeated, this time using $\nu = 0.01$. Figure 3 shows a plot of the average values of m_{stop} when $\nu = 0.1$ versus the average values of m_{stop}, divided by 10, when $\nu = 0.01$ ($P = 50$). Figure 3 clearly indicates the approximate inversely proportional relationship between m_{stop} and ν. Even more important, the MSE values of the two component-wise FDG algorithms ($\nu = 0.1$ and $\nu = 0.01$) are very similar (see Figure 4). Thus, decreasing an already small value of ν does not lead to an improvement in the predictive performance of component-wise FGD. For all other values of P, similar results were obtained.

Furthermore, to assess the systematic influence of df, P, and ν on the PSP fit, we estimated the following linear mixed model from the simulated data:

$$\texttt{MSE} \sim \texttt{df} + \texttt{df}^2 + \texttt{P} + \texttt{nu} \mid \texttt{sample} , \qquad (12)$$

where `df` and `P` are (quasi-)continuous covariates corresponding to df and P, respectively, and `nu` is a binary factor variable with levels "$\nu = 0.1$" and "$\nu = 0.01$". Since the same 100 training and test samples were used for all value combinations of df, P, and ν, model (12)

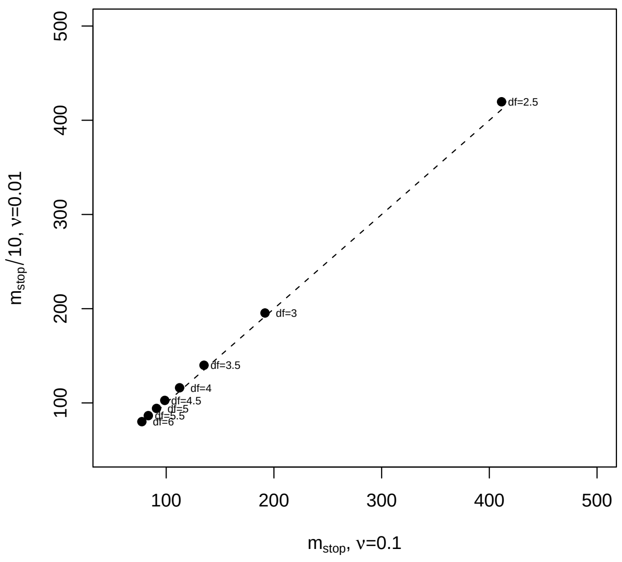

Figure 3: Simulation study 1 - average stopping iterations m_{stop} for step sizes $\nu = 0.1$ and $\nu = 0.01$ (obtained from PSP, $P = 50$).

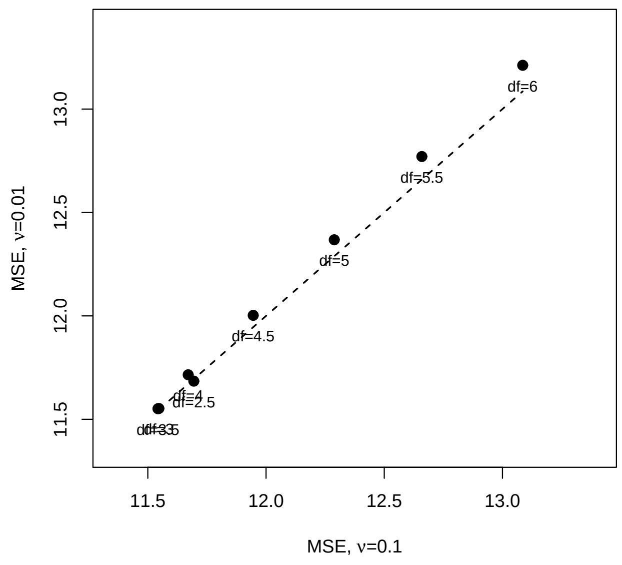

Figure 4: Simulation study 1 - average MSE values for step sizes $\nu = 0.1$ and $\nu = 0.01$ (obtained from PSP, $P = 50$).

Table 1: Simulation study 1 - parameter estimates and hypothesis tests of the fixed effects in model (12). A χ^2-test on the hypothesis "`df` $=$ `df`$^2 = 0$" resulted in a p-value < 0.001.

