Generalized Additive Models with Implicit Variable Selection by Likelihood-Based Boosting

G. Tutz and H. Binder

A Discussion, by Robin Evans

Abstract

We examine the GAMBoost method and `R` package of Tutz and Binder (2006), and its effectiveness. Whilst in many examples the algorithm performs relatively well, we find significant difficulties with the approach taken, particularly in terms of computational time, automatic smoothing parameter selection and the claimed 'implicit' variable selection. We also find that GAMBoost performs particularly poorly in cases of high signal-to-noise ratio.

1 Introduction

Generalized additive models (GAMs) were first introduced by Hastie and Tibshirani (1986), as a natural extension of generalized linear models (GLMs). As with GLMs, the data $Y_1, \dots, Y_n$ are assumed to be generated independently from some exponential family, with $\mathbb{E}Y_i = \mu_i = g^{-1}(\eta_i)$ for some smooth, invertible link function g. In the case of GAMs however, the predictor η_i is an additive function of the covariates $x_{i1}, \dots, x_{ip}$ rather than a linear one. That is,

$$\eta_i = f_1(x_{i1}) + \cdots + f_p(x_{ip})$$

for some unknown functions $f_1, \dots, f_p$, usually chosen to be smooth. This clearly leads to a far broader and more flexible class of models, and it is necessary for restrictions to be placed on the functions f_j in order to be able to fit them sensibly to observed data; for instance, if we choose the f_js to be polynomials of large enough degree, we can easily find a curve which passes through all the observations, however it is unlikely to be the optimal method for the purposes of prediction. The usual strategy is to consider each f_j to be a linear combination of some possibly infinite known set of basis functions, $B_1, B_2, \dots$

$$f_j(x) = \sum_{k=1}^{\infty} \beta_{jk} B_k(x),$$

where the β_{jk} coefficients are to be estimated. In general it is not required that each of the functions f_j have the same basis. Even with a finite basis, it can quickly become impossible to identify a model unless the number of covariates is much smaller than the number of observations.

There are various strategies available to deal with this difficulty; Wood (2006) advocates restricting the degrees of freedom available to each covariate to a relatively small value (e.g. 4) and then performing model selection afterwards if necessary.

Tutz and Binder introduce a new procedure, GAMBoost, in their paper *Generalized Additive Modeling with Implicit Variable Selection by Likelihood-Based Boosting*; we will occasionally refer to this as 'the paper', and Tutz and Binder as 'the authors'. The idea behind boosting is to provide a method which can cope with large numbers of possibly uninformative covariates, whilst avoiding over-fitting. GAMBoost uses an approach designed to avoid any fitting of covariates which are uninformative, hence 'implicit variable selection' is carried out. Other methods such as Thin Plate Regression Splines (see for example Wood (2006)) and Back Fitting (Hastie and Tibshirani (1986)) do not shrink parameters completely to zero even when there is no significant evidence of any effect.

We will consider the problem of prediction, and we introduce Tutz and Binder's measure of predictive error.

Definition 1
Suppose that independent observations $(x_1, Y_1), \ldots, (x_n, Y_n)$ from a family of distributions indexed by a parameter θ, are used to fit a model, and that a further set of independent observations are given by $(x_1^*, Y_1^*), \ldots, (x_N^*, Y_N^*)$. Then the *predictive deviance* is given by

$$-2\sum_{i=1}^{N}[l(Y_i^*|\hat{\theta}_i) - l(Y_i^*|\hat{\theta}_i^*)]$$

where $\hat{\theta}_i$ is the parameter estimate from the fitted model, and $\hat{\theta}_i^*$ is the parameter estimate under the saturated model. In the GLM or GAM case, the saturated model sets $\hat{\mu}_i^* = Y_i^*$.

Justification for this loss function can be found in Spiegelhalter et al. (1998) and an example of its use in Knorr-Held and Rainer (2001).

In section 2 we introduce boosting and Tutz and Binder's GAMBoost algorithm, as well as discussing smoothing parameter selection. In section 3 we replicate some of the paper's initial simulations, and in section 4 we replicate the paper's comparison of GAMBoost and other methods. In section 5 we fit the model to some real data, and section 6 contains conclusions and discussion of work in this area since the publication of the paper.

2 Boosting

Boosting has its origins in classification problems from the field of machine learning. The idea, according to Schapire (1990), is to use a *weak learner*, which is an algorithm that improves a model (or classification) by a small amount at each iteration. This is then repeatedly run on different sections of the training data, and the results combined in some form of 'committee vote'. A simple example for a two-group classifier comes from Friedman et al. (2000):

1. Run the weak learner on a subset of the training data of size $k < n$; call the resulting classifier h_1.
2. Run again on a new subset of size k, of which half were misclassified by h_1; call this h_2.

3. Run again on a set of k points which h_1 and h_2 classify differently; call this h_3.

4. Call the boosted classifier $h = \text{MajorityVote}(h_1, h_2, h_3)$.

Schapire shows that such a procedure will always improve the accuracy of the weak learner alone. The term 'boosting' refers to the act of boosting the accuracy of a classification hypothesis. Friedman et al. (2000) show that in the case of a two state classifier with covariates, that a boosting procedure called 'Adaboost' is approximately equivalent to fitting GAMs. This led to the further development of regression boosting procedures—see for example Bühlmann and Yu (2003).

A general description of a regression-type boosting procedure, as described by Bühlmann and Hothorn (2007), is as follows:

1. Take a 'base procedure', which maps weighted data $(X_1, Y_1), \ldots, (X_n, Y_n)$ to a real-valued function estimate $\hat{g}$.

2. Collect some initial weights $\boldsymbol{w}^{(0)}$ (usually $w_i^{(0)} = n^{-1}$ for each i).

3. For each $m = 1, \ldots, M$:

   (i) fit the base procedure to the data with weights $\boldsymbol{w}^{(m-1)}$ to obtain an estimate $\hat{g}^{(m)}$;

   (ii) use the results to obtain new weights $\boldsymbol{w}^{(m)}$, such as up-weighting poorly fitted observations. Weights based on (working) residuals are common.

4. Aggregate the estimates to obtain

$$\hat{f} = \sum_{m=1}^{M} \alpha_m \hat{g}^{(m)}$$

It is easy to see that the structure of the aggregated estimate lends itself naturally to additive functions of the covariates: if the base procedure gives an additive function, then $\hat{f}$ will also be additive. The choice of base procedure, reweighting method, M, and aggregation method are, of course, crucial, and thus the meaning of boosting in a regression context is potentially very broad. The base procedure usually tries to minimise some loss function; in the case of fitting GAMs, it is common to proceed by fitting one covariate at a time.

2.1 Functional Gradient Descent

Suppose that we are working to minimise

$$\mathbb{E}\rho(Y, f(X))$$

over a set of functions $f \in \mathcal{F}$ for some loss function ρ. Then given some data, we can seek to minimise the empirical risk, i.e.

$$n^{-1} \sum_{i=1}^{n} \rho(Y_i, f(X_i)).$$

Functional gradient descent (FGD) is a procedure of the following form, as given by Bühlmann and Hothorn (2007):

1. Choose an offset value $f^{(0)} \equiv c$ to start, such as the value which minimises the empirical risk.
2. For $m = 1, \ldots, M$:
   (i) set
   $$U_i = - \frac{\partial}{\partial f^{(m-1)}(x)} \rho(y, f^{(m-1)}(x)) \Big|_{y=Y_i,\, x=X_i}.$$
   (ii) fit $(X_1, U_1), \ldots, (X_n, U_n)$ using the base procedure, giving $\hat{g}^{(m)}$;
   (iii) update $\hat{f}^{(m)} = \hat{f}^{(m-1)} + \nu \hat{g}^{(m)}$.

The pseudo-residuals U_i are refitted to the base procedure in order to determine what additional information exists in the covariates; this additional explanatory power is then added to the predictor. Justification for this form is given in Appendix B.

Here $0 < \nu \leq 1$ is chosen in such a way as to prevent the algorithm from becoming too 'greedy'. An analogy can be made with surfaces in $\mathbb{R}^n$, where the steepest line of descent may be a curve; in this case $\hat{g}^{(m)}$ is a local linear approximation to this curve, and if we attempt to follow it 'too far' it will significantly diverge from the optimal, curved, path. Instead we do better by taking small steps, and re-approximating the line of steepest descent at each step. $\nu = 0.1$ is a common choice, and the default in `mboost`. If ν is chosen too high, the the algorithm will tend to over-fit; a simulation study in Friedman (2001) illustrates this effect.

In the case of L_2-boosting, which uses the squared error loss, this procedure is equivalent to repeatedly fitting the vector of residuals at each step. In *likelihood-based boosting*, the loss function is minus the log-likelihood, or equivalently the deviance, which enables us to deal with any exponential family of distributions. In spite of the persistence of the name 'boosting' in the context of regression, we feel that 'functional gradient descent' is a more apt description.

2.2 P-splines and Stumps

The two classes of weak learning procedure considered by Tutz and Binder are based upon P-splines and stumps, which we now introduce.