Effect	Estimate	Std. Error	z-value	p-value
(Intercept)	13.3087	0.1910	69.6764	< 0.001
df	-1.1946	0.0275	-43.3972	< 0.001
df^2	0.1938	0.0032	60.2473	< 0.001
P	0.0012	0.0002	6.7374	< 0.001
$\nu = 0.01$	0.0522	0.0074	7.0824	< 0.001

includes a random intercept `sample` indicating the number of the training/test sample. The estimation results from model (12), which are shown in Table 1, confirm the results presented in Figure 2: the estimates of the fixed effects corresponding to `df` and `df`^{2} are different from zero, while the fixed effects for `P` and `nu` are negligibly small. Although `P` and `nu` have a significant influence on the predictive performance of PSP (at level $\alpha = 0.05$), this finding is most likely due to the large number of simulation runs ($N = 11200$).

Finally, we investigated whether component-wise FGD did a good job at selecting the relevant variables X_1, X_2, X_7, and X_8 from the data. Since the true functions f_1, f_2, f_7, and f_8 are known, we were able to compare these functions with their boosting estimates. In Figure 5 the PSP estimates obtained from component-wise FGD (based on the first of the 100 training data sets) are plotted along with the respective true functions ($\nu = 0.1$, $P = 50$, df $= 3.5$). We see that f_1, f_2, f_7, and f_8 are well-approximated by their corresponding estimates. In addition, although the variables X_4, X_5, and X_9 have been erroneously selected, the corresponding function estimates are close to zero (and can therefore be interpreted as random errors). Similar results were obtained for all 100 training data sets in simulation study 1. This suggests that component-wise FGD with P-spline base-learners is able to select the correct subset of relevant covariates from a data set.

3.2 Results of the simulation study on example 2

Figure 6 shows that the simulation results obtained from the body fat data are very similar to the results obtained from simulation study 1: For all values of df, P-spline base-learners are a good approximation of smoothing spline base-learners in terms of MSE. This is also true if the number of knots P is altered (figures not shown here), so that P does not have a big effect on the P-spline boosting fit. The medians of the MSE values in Figure 6 are smallest when df is small and are only very slightly increasing in df. The inversely proportional relationship between ν and m_{stop} stated in Theorem 1 was also confirmed, see Figures 7 and 8. In addition, we estimated model (12) from the body fat data. The results, which are shown in Table 2, are basically the same as the results obtained from simulation study 1. Note that, despite the large number of simulation runs ($N = 11200$), `nu` does not

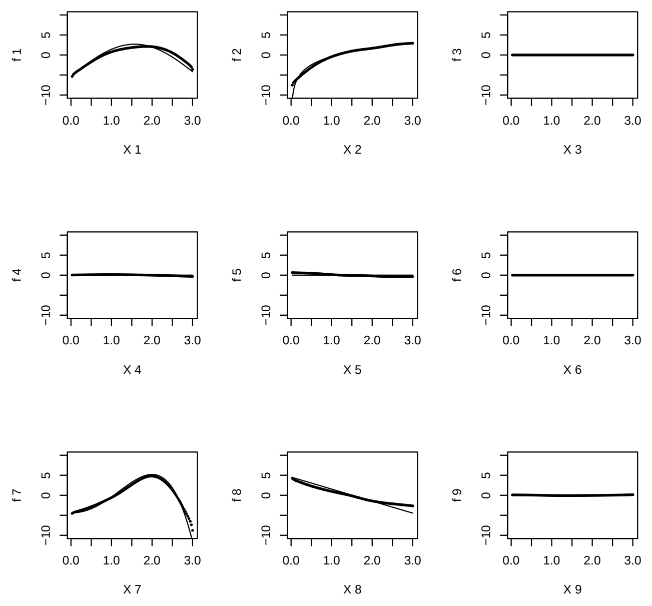

Figure 5: Simulation study 1 - plots of function estimates vs. the data values of $X_1 - X_9$ (training data set #1, $\nu = 0.1$, $P = 50$, df $= 3.5$). The solid lines correspond to the true functions f_1, f_2, f_7, and f_8.

have a significant influence on the predictive performance of PSP (at level $\alpha = 0.05$).

4 Implementation of smoothing spline and P-spline base-learners

In the previous section we have seen that boosting with P-spline base-learners and boosting with smoothing spline base-learners yield very similar prediction results. From a computa-

Table 2: Simulation study 2 - parameter estimates and hypothesis tests of the fixed effects in model (12). A χ^2-test on the hypothesis "$\texttt{df} = \texttt{df}^2 = 0$" resulted in a p-value < 0.001.