Definition 2
Let $z_1 < \cdots < z_{K+m+2}$ be a series of evenly spaced knots. The ith *B-spline* basis function of order m,

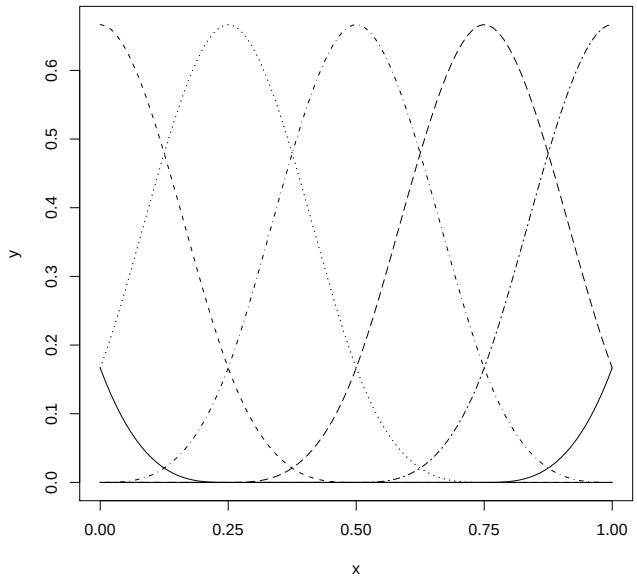

Figure 1: B-spline basis for $m = 2$, $K = 7$, with knots $z_1 = -0.75 < -0.5 < -0.25 < \cdots < 1.75 = z_{11}$.

denoted by B_i^m is a polynomial of degree $m + 1$ on the interval (z_i, z_{i+m+2}), such that B_i^m and its first m derivatives vanish at the end points z_i and z_{i+m+2}. Outside this interval, the function is defined to be zero.

A complete B-spline basis consists of K such functions, all identical except for a shifted domain. The actual interval over which the splines will be evaluated is $[x_{m+2}, x_{K+1}]$ See figure 1 for an illustration.

Fitting then takes the form

$$f(x) = \sum_{i=1}^{K} \beta_i B_i^m(x)$$

This basis has the nice property that each function B_i^m is only locally non-zero, and hence the coefficients are easier to interpret; it may be used to suggest, for example, whether particular ranges of a covariate have any significant effect on the response. The choice $m = 2$ leads us to a cubic spline basis, although this is not how cubic splines are usually presented.

A *P-spline* fit penalises differences in coefficients between neighbouring B-spline basis functions. For example, a first order penalty would be

$$\lambda \sum_{i=1}^{K-1} (\beta_{i+1} - \beta_i)^2.$$

for some $\lambda > 0$. Tutz and Binder fit P-splines by using one step of Fisher scoring on one covariate at a time, which is a form of Newton's method applied to the log-likelihood. The algorithm is derived below, in section 2.3.

An alternative is stumps, which fit piecewise constant functions; the learner begins with a constant predictor $\eta_i^{(0)} \equiv c$, and at each step k chooses a 'split' δ in one of the covariates (say the jth). Then the new predictor is given by

$$\eta_i^{(k+1)} = \eta_i^{(k)} + c_1 \mathbb{1}_{\{x_{ij} \leq \delta\}} + c_2 \mathbb{1}_{\{x_{ij} > \delta\}}.$$

In the likelihood-based boosting case, the choice of j, δ, c_1 and c_2 is made to reduce the deviance by the greatest amount. In order to reduce over-fitting, we can penalise the stumps; Tutz and Binder subtract the penalty $\lambda(c_1 - c_2)^2$ from the log-likelihood being maximised, which encourages shrinkage in the difference between c_1 and c_2.

Although unpenalised stumps may be considered a weak learner, they are not necessarily 'weak enough'; that is to say that they tend to over-fit because they are too sensitive to the data. The penalty has a similar effect to the parameter ν in the functional gradient descent above, and reduces over-fitting. This phenomenon is observed by Bühlmann and Yu (2003), and we will encounter it in section 3.

2.3 GAMBoost

Tutz and Binder (2006) introduce methods for fitting generalized additive models using likelihood-based boosting; any procedure of this form yielding a predictor which is additive in the covariates is called *GAMBoost*. Looking at the form of the regression-type boosting and FGD procedures outlined above, it is clear that if we fit only one covariate at each iteration, then the predictor will indeed be additive in the covariates.

This *componentwise approach* of fitting one covariate at a time can be seen as a form of FGD where the steepest descent along one axis is chosen. This also has an advantage in models which have many covariates, some of which are redundant, since if a covariate has very little information the procedure is discouraged from fitting it at all. This is the heart of the implicit variable selection in GAMBoost. Tutz and Binder implement this procedure in the `R` package `GAMBoost`, using P-splines with 20 evenly spaced knots. They choose to use a penalty based on second order differences, with a parameter λ.

Since we are dealing with a simple exponential family, we can write the log-likelihood in the form

$$l(\theta, \phi \mid y) = \phi^{-1}(y\theta - b(\theta)) + c(y, \phi).$$

The next result establishes the GAMBoost algorithm.

Lemma 1
Suppose that $Y_1, \ldots, Y_n$ are independent observations from an exponential family and $\mathbb{E}Y_i = g^{-1}(\boldsymbol{z}_i^T \boldsymbol{\beta}) = \mu_i$ for an $n \times p$ matrix of covariates Z and a p-dimensional vector of parameters $\boldsymbol{\beta}$. Suppose further that the dispersion parameter ϕ is known. Then one Fisher scoring iteration on the penalised likelihood amounts to

$$\hat{\beta}_{\text{new}} = (Z^T \hat{W} Z + \lambda \Lambda)^{-1} Z^T \hat{W} \hat{D}^{-1} (\boldsymbol{y} - \hat{\boldsymbol{\mu}}),$$

where Z is the model matrix, $\hat{D} = D(\hat{\boldsymbol{\mu}})$ is a diagonal matrix with entries $g'(\hat{\mu}_i)^{-1}$ and $\hat{W} = W(\hat{\boldsymbol{\mu}})$ is a diagonal matrix of weights

$$\hat{W}_{ii} = \frac{\hat{D}_{ii}^2}{\phi b''(\hat{\theta}_i)}.$$

Proof. See appendices. □

Remark 1
It is clear from the proof of Lemma 1, that the result extends to the Gaussian case even if the dispersion parameter is unknown, because of the orthogonality of ϕ and θ.

Then using the above result, we can establish the GAMBoost algorithm for P-splines given in the paper.

1. Fit an intercept model $\eta_i = \eta^{(0)}$ by maximum likelihood estimation.
2. For each $m = 0, 1, \ldots, M-1$
   (i) For each covariate $s = 1, \ldots, p$, compute
   $$f_{s,\text{new}} = Z_s \hat{\beta}_{s,\text{new}} = Z_s (Z_s^T \hat{W} Z_s + \lambda\Lambda)^{-1} Z_s^T \hat{W} \hat{D}^{-1} (\boldsymbol{y} - \hat{\boldsymbol{\mu}}),$$
   where each of $\hat{W}, \hat{D}$ and $\hat{\boldsymbol{\mu}}$ are computed at step m, and Z_s is the matrix of parameters relating to the sth covariate.
   (ii) For each covariate $s = 1, \ldots, p$: compute the deviance reduction
   $$\text{reduction}(s) = \text{Dev}(\hat{\eta}_{(m)}) - \text{Dev}(\hat{\eta}_{(m)} + f_{s,\text{new}}).$$
   Choose $j = \arg\min_s \text{reduction}(s)$, the covariate which maximises this reduction in the deviance.
   (iii) Update the linear predictor using the jth covariate: $\hat{\eta}_{(m+1)} = \hat{\eta}_{(m)} + f_{j,\text{new}}$.

The algorithm for stumps is very similar, but replaces step 2 with:

2. For each $m = 0, 1, \ldots, M-1$
   (i) Define
   $$(c_1,\ c_2)^T = \boldsymbol{c}(s, \xi) = (Z_{s,\xi}^T \hat{W} Z_{s,\xi} + \lambda\Lambda)^{-1} Z_{s,\xi}^T \hat{W} \hat{D}^{-1} (\boldsymbol{y} - \hat{\boldsymbol{\mu}}),$$
   where $Z_{s,\xi}$ is an $n \times 2$ matrix with ith row $(\mathbb{1}_{\{x_{is} \le \xi\}},\ \mathbb{1}_{\{x_{is} > \xi\}})$, and $\Lambda = \left(\begin{smallmatrix} 1 & -1 \\ -1 & 1 \end{smallmatrix}\right)$.
   (ii) Set $f_{s,\xi,\text{new}} = c_1 \mathbb{1}_{\{x_{is} \le \xi\}} + c_2 \mathbb{1}_{\{x_{is} > \xi\}}$, and then
   $$\text{reduction}(s, \xi) = \text{Dev}(\hat{\eta}_{(m)}) - \text{Dev}(\hat{\eta}_{(m)} + f_{s,\xi,\text{new}}).$$
   Choose $(j, \delta) = \arg\min_{s,\xi} \text{reduction}(s, \xi)$, the covariate and split which maximise this reduction in the deviance.
   (iii) Update the linear predictor using the jth covariate with a split at δ: $\hat{\eta}_{(m+1)} = \hat{\eta}_{(m)} + f_{j,\text{new}}$.

2.4 Choosing λ

The next two subsections concern the two parameters which must be chosen in GAMBoost: the smoothing parameter λ, and the number of steps used, M. The two choices are interdependent, since if we increase λ, the fitting performed at each step will be more conservative, and so we will need to increase M as well in order to arrive at the best fit.

Choosing λ is, as we will see, crucial to ensuring that the procedure behaves sensibly. Tutz and Binder reason that the number of steps used to fit the model should be sufficiently large so as to prevent the algorithm being too greedy; further they state that if M is too large, then the procedure will simply take too long. They also claim that choosing a perfectly optimal λ does not improve the model enough to make it a priority, and choosing λ coarsely is sufficient; our empirical investigations appeared to confirm this, although it is hardly a satisfactory solution.

The authors therefore recommend choosing λ so as to ensure that the number of steps taken is between 50 and 200. The ad hoc nature of this choice turns out to be a problem, as we see in section 4. To implement this, `GAMBoost` uses a *coarse line search*, which proceeds approximately as follows. An initial value for the smoothing parameter λ_1 is chosen, and set $k = 1$.

1. The fitting procedure is run for $M = 200$ iterations with $\lambda = \lambda_k$, and each fit between 50 and 200 is examined.
2. Let $m \in \{50, 51, \ldots, 200\}$ be the iteration with the best observed fit; the authors use AIC to decide, see section 2.5 below.
   (i) if m lies well between 50 and 200, then $\lambda = \lambda_k$ is considered acceptable, and we stop.
   (ii) if m is very close to 200, then λ_k is assumed to be too large, and we take $\lambda_{k+1} = \lambda_k/2$, increase k by 1 and return to step 1.
   (iii) if m is very close to 50, then λ_k is assumed to be too small, and we take $\lambda_{k+1} = 2\lambda_k$, increase k by 1 and return to step 1.

In the event that we 'overshoot' (i.e. λ_k is too small and λ_{k+1} is too large or vice-versa), then we move in the opposite direction more conservatively. The default in `GAMBoost` is to take $\lambda_1 = 500$, and m is considered 'very close' to the upper-limit M if $m > 0.95M$.

2.5 Stopping Criteria

As we shall see in section 3 the number of fitting iterations is also crucial to how well the model will fit. Tutz and Binder consider two stopping criteria: AIC and cross-validation.

AIC tries to minimise the deviance subject to a penalty based on the number of degrees of freedom in the model. In order to calculate the approximate degrees of freedom associated with a fit, we consider the *hat matrix*, defined by

$$\hat{\boldsymbol{\mu}} = H\boldsymbol{y}.$$

Then the effective degrees of freedom is

$$\mathrm{df} = \mathrm{tr}(H) = \sum_{i=1}^{n} \frac{\partial \mu_i}{\partial y_i},$$

i.e. the total amount of sensitivity in the fitted values to changes in the observations. This is consistent with the theory of linear models. In general the relationship between $\hat{\boldsymbol{\mu}}$ and $\boldsymbol{y}$ is not linear, so we need some approximation $\hat{\boldsymbol{\mu}} \approx \tilde{H}\boldsymbol{y}$.

Lemma 2
An approximate expression for the hat matrix H after m boosting iterations is given by

$$H_m = L_0 + \sum_{j=1}^{m} L_j(I - L_{j-1}) \cdots (I - L_0),$$

for $n \times n$ matrices $L_0, \ldots, L_m$.

Proof. See appendices for proof and definition of matrices L_i. □

The accuracy of this approximation is not formally discussed by the authors, but it appears to depend upon the sum of squared differences between fits at successive iterations; thus the accuracy should increase if smaller steps are taken. The AIC is defined by

$$\mathrm{AIC} = \mathrm{Dev}\ \hat{\boldsymbol{\eta}}_{(m)} + 2\mathrm{df}_m,$$

and hence approximated using Lemma 2, $\mathrm{df}_m \approx \mathrm{tr}(H_m)$. With this result we can fit the model over many boosting steps, and then look to see which number of steps minimises the AIC.

An alternative, and in our view more natural approach, is to use cross-validation. A K-fold cross-validation approach divides the data randomly and as evenly as possible into K groups. The model is then fitted to the data with each group removed in turn, and then omitted data is used to judge the quality of the fit; in particular Tutz and Binder's `GAMBoost` uses the predictive deviance by default. The mean of the predictive deviances at each iteration is calculated across the K 'folds', and the step which minimises this average is selected as the stopping point of the full model.

The difficulty with this approach is computational cost: the accuracy of the method increases with K, with $K = n$ being the usual 'leave one out' cross-validation, but the computational complexity also increases. An advantage is that it is also possible to experiment with different smoothing parameters, but again this slows down the procedure dramatically.

2.6 Standard Deviations

The following result can be used to derive approximate confidence bands for GAMBoost fits.

Lemma 3
Let $\hat{f}_j^{(m)}$ be the fitted estimate of f_j after m iterations. Then

$$\begin{pmatrix} \hat{f}_j^{(m)}(x_{1j}) \\ \vdots \\ \hat{f}_j^{(m)}(x_{nj}) \end{pmatrix} \approx Q_{m,j}\boldsymbol{y}$$

for some matrix $Q_{m,j}$.

Proof. Proof and definition of $Q_{m,j}$ given in appendices. □

We then use $\operatorname{Cov} \hat{f}_j^{(m)} \approx \operatorname{Cov}(Q_{m,j}\boldsymbol{y}) = Q_{m,j} \operatorname{diag}(\phi b''(\theta_i)) Q_{m,j}^T$. The authors report that a small simulation study, not produced in the paper, suggested that the approximate confidence bands behaved well; we discuss this further in section 6.

3 Exploratory Simulation Study

Tutz and Binder perform various simulations in order to compare the effectiveness of GAMBoost to other methods. In all simulations, data are generated as follows: for each observation in a sample of size n, the p covariates $x_{i1}, \ldots, x_{ip}$ are generated as independent uniform random variables on $[-1, 1]$. Then the Y_i are generated as independent random variables from a simple exponential family with canonical link (either Poisson or binary) with predictor $g(\mu_i) = \eta_i$.

3.1 Stumps or splines?

The first simulations are designed to compare the properties of using P-splines and stumps as weak learners. We take $n = 100$ and $p = 5$, and generate from a binary distribution with an additive predictor

$$g(\mu_i) = c(-0.7 + x_{i1} + 2x_{i3}^2 + \sin(5x_{i5})). \tag{1}$$

In this case $c = 3$. GAMBoost was applied to these data with smoothing parameter $\lambda = 30$ for P-splines and $\lambda = 2$ for stumps; the number of boosting steps used is not mentioned, so we use $M = 30$ for stumps, and $M = 100$ for P-splines, which gives visually similar results to the paper's*. Fits for each of the five covariates, both methods and two separate data sets are given in figure 2; these should be compared to figure 1 in the paper. The first data set, shown in the top half of the diagram, gives a very similar fit to that illustrated by Tutz and Binder: both methods track the first and third covariates closely, and do not over-fit the uninformative second and fourth covariates. Both slightly miss the peak and trough of the sine wave in the fifth covariate, because of the inherent smoothing.

However, the second data set (bottom half) shows a different picture; here the stumps perform very poorly, incorrectly assigning values close to -2 for both uninformative covariates, and failing to model the first and fifth covariates well at all. P-splines continue to work well, which suggests that the data is not unusual. The experiment being replicated here is rather ad hoc, and yields mixed results.

A further simulation was carried out to assess how mean square error (MSE) is related to the number of boosting steps. Data was generated from the same model, and P-splines, penalised stumps and unpenalised stumps are fitted; the mean square error between the predicted $\eta(\boldsymbol{x})$ and the fitted $\hat{\eta}(\boldsymbol{x})$ was then calculated at each step.

$$\text{MSE} = 2^{-p} \int (\eta(\boldsymbol{x}) - \hat{\eta}(\boldsymbol{x}))^2 \, d\boldsymbol{x}.$$

*Since the smoothing parameters have been chosen differently and apparently arbitrarily, there seems to be no reason to make the number of boosting steps the same for the two methods. One can count the number of splits in Tutz and Binder's figure 1 to come up with the figure of around 30 steps for stumps.