Effect	Estimate	Std. Error	z-value	p-value
(Intercept)	9.6489	0.3096	31.1631	< 0.001
df	0.2105	0.0634	3.3225	< 0.001
df^2	0.0083	0.0074	1.1210	0.2623
P	0.0009	0.0004	2.1838	0.0290
$\nu = 0.01$	0.0056	0.0170	0.3309	0.7407

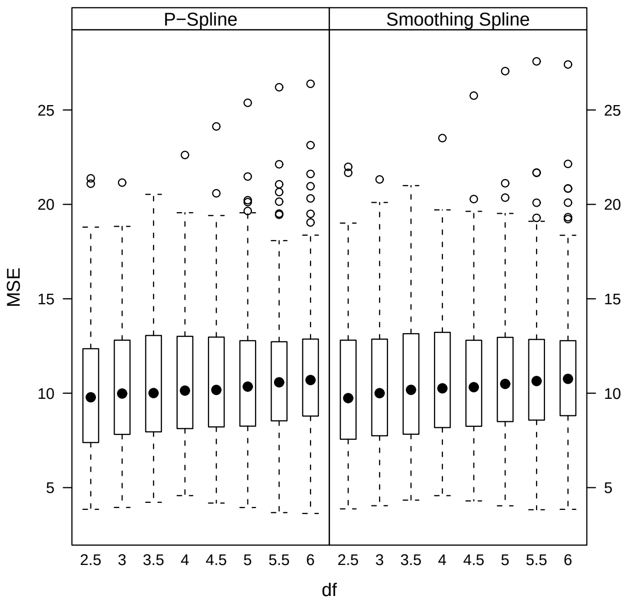

Figure 6: Simulation study 2 - mean squared predictive errors obtained from component–wise FDG with P–spline and smoothing spline base learners ($P = 50$, $\nu = 0.1$).

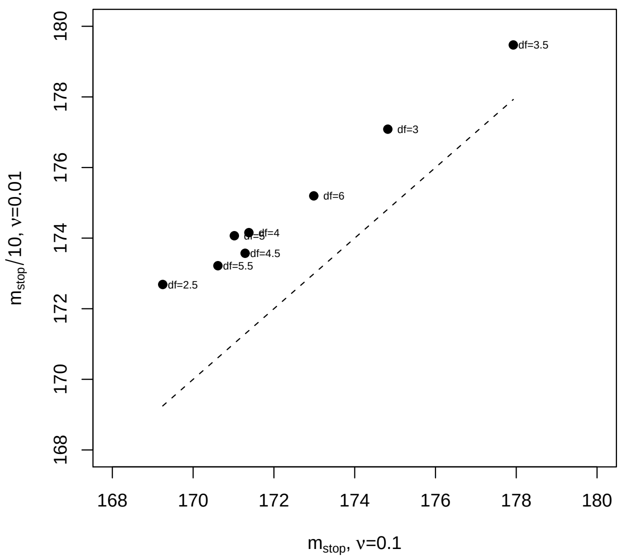

Figure 7: Simulation study 2 - average stopping iterations m_{stop} for step sizes $\nu = 0.1$ and $\nu = 0.01$ (obtained from PSP, $P = 50$).

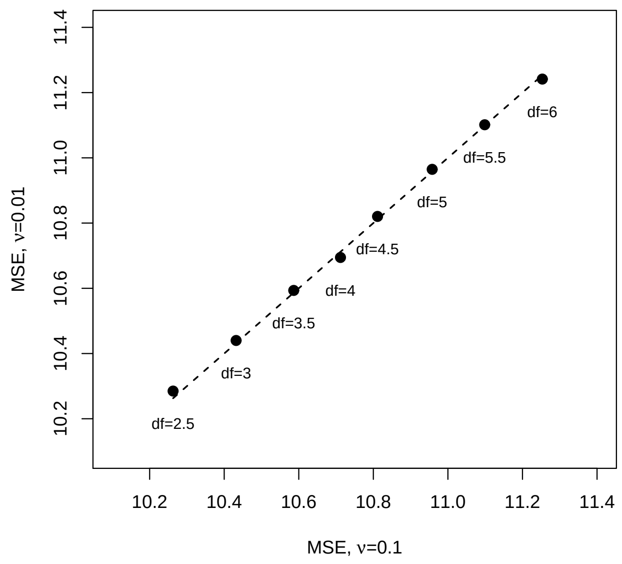

Figure 8: Simulation study 2 - average MSE values for step sizes $\nu = 0.1$ and $\nu = 0.01$ (obtained from PSP, $P = 50$).