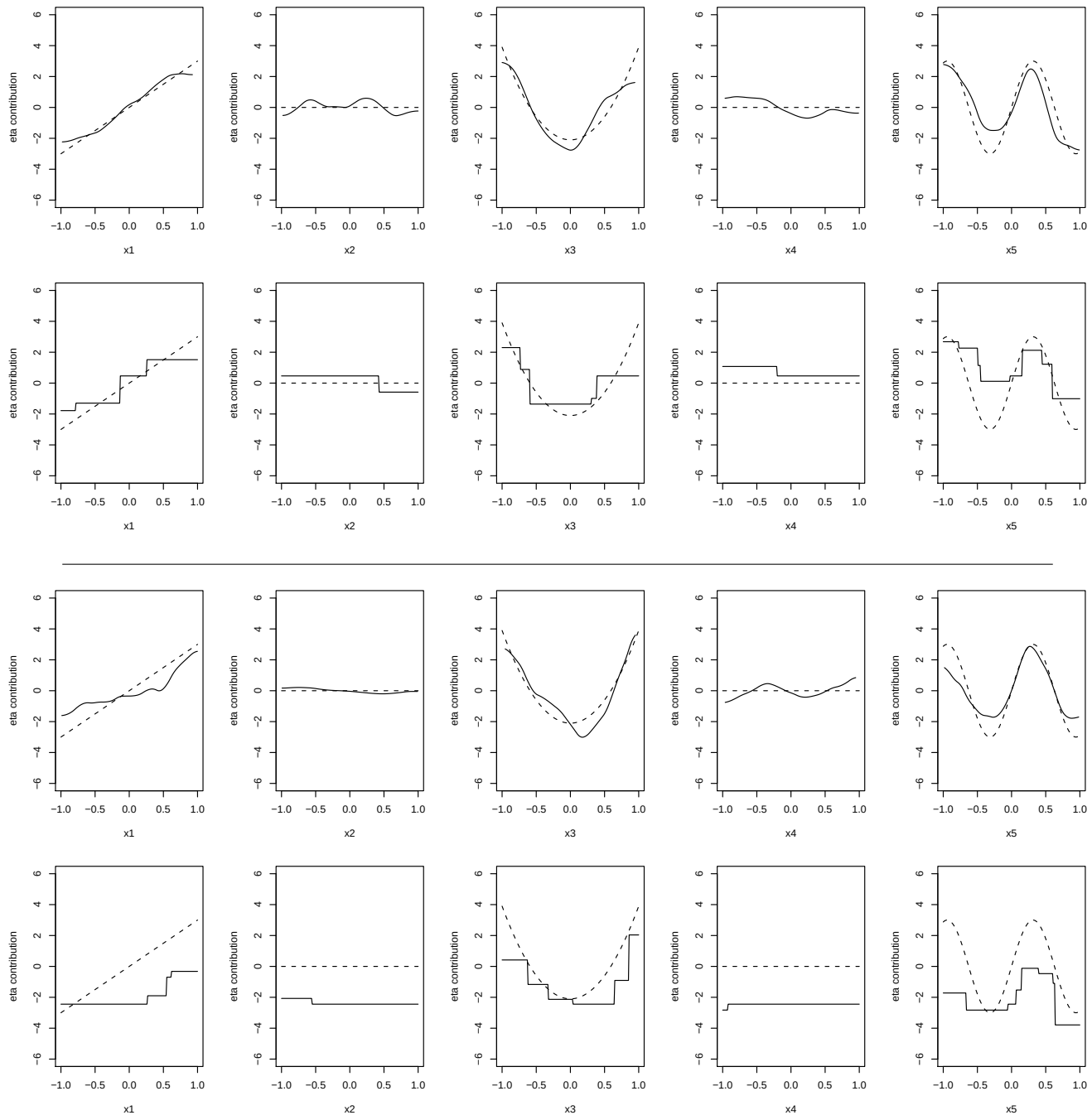

Figure 2: Comparison of fits for two data sets (top 10 panels are one data set, bottom 10 the other) with P-splines (first and third rows) and penalised stumps (second and fourth rows). Dashed lines indicate true values.

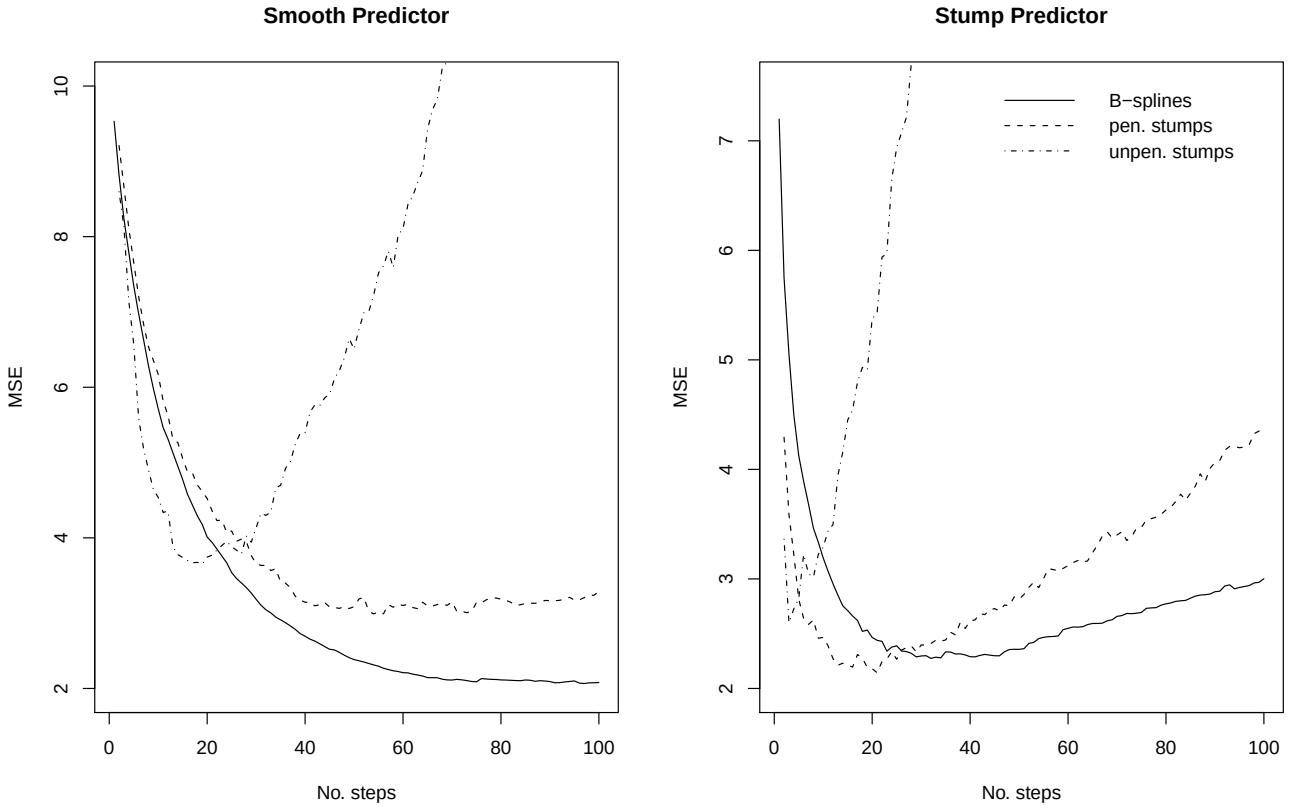

Figure 3: Mean curves of mean square error for P-splines (solid line), penalised stumps (dashed) and unpenalised stumps (dot-dashed) at different numbers of boosting steps. Left panel data uses smooth true predictor, right panel uses piecewise constant predictor. All based upon 20 repetitions.

This was repeated 20 times, and the mean MSE over these 20 realisations recorded. The results are found in the first panel of figure 3, and should be compared to figure 2 of Tutz and Binder. The results are entirely consistent with those in the paper. The lack of smoothness in our reproduced curves is due to how the MSE was calculated: we used the `R` function `adapt` to integrate over the five-dimensional surface, and in order to obtain a reasonable running time the relative error tolerance was set at 0.5, which is quite high.

The right hand panel from figure 3 is generated very similarly, but uses a piecewise constant predictor to favour stumps: let $K(x) = 2\mathbb{1}_{\{x \geq 0\}} - 1$, then the predictor is given by

$$c(0.5K(x_{i1}) + 0.25K(x_{i3}) + K(x_{i5})). \tag{2}$$

From these results we see that while unpenalised stumps converge more quickly to the correct solution initially, they quickly become over-fitted, and do not reach a minimum as low as the other two methods; this is what we would expect, since it moves greedily to reduce the deviance at each step. Further, P-splines fit better than stumps in the smooth case, and are less vulnerable to over-fitting. As we might suspect, stumps perform better than splines in terms of minimum MSE when the predictor is itself piecewise constant. Based on these two experiments, Tutz and Binder conclude that P-splines are a better approach when the underlying predictors are believed to be smooth, both because the mean square error

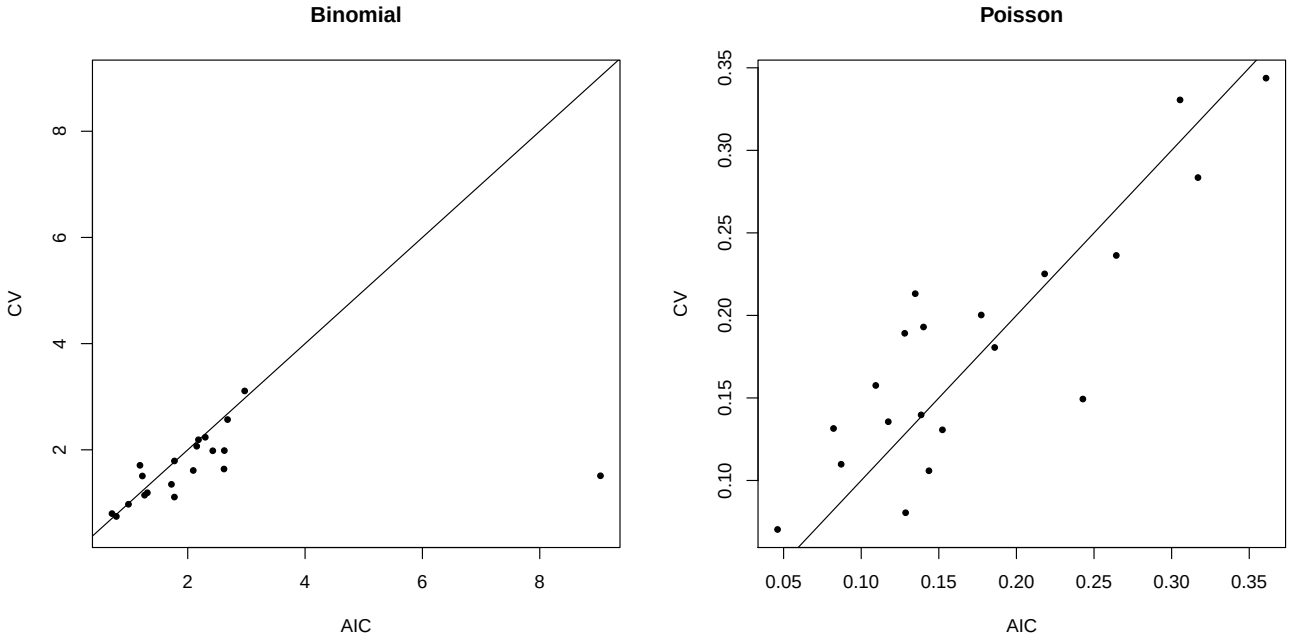

Figure 4: Mean square error of model fitted using cross-validation plotted against that of model fitted using AIC. 20 repetitions were used to both binary data (left panel) and Poisson (right).

appears to be smaller, and because the fits are more visually pleasing. Intuitively, it also makes sense to use a smooth estimator for a smooth function.

3.2 Stopping Criteria

To compare the relative merits of cross-validation and AIC as stopping criteria, the paper includes a short simulation. Data were generated using the predictor (1) with $n = 100$ and $p = 5$, both for the binary case with $c = 2.5$ and the Poisson case with $c = 1$. In the paper it states that λ was chosen to ensure that the number of steps used is similar for both cross-validation and AIC procedures; we use the default value $\lambda = 100$ for the cross-validation, and Tutz and Binder's coarse line search approach (see below) for AIC. 5-fold cross-validation was used, and the MSE over the two approaches recorded for 20 separate realisations of the data. The results are shown in our figure 4, which should be compared to figure 3 in the paper.

Contrary to Tutz and Binder's findings, we see almost no difference between AIC and cross-validation for the Poisson model, and indeed one example where AIC does very poorly for the Binomial model. The authors suggest that the relative consistency of the mean square error under the AIC suggests that the approximation given in Lemma 2 is a good one; we do not follow this argument. We caution that the MSE in our figure 4 was calculated with the same high relative error discussed for figure 3, but this does not explain the extremely large value seen in one case; it may be the result of numerical instability in `adapt`.

4 Comparison to Alternative Methods

The authors provide two larger simulations to compare the relative merits of GAMBoost to better known fitting procedures. The various models fitted and compared were:

Base. This was simple a GLM with only a constant term, used as a baseline measure for comparison purposes. This was reproduced using the `R` function `glm`. The value of the constant c is supposedly chosen so that the full GLM model was able to improve significantly upon this baseline.

GLM. The full GLM model using all p covariates. Again, this was easily reproduced using `R`.

GLMse. The full GLM model was fitted, but then subjected to step-wise selection using AIC. The `R` function `stepAIC` facilitated this. As in the paper, the degrees of freedom in the model were multiplied by 1.4 to conform with wGAM (see below).

bfGAM. Hastie and Tibshirani's back fitting algorithm for GAMs. This is easily implemented using the function `gam` in the package `gam`. Degrees of freedom also multiplied by 1.4.

wGAM. Fitting of GAMs using P-splines with a first order penalty and smoothing parameter chosen by cross-validation. The effective degrees of freedom are scaled by a factor of 1.4 as suggested by Wood (2006); this modification was observed by Kim and Gu (2004) to eliminate occasional over-fitting, though it has little theoretical justification.

wGAMfb. Wood's model as above, but with stepwise selection applied afterwards. It was unclear how this procedure was implemented, and so we were unable to assess this model.

GAMBoost. Tutz and Binder's model with a coarse line search. This was implemented using their own `GAMBoost` package and the function `optimGAMBoostPenalty`.

Optimal. GAMBoost with an optimal number of boosting steps, as chosen by K-fold cross-validation. K was unspecified, so we used the default value 10 from the function `cv.GAMBoost`.

The precise `R` code we use for each model is presented in the appendices.

In each part, the models were trained on $n = 100$ complete data points; then, a further $N = 1000$ observations (x_i^*, Y_i^*) were generated in the same manner, and the fitted models were used to predict the outcomes using the covariates only. The quality of their prediction was assessed using the predictive deviance. In the binary case, this is

$$\begin{aligned}
\mathrm{Dev}(Y_i^*; \hat{Y}_i) &= 2(Y_i^* \log Y_i^* + (1 - Y_i^*) \log(1 - Y_i^*) - Y_i^* \log \hat{Y}_i - (1 - Y_i^*) \log(1 - \hat{Y}_i)) \\
&= 2Y_i^* \log\left(\frac{Y_i^*}{\hat{Y}_i}\right) + 2(1 - Y_i^*) \log\left(\frac{1 - Y_i^*}{1 - \hat{Y}_i}\right) \\
&= -2Y_i^* \log \hat{Y}_i - 2(1 - Y_i^*) \log(1 - \hat{Y}_i)
\end{aligned}$$

where the last simplification follows from the fact that Y_i^* is either 0 or 1; here $\hat{Y}_i$ is the fitted probability of the response being 1. In the Poisson case,

$$\begin{aligned}
\mathrm{Dev}(Y_i^*; \hat{\mu}_i) &= 2(Y_i^* \log Y_i^* - Y_i^* - Y_i^* \log \hat{\mu}_i + \hat{\mu}_i) \\
&= 2Y_i^* \log\left(\frac{Y_i^*}{\hat{\mu}_i}\right) + 2(\hat{\mu}_i - Y_i^*)
\end{aligned}$$

where $\hat{\mu}_i$ is the fitted mean.

4.1 Linear Predictors

In the first part, data were generated from exponential families via a linear function of the covariates, so a GLM ought to be an adequate model; the purpose of this was to check that GAMBoost does not over-fit models where simple relationships are the correct ones. Data were generated as in section 3, but with predictor

$$g(\mu_i) = c(x_{i1} + 0.7x_{i3} + 1.5x_{i5}), \tag{3}$$

where $c = 2$ in the binomial case, and $c = 0.5$ in the Poisson; $p = 5$ and $p = 10$ were both tested. There were 50 repetitions of each experiment, and in each case the mean predictive deviance over the 1000 predictions was recorded. The results are shown in figure 5, where each boxplot shows the spread of the 50 repetitions; this should be compared to figure 4 from the paper.

We find that the results for the Base, GLM, GLMse, GAMBoost and Optimal models are all completely comparable with those given by Tutz and Binder. Our results give noticeably worse predictions for the other GAM fitting methods, which we assume is due to choosing different settings from the authors. The results suggest that GAMBoost performs very well, and in particular does no worse than GLM, which should be the 'correct' procedure given the true model. There is no evidence of GAMBoost over-fitting, whereas bfGAM and wGAM both appear to over-fit, especially in the $p = 10$ Binomial case.

However, we encounter significant problems when c is increased. Repeating the same experiment with $c = 2$ in the Poisson case, which increases the signal-to-noise ratio in the model, we find that GAMBoost performs extremely badly. In many, although not all examples, it is observed to massively over-fit the data, resulting in a model which does not predict well at all. The appropriate remedy appeared to be to increase the upper limit on the number of steps allowed from its default of 200, and use a high value of λ; however in order to achieve this, the time taken to fit the model becomes very large. In the worst cases observed, increasing the number of steps to 5000 gave a good fit, but required around 10 minutes computing time.

4.2 Additive Model

For their main simulation, the authors return to model (1), using $p = 3, 10, 50$, with $c = 1, 2, 3$ for the binomial case, and $c = 0.5, 0.75, 1$ for the Poisson case. They give results for both $n = 100$ and $n = 400$, but the latter case proved to be far too slow for reproduction[†] (when fitted using their `GAMBoost` package). Again there were 50 repetitions of each experiment; the mean over the 50 repetitions is recorded in table 4.2, and should be compared to table 1 from the paper. GLMse, bfGAM and wGAM could not be fitted to the $p = 50$ case, as the fits were either intractable or had too many degrees of freedom.