tional point of view, P-splines are clearly preferable: In contrast to the integrated squared second derivative penalty of smoothing splines, the P-spline difference penalty is discrete and can easily be evaluated. Also, as pointed out by Eilers and Marx (1996) and Marx and Eilers (1998), the dimensionality of the P-spline penalty is equal to P, while the smoothing spline penalty has a dimensionality equal to n (which is typically much larger than P). For these reasons, using P-spline base-learners will usually be a more efficient strategy than using smoothing spline base-learners.

As a consequence of these results, we have applied the following strategy to implement smoothing spline and P-spline base-learners in R package **mboost**: If smoothing splines are chosen as base-learners, `gamboost()` uses the well-established `stats::smooth.spline()` function. `smooth.spline()` computes the integrated squared second derivative penalty, along with the corresponding smoothing spline fit of the negative gradient.

In case of P-spline base-learners, `gamboost()` uses a similar but improved strategy: In fact, `smooth.spline()` recomputes the values of the basis functions in each iteration of component-wise FGD, as well as the elements of the p penalty matrices. Moreover, in each iteration, `smooth.spline()` converts the value of df into the corresponding values of the p smoothing parameters $\lambda_1, \ldots, \lambda_p$. When considering the component-wise FDG algorithm in Section 2.1, however, it is clear that re-computing these expressions in each iteration is not necessary. Instead, evaluating the basis functions and penalty matrices is only required once at the beginning of component-wise FGD, as well as computing $\lambda_1, \ldots, \lambda_p$

from df. This is exactly the strategy we pursued when implementing P-spline base-learners in `gamboost()`: The hat matrices $\mathcal{H}^1, \ldots, \mathcal{H}^p$ are computed once and are then stored, so that they can be re-used in each iteration. Concerning the conversion of df into $\lambda_1, \ldots, \lambda_p$ (which is needed for computing $\mathcal{H}^1, \ldots, \mathcal{H}^p$), we make use of the following theorem:

Theorem 3. *Consider applying component-wise FDG with P-spline base-learners to a data set with p covariates. Denote by B_j the matrix containing the evaluated B-spline basis functions corresponding to the jth covariate and by D the penalty matrix based on kth order differences. Then the degrees of freedom* df *of the P-spline base-learner, which are equal to the trace of the hat matrix $\mathcal{H}^j = B_j(B_j^\top B_j + \lambda_j D)^{-1} B_j^\top$, can be written as*

$$\begin{aligned} \mathrm{df} &= \mathrm{tr}\big(B_j(B_j^\top B_j + \lambda_j D)^{-1} B_j^\top\big) \qquad (13) \\ &= \mathrm{tr}\left(\mathrm{diag}(d)\left[(\mathrm{diag}(d) + \lambda_j U^\top D V)\right]^{-1}\right) \end{aligned}$$

where $U \cdot \mathrm{diag}(d) \cdot V^\top$ is the singular value decomposition of $B_j^\top B_j$, i.e., U and V are orthogonal matrices and $\mathrm{diag}(d)$ *is a diagonal matrix having the same rank as $B_j^\top B_j$.*

Proof. Define $A := B_j^\top B_j$. We have

$$\begin{aligned} \mathrm{tr}\big(B_j(B_j^\top B_j + \lambda_j D)^{-1} B_j^\top\big) &= \mathrm{tr}\big((B_j^\top B_j + \lambda_j D)^{-1} B_j^\top B_j\big) \qquad (14) \\ &= \mathrm{tr}\big((A + \lambda_j D)^{-1} A\big) \qquad (15) \end{aligned}$$