There were also problems of non-convergence in the binary case—if fitted probabilities are too close to 0 or 1 we exclude them because it results in much larger deviances for wGAM and bfGAM. The number of times this happened is recorded in table 4.2.

The results are broadly comparable to those obtained by Tutz and Binder, although the GAMBoost deviances for binomial data with $p = 3$ are higher than expected; we can think of no satisfactory explanation for this. The Optimal fit seems to behave as expected, however. GAMBoost appears to fit as well as bfGAM and wGAM in the $p = 3$ and $p = 10$ cases, and has the advantage of being able to make sensible fits in the $p = 50$ case, which the other two methods cannot do without some adjustment. Simple GLM

[†]The entire experiment for the $n = 100$ case ran in approximately 24 hours, whereas in the $n = 400$ case, only one of the eighteen cases was finished in this time.

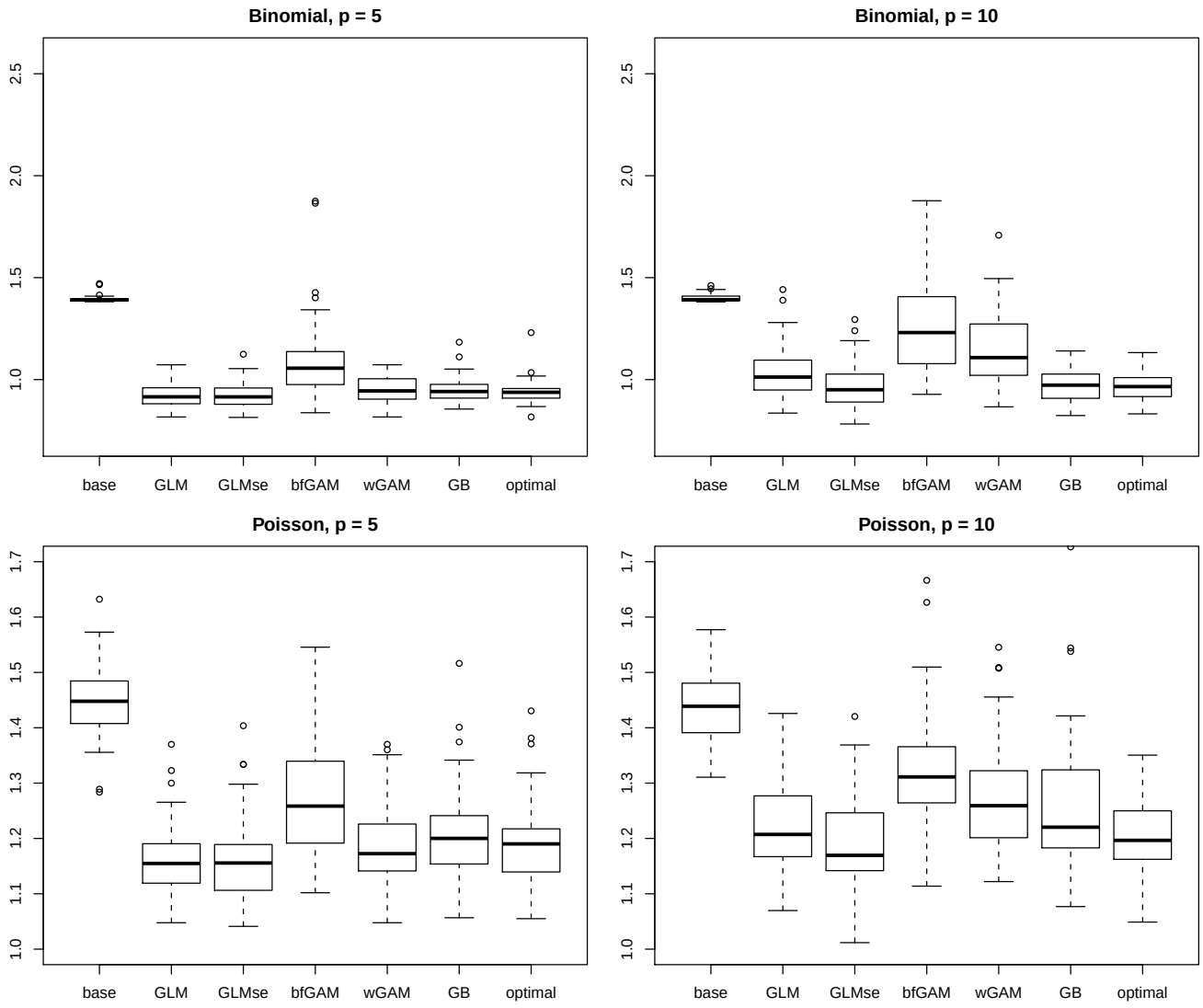

Figure 5: Mean predictive deviance for 50 repetitions over various models and methods. Here GB = GAMBoost and other methods are explained above.

Family	c	p	Base	GLM	GLMse	bfGAM	wGAM	GAMBoost	Optimal
Bin	1	3	1.40	1.36	1.36	1.32	1.30	1.40	1.28
		10	1.39	1.47	1.41	1.50 (1)	3.06	1.40	1.35
		50	1.40	5.05 (40)	–	–	–	1.61	1.41
	2	3	1.39	1.30	1.31	1.02 (1)	1.02	1.92	1.00
		10	1.40	1.39	1.34	1.24 (1)	8.27	1.07	1.06
		50	1.39	5.35 (42)	–	–	–	1.31	1.27
	3	3	1.39	1.28	1.28	0.81	1.90	1.71	0.82
		10	1.40	1.40	1.34	0.92	10.38 (3)	0.90	0.89
		50	1.39	4.50 (45)	–	–	–	1.06	1.05
Pois	0.5	3	1.44	1.38	1.38	1.26 (2)	1.22	1.32	1.25
		10	1.44	1.51	1.45	1.34	1.39	1.48	1.35
		50	1.46	3.78	–	–	–	1.75	1.42
	0.75	3	1.95	1.78	1.77	1.21	1.19	1.32	1.21
		10	1.93	1.94	1.89	1.30 (1)	1.40	1.57	1.39
		50	1.94	5.91	–	–	–	2.01	1.58
	1	3	2.92	2.48	2.48	1.20	1.16	1.32	1.20
		10	2.87	2.77	2.73	1.27	1.31	1.58	1.36
		50	2.88	16.01	–	–	–	2.24	1.69

Table 1: Mean predictive deviance over 50 repetitions for various model fits, and various methods of data generation. Brackets indicate the number of times convergence failed in each example.

performs poorly due to over-fitting in the $p > 3$ cases. Overall, the results show the benefits of GAMBoost in being able to handle large numbers of uninformative covariates.

4.3 Computational Cost

The time taken to fit complex models is a concern, as shown in table 2. Although GAMBoost takes a comparable amount of time to wGAM and bfGAM for the $n = 100$ case, its complexity appears to increase rapidly for larger n; in contrast, none of the other methods seem significantly to depend upon n[‡]. Apparently paradoxically, the Optimal method does not seem to slow down with larger n; we suspect that this is because as n increases, the default settings for the smoothing parameter in GAMBoost become unhelpful, and thus the coarse line search is invoked many more times to choose a sensible value. Further work to establish a good initial 'guess' for λ, varying with n, might be able to resolve this problem.

5 Bodyfat Data

Tutz and Binder analyse some psychiatric data, which we were unable to obtain. Instead we look at the 'bodyfat' data, found in the package `mboost`, and analysed by Garcia et al. (2005). It consists of $p = 9$ covariates from $n = 71$ women, as well as body fat percentage measured using Dual Energy X-Ray Absorptiometry, which is accurate but very expensive; the covariates are age and eight anthropometric measurements. The question of interest is to create a model which is simple and accurately predicts body

[‡]The observed decrease in, for example, wGAM between $n = 100$ and $n = 200$ is almost certainly due to the small sample size used in this experiment (10 fits).

n	**Base**	**GLM**	**GLMse**	**bfGAM**	**wGAM**	**GAMBoost**	**Optimal**
100	0.0084	0.0164	0.273	16.2	40.0	21	369
200	0.0092	0.0172	0.331	13.2	12.8	140	57
400	0.0108	0.0216	0.488	18.3	8.0	491	120
800	0.0144	0.0269	0.779	26.6	10.9	3780	350

Table 2: Time in seconds to fit model to various methods and sizes of data set. Each value is an average of 10 fits to binary responses using model (1) with $p = 10$, $c = 2$.

fat from the easily measured covariates. These data are comparable to Tutz and Binder's data as they were also trying to fit a predictive model with $n = 59$ and $p = 13$.

Exploratory plots of the bodyfat data (not shown) make it clear that the covariates are highly dependent, and so the aim is to keep the model simple, and not introduce covariates which provide little additional information. From a healthcare provider's perspective, a quick and simple way of measuring body fat, perhaps from three or four measurements is desired. Garcia et al. suggest a linear model using three covariates. We will attempt to fit GAMs with Gaussian responses.

GAMBoost with 10-fold cross-validation fits 8 of the 9 covariates; the contributions of each covariate are shown in figure 6. This presents a problem, since we do not want a model which uses this many covariates. We could use the confidence bands to try and eliminate variables which are not significant, but this seems to defeat the 'implicit variable selection' claimed to be part of GAMBoost. The age covariate in particular does not appear to be significant, and other model fits corroborate this, but it is selected for two iterations. In Tutz and Binder's example, 12 of the 13 covariates are fitted, but the confidence bands shown in the paper suggest that many are not significant for a large proportion of their range.

This may be a fundamental problem with having such a small sample size; we ought to be suspicious of any model with $n = 59$ which uses 12 covariates, especially when the covariates are certainly highly dependent, which is the case in both data sets. The curse of dimensionality suggests that we should not expect to get much information, and using a class of curves as broad as cubic splines may cause the model to imply nuance which does not really exist.

As an alternative using `gam` from package `mgcv`, we try fitting the data with a P-spline basis and second order penalisation. Setting the basis dimension of the P-splines to 5 allows us to fit all the covariates, and we use the correction factor of 1.4 for degrees of freedom. Using the standard summary table, we are then easily able to identify candidate covariates to remove.

The approximate procedure we use is as follows: if the effective degrees of freedom (e.d.f.) are very close to 1, this suggests that this covariate really only has a linear effect, and so we try refitting it as a linear term. If the p-value indicated for the linear term is not significant, then we remove the covariate altogether. To choose between covariates with e.d.f. close to 1, we look at the approximate p-values provided and remove the covariate whose p-value is highest[§]. Proceeding in this manner leads to the removal of 4 covariates, and keeps only linear terms of 2 more; this much more parsimonious model explains 94.3% of the total deviance, compared to the original model's 94.8%.

GAMBoost does not have such a simple method for eliminating unnecessary covariates.

[§]We do not use this p-value for model selection directly because, as Wood cautions, it is only *very* approximate.

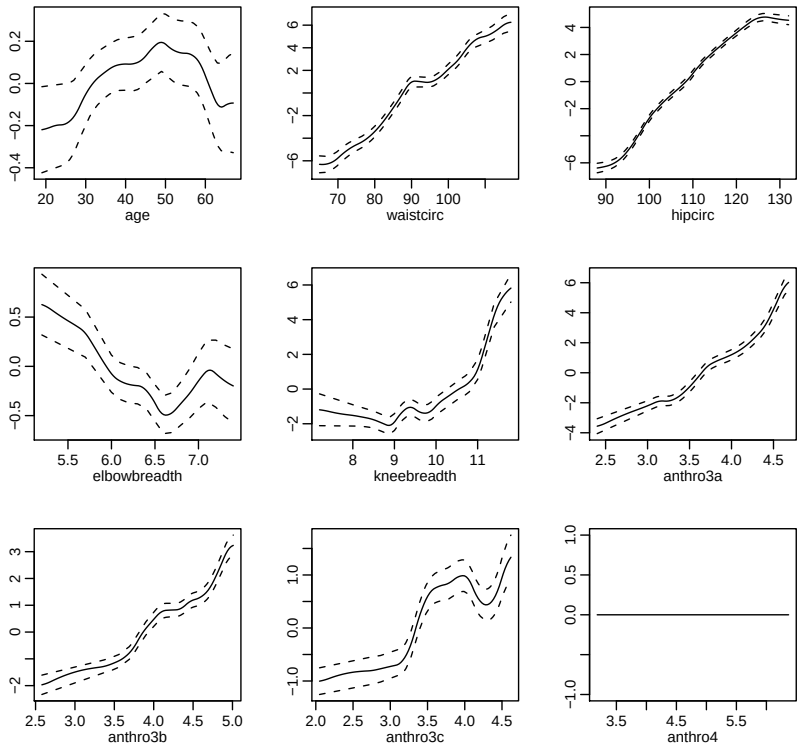

Figure 6: Contributions to predictor for each covariate as fitted using GAMBoost with cross-validation. Dashed lines are approximate 95% confidence bands.

6 Conclusions

GAMBoost as a procedure seem fairly limited when we take full account of its merits and difficulties. Choice of smoothing parameter is a matter of particular concern: the coarse line search advocated by Tutz and Binder seems ad hoc at best, and as we saw in section 4, it sometimes fails completely.

The failure of the model in cases of high signal-to-noise ratio is the most worrying problem we have encountered. It is counter intuitive that having relatively more information should cause over-fitting, but because of the attempt to place an upper limit on the number of steps used in the coarse line search, the model attempts to converge too quickly. The only way around this is to increase the number of steps and the smoothing parameter, but this makes the fitting procedure extremely slow.

The authors' choice of AIC for model selection is confusing given the lack of justification for their approximation of the effective degrees of freedom. The proof reproduced here relies on the sum of m approximate expressions, and it is not clear that this is necessarily a good approximation; presumably it would improve as the sample size n increases. cross-validation is a more robust method for model evaluation, and we would recommend it unless more results are provided concerning the AIC approximation. The same approximation is used to estimate confidence bands, so these are not well justified either; in their data analysis, Tutz and Binder suggest that over dispersion may be present in their example, and so use a bootstrap method to widen their bands. Altogether this seems a rather unsatisfactory approach. We believe that it would required only simple modifications to allow GAMBoost to support models where

dispersion parameter estimation is required. This would allow quasi-likelihood fits as well as use of gamma distributions, whereas currently only Gaussian, binomial and Poisson distributions are supported.

More positively, GAMBoost does fit models with large numbers of covariates without resort to artificially restricting the degrees of freedom, and seems to perform particularly well when fitting binary data, which is known to be a difficult problem. Computational complexity is a potential worry but sample computation times (table 2) show that GAMBoost performs well in comparison to other fitting methods, at least in the $n = 100$ case. The authors mention that this is particularly impressive given that model selection may need to be performed afterwards for fitting procedures such as wGAM, but we dispute the idea that GAMBoost is completely automatic.

Since its publication, the paper has largely been cited in papers which explore variations of boosting applications. Binder and Tutz (2008) perform more simulations to come to essentially the same conclusion as in their 2006 paper, that GAMBoost performs well when applied to problems with many uninformative covariates. An overview of various regression boosting methods, including brief mention of GAMBoost, is found in Bühlmann and Hothorn (2007). Tutz and Reithinger (2007) apply the same techniques seen here to semi-parametric mixed models, and Tutz and Binder (2007) apply what they term 'partial boosting' to ridge regression. There is no sign yet of a unifying framework for these techniques.

More widespread practical use of boosting is unlikely to emerge until a more solid foundation of theoretical results is obtained. Furthermore, the problems which Tutz and Binder suggest that GAMBoost can be used to solve, such as implicit covariate selection and smoothing parameter selection, are not really addressed in a satisfactory way; the coarse line search is an undesirable tool to use. Hastie and Tibshirani's back fitted GAMs and Wood's spline methods for GAMs are much better understood, and as we saw, may be used for covariate selection; thus we recommend an applied statistician give these methods first preference.

References

H. Binder and G. Tutz. A comparison of methods for the fitting of generalized additive models. *Stat Comput*, 18:87–99, 2008.

P. Bühlmann and T. Hothorn. Boosting algorithms: regularization, prediction and model fitting. *Stat. Sci.*, 22(4):477–505, 2007.

P. Bühlmann and B. Yu. Boosting with the l_2 loss: regression and classification. *Journal of the American Statistical Association*, 98(462):324–339, 2003.

J. Friedman. Greedy function approximation: a gradient boosting machine. *Ann. Stat.*, 29(5):1189–1232, 2001.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *Ann. Stat.*, 28(2):337–374, 2000.

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

T. Hastie and R. Tibshirani. Generalized additive models. *Statistical Science*, 1(3):297–310, 1986.

Y.-J. Kim and C. Gu. Smoothing spline gaussian regression: More scalable computation via effcient approximation. *Journal of the Royal Statistical Society, Series B*, 66(2):337–356, 2004.

L. Knorr-Held and E. Rainer. Projections of lung cancer mortality in West Germany: a case study in Bayesian prediction. *Biostatistics*, 2(1):109–129, 2001.

R.E. Schapire. The strength of weak learnibility. *Machine Learning*, 5:197–227, 1990.

D.J. Spiegelhalter, N.G. Best, and B.P. Carlin. Bayesian deviance, the effective number of parameters, and the comparison of arbitrarily complex models. Technical report, MRC Biostatistics Unit, Cambridge, 1998.

G. Tutz and H. Binder. Generalized additive modeling with implicit variable selection by likelihood-based boosting. *Biometrics*, 62:961971, 2006.

G. Tutz and H. Binder. Boosting ridge regression. *Computational Statistics & Data Analysis*, 51(12): 2872–2900, 2007.

G. Tutz and F. Reithinger. A boosting approach to flexible semiparametric mixed models. *Statistics in Medicine*, 26(14):2872–2900, 2007.

G. Wahba. *Spline models for observational data.* Philidelphia: SIAM, 1990.

S.N. Wood. *Generalized linear models: an introduction with R.* Chapman Hall, 2006.