and

$$\begin{aligned} \mathrm{tr}\left((A + \lambda_j D)^{-1} A\right) &= \mathrm{tr}\left((U \cdot \mathrm{diag}(d) \cdot V^\top + \lambda_j D)^{-1} U \cdot \mathrm{diag}(d) \cdot V^\top\right) \\ &= \mathrm{tr}\left(\left[U^\top (U \cdot \mathrm{diag}(d) \cdot V^\top + \lambda_j D)\right]^{-1} \mathrm{diag}(d) \cdot V^\top\right) \\ &= \mathrm{tr}\left(\mathrm{diag}(d) \cdot V^\top \left[(\mathrm{diag}(d) \cdot V^\top + \lambda_j U^\top D)\right]^{-1}\right) \\ &= \mathrm{tr}\left(\mathrm{diag}(d) \left[(\mathrm{diag}(d) \cdot V^\top + \lambda_j U^\top D) V\right]^{-1}\right) \\ &= \mathrm{tr}\left(\mathrm{diag}(d) \left[(\mathrm{diag}(d) + \lambda_j U^\top D V)\right]^{-1}\right) \, . \qquad (16) \end{aligned}$$

□

By comparing (14) to (16), the benefit of Theorem 3 can be easily seen: Since d can be stored as a vector and since multiplying a matrix with diag(d) is the same as multiplying its columns or rows with the elements of d, the number of elementary multiplications is much smaller when evaluating (16) than when evaluating (14). The singular value decomposition of A can be efficiently computed in R by using the `svd()` function in the R add-on package **base**.

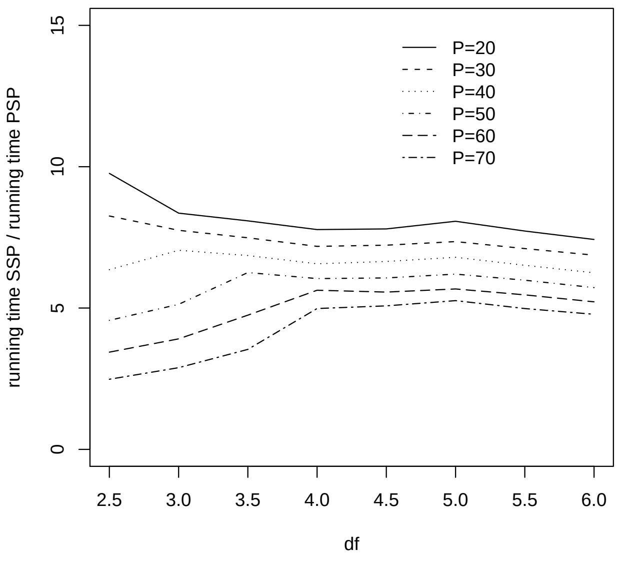

Figure 9: Simulation study 1 - ratio of the average running times of SSP and PSP ($\nu = 0.1$).

Finally, to obtain λ_j from a given value of df, (16) has to be solved numerically for λ_j. An empirical study we have conducted has shown that solving (16) for λ_j is about 30% faster than solving (14) for λ_j.

The benefit of using P-spline base-learners with $k = 2$ instead of smoothing spline base-learners can be seen in Figure 9, which shows the average running times of simulation study 1 ($\nu = 0.1$): Obviously, the running time of SSP using the `smooth.spline()` function is always higher on average than the running time of PSP using the strategy described above. As an example, if $P = 50$, the running time of SSP is about five times larger than the running time of PSP.

5 Concluding remarks

Since their introduction by Eilers and Marx (1996), P-splines have been successfully used in regression as an approximation of smoothing splines. We have shown that this approximation is also successful in a boosting context: By using P-spline base-learners instead of smoothing spline base-learners, the computational effort of component-wise L_2Boosting can be greatly reduced, while there is only a minor effect on the predictive performance of the boosting algorithm.

As P-spline hat matrices can easily be derived, stopping of the component-wise FGD algorithm can also be done very efficiently by means of the corrected AIC criterion. Moreover, by using the AIC-based stopping approach, we have shown that the choice of the step length factor ν is of minor importance: Decreasing ν will only lead to an increase of the number of boosting iterations (in an inversely proportional way) but will not improve the boosting fit. The results of the simulation studies in Section 3 suggest that setting $\nu = 0.1$ is a suitable strategy for obtaining good predictive performance results.

Concerning the choice of P-spline hyper-parameters, we have seen that the number of knots only has a very small effect on the boosting fit. Just as with standard regression problems, choosing 20–50 knots should be a suitable strategy for obtaining a good boosting fit. Moreover, the results of Section 3 suggest that the degrees of freedom of smoothing spline / P-spline base-learners should be small (df ≈ 3.5). By Theorem 2, small values of df correspond to "weak" base-learners, whose favorable properties have been analyzed previously by Bühlmann and Yu (2003).

There is a number of possible extensions of the P-spline approach presented in this paper. First, in order to approximate smoothing splines by P-splines, we kept the degree of the B-spline basis functions and the difference order of the P-spline penalty fixed. These parameters could additionally be altered and could possibly improve the boosting fit in some situations. Second, we exclusively used the B-spline basis in our simulation study. Although B-splines are advantageous due to their numerical stability (cf. Eilers and Marx, 1996, 2004), there are various other approaches on how to construct a spline basis. As an example, Ruppert et al. (2003) used a truncated power series basis for their penalized spline approach. In a boosting context, Bühlmann (2006) used thin plate splines, while Leitenstorfer and Tutz (2007) used a base-learner constructed from radial basis functions. Future work should thus include an investigation on the effect of different spline bases on the performance of L_2Boosting. We finally point out that we exclusively considered Bühlmann and Yu's L_2Boosting algorithm with the *squared error loss*. It is therefore not guaranteed that boosting with other loss functions (such as the negative log-likelihood loss in Binomial and Poisson regression) shows a similar behavior. Although we believe that the results presented in this paper carry over to these loss functions, there is a need for further investigating the use of P-spline base-learners when fitting generalized additive models by means of boosting algorithms.

References

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

Breiman, L., 1999. Prediction games and arcing algorithms. Neural Computation 11, 1493–1517.

Bühlmann, P., 2006. Boosting for high-dimensional linear models. The Annals of Statistics 34 (2), 559–583.

Bühlmann, P., Hothorn, T., 2008. Boosting algorithms: regularization, prediction and model fitting. Statistical Science, accepted.

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

de Boor, C., 1978. A practical guide to splines. Springer, New York.

Eilers, P. H. C., Marx, B. D., 1996. Flexible smoothing with B-splines and penalties (with comments and rejoinder). Statistical Science 11 (2), 89–121.

Eilers, P. H. C., Marx, B. D., 2004. Splines, knots and penalties, unpublished manuscript. URL `http://www.stat.lsu.edu/faculty/marx/splines_knots_penalties.pdf`

Freund, Y., Schapire, R., 1996. Experiments with a new boosting algorithm. In: Proceedings of the Thirteenth International Conference on Machine Learning Theory. San Francisco: Morgan Kaufmann Publishers Inc.

Freund, Y., 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.

Friedman, J. H., 2001. Greedy function approximation: a gradient boosting machine. The Annals of Statistics 29, 1189–1232.

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

Garcia, A. L., Wagner, K., Hothorn, T., Koebnick, C., Zunft, H.-J. F., Tippo, U., 2005. Improved prediction of body fat by measuring skinfold thickness, circumferences, and bone breadths. Obesity Research 13 (3), 626–634.

Green, P. J., Silverman, B. W., 1994. Nonparametric regression and generalized linear models: a roughness penalty approach. Chapman & Hall, London.

Hastie, T., Tibshirani, R., 1990. Generalized additive models. Chapman & Hall, London.

Hothorn, T., Bühlmann, P., Kneib, T., Schmid, M., 2007. mboost: Model-Based Boosting. R package version 0.6-2.
URL `http://R-forge.R-project.org`

Hurvich, C. M., Simonoff, J. S., Tsai, C.-L., 1998. Smoothing parameter selection in non-parametric regression using an improved akaike information criterion. Journal of the Royal Statististical Society, Series B 60, 271–293.

Leitenstorfer, F., Tutz, G., 2007. Knot selection by boosting techniques. Computational Statistics & Data Analysis 51, 4605–4621.

Marx, B. D., Eilers, P. H., 1998. Direct generalized additive modeling with penalized likelihood. Computational Statistics & Data Analysis 28, 193–209.

R Development Core Team, 2007. 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`

Ruppert, D., 2002. Selecting the number of knots for penalized splines. Journal of Computational and Graphical Statistics 11, 735–757.

Ruppert, D., Wand, M. P., Carroll, R. J., 2003. Semiparametric Regression. University Press, Cambridge.

Tutz, G., Binder, H., 2006. Generalized additive modelling with implicit variable selection by likelihood based boosting. Biometrics 62, 961–971.