A Proofs

Proof of Lemma 1
Since we are considering an exponential family, we can write the penalised log-likelihood in the form

$$l_\lambda(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left[\phi^{-1}(y_i\theta_i - b(\theta_i)) + c(y_i,\phi)\right] + \tfrac{1}{2}\lambda\boldsymbol{\beta}^T\Lambda\boldsymbol{\beta}.$$

Fisher scoring is given by

$$\hat{\beta}_{n+1} = \hat{\beta}_n + \hat{I}_\lambda^{-1}(\hat{\beta}_n)\hat{s}_\lambda(\hat{\beta}_n)$$

where I_λ is the penalised Fisher information matrix, and s_λ is the penalised score function.

$$\begin{aligned}
\frac{\partial}{\partial\boldsymbol{\beta}} l_\lambda &= \frac{\partial\boldsymbol{\eta}}{\partial\boldsymbol{\beta}}\frac{\partial\boldsymbol{\mu}}{\partial\boldsymbol{\eta}}\frac{\partial\boldsymbol{\theta}}{\partial\boldsymbol{\mu}}\frac{\partial}{\partial\boldsymbol{\theta}}\sum_{i=1}^{n}\left[\phi^{-1}(y_i\theta_i - b(\theta_i)) + c(y_i,\phi)\right] + \tfrac{1}{2}\frac{\partial}{\partial\boldsymbol{\beta}}\lambda\boldsymbol{\beta}^T\Lambda\boldsymbol{\beta} \\
&= \frac{\partial\boldsymbol{\eta}}{\partial\boldsymbol{\beta}}\frac{\partial\boldsymbol{\mu}}{\partial\boldsymbol{\eta}}\frac{\partial\boldsymbol{\theta}}{\partial\boldsymbol{\mu}}\frac{\partial}{\partial\boldsymbol{\theta}}\sum_{i=1}^{n}\left[\phi^{-1}(y_i\theta_i - b(\theta_i)) + c(y_i,\phi)\right] + \lambda\Lambda\boldsymbol{\beta} \\
&= \frac{\partial\boldsymbol{\eta}}{\partial\boldsymbol{\beta}}\frac{\partial\boldsymbol{\mu}}{\partial\boldsymbol{\eta}}\frac{\partial\boldsymbol{\theta}}{\partial\boldsymbol{\mu}}\phi^{-1}(\boldsymbol{y}-\boldsymbol{\mu}) + \lambda\Lambda\boldsymbol{\beta},
\end{aligned}$$

since $\mu_i = b'(\theta_i)$. Now,

$$\begin{aligned}
\eta_i &= \boldsymbol{z}_i^T\boldsymbol{\beta} \\
\boldsymbol{\eta} &= Z\boldsymbol{\beta} \\
\frac{\partial\boldsymbol{\eta}}{\partial\boldsymbol{\beta}} &= Z
\end{aligned}$$

where Z is the standard matrix of covariates. Also,

$$\begin{aligned}
\boldsymbol{\eta} &= \begin{pmatrix} g(\mu_1) \\ \cdots \\ g(\mu_n) \end{pmatrix} \\
\text{so}\quad \frac{\partial\boldsymbol{\mu}}{\partial\boldsymbol{\eta}} &= \operatorname{diag} g'(\mu_i)^{-1} := D(\boldsymbol{\mu})
\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial\mu_i}{\partial\theta_j} &= \frac{\partial b'(\theta_i)}{\partial\theta_j} \\
&= b''(\theta_i)\mathbb{1}_{\{i=j\}} \\
\text{so}\quad \frac{\partial\boldsymbol{\mu}}{\partial\boldsymbol{\theta}} &= \operatorname{diag} b''(\theta_i) \\
\frac{\partial\boldsymbol{\theta}}{\partial\boldsymbol{\mu}} &= \operatorname{diag} b''(\theta_i)^{-1}
\end{aligned}$$

and letting $\Sigma = \text{diag}\ \sigma_i^2$ where $\sigma_i^2 = \phi\, b''(\theta_i)$ we have

$$\frac{\partial}{\partial \boldsymbol{\beta}} l_\lambda = Z^T D \Sigma^{-1} (\boldsymbol{y} - \boldsymbol{\mu}) + \lambda \Lambda \boldsymbol{\beta}.$$

Letting $W = D\Sigma^{-1}D$ then

$$\frac{\partial}{\partial \boldsymbol{\beta}} l_\lambda = Z^T W D^{-1} (\boldsymbol{y} - \boldsymbol{\mu}) + \lambda \Lambda \boldsymbol{\beta}.$$

Then the penalised Fisher information is

$$I_\lambda = \mathbb{E}\left[-\frac{\partial^2 l_\lambda}{\partial \boldsymbol{\beta}^T \partial \boldsymbol{\beta}} \right].$$

It is easy to see that when the differentiation and expectation are applied to the second term we will get $\lambda\Lambda$. The first term is just the unpenalised score, and hence we can use the well known identity,

$$\begin{aligned}
\mathbb{E}\left[-\frac{\partial^2 l}{\partial \boldsymbol{\beta}^T \partial \boldsymbol{\beta}} \right] &= \mathbb{E}\left[\left(\frac{\partial l}{\partial \boldsymbol{\beta}} \right) \left(\frac{\partial l}{\partial \boldsymbol{\beta}} \right)^T \right] \\
&= \mathbb{E} Z^T D \Sigma^{-1} (\boldsymbol{y} - \boldsymbol{\mu})(\boldsymbol{y} - \boldsymbol{\mu})^T \Sigma^{-1} D Z \\
&= Z^T D \Sigma^{-1} \Sigma \Sigma^{-1} D Z \\
&= Z^T W Z.
\end{aligned}$$

Thus the Fisher scoring iteration involves

$$l\hat{\boldsymbol{\beta}}_{n+1} = \hat{\boldsymbol{\beta}}_n + (Z^T \hat{W} Z + \lambda\Lambda)^{-1} (Z^T \hat{W} \hat{D}^{-1} (\boldsymbol{y} - \hat{\boldsymbol{\mu}}) + \lambda \Lambda \hat{\boldsymbol{\beta}}_n);$$

where $\hat{W}$, $\hat{D}$ and $\hat{\mu}$ are all evaluated at $\hat{\beta}_n$. For the first iteration, we have $\boldsymbol{\beta}_0 = 0$, so

$$\hat{\boldsymbol{\beta}}_1 = (Z^T \hat{W} Z + \lambda\Lambda)^{-1} Z^T \hat{W} \hat{D}^{-1} (\boldsymbol{y} - \hat{\boldsymbol{\mu}}).$$

This gives the result. □

Proof of Lemma 2
Letting $\hat{\boldsymbol{\eta}}_{(k)}$ denote the vector of fitted predictors after k iterations, we have

$$\hat{\boldsymbol{\eta}}_{(k+1)} = \hat{\boldsymbol{\eta}}_{(k)} + Z_j \hat{\boldsymbol{\gamma}}_j$$

where j is the covariate selected for updating, and Z_j is matrix of parameters for the jth covariate. Then

$$\hat{\boldsymbol{\gamma}}_j = (Z_j^T \hat{W}_k Z_j + \lambda\Lambda)^{-1} Z_j^T \hat{W}_k \hat{D}_k^{-1} (\boldsymbol{y} - \hat{\boldsymbol{\mu}}_{(k)})$$

as shown in Lemma 1. Let $h(\boldsymbol{\eta}) = (h(\eta_1), \ldots, h(\eta_n))^T$, where $h(\cdot) = g^{-1}(\cdot)$. Then a Taylor expansion of the inverse link function (which is always twice differentiable) gives

$$h(\boldsymbol{\eta}') = h(\boldsymbol{\eta}) + \left(\frac{\partial h(\boldsymbol{\eta})}{\partial \boldsymbol{\eta}} \right)^T (\boldsymbol{\eta}' - \boldsymbol{\eta}) + o(\boldsymbol{\eta}' - \boldsymbol{\eta}^T)$$

and thus

$$h(\hat{\boldsymbol{\eta}}_{(k+1)}) \approx h(\hat{\boldsymbol{\eta}}_{(k)}) + \hat{D}_k(\hat{\boldsymbol{\eta}}_{(k+1)} - \hat{\boldsymbol{\eta}}_{(k)})$$
$$\hat{\boldsymbol{\mu}}_{(k+1)} \approx \hat{\boldsymbol{\mu}}_{(k)} + \hat{D}_k(\hat{\boldsymbol{\eta}}_{(k+1)} - \hat{\boldsymbol{\eta}}_{(k)})$$

where $\hat{D}_k = D(\hat{\boldsymbol{\mu}}_k)$ is as in Lemma 1.

$$\begin{aligned}\hat{\boldsymbol{\mu}}_{(k+1)} - \hat{\boldsymbol{\mu}}_{(k)} &\approx \hat{D}_k Z_j (Z_j^T \hat{W}_k Z_j + \lambda\Lambda)^{-1} Z_j^T \hat{W}_k \hat{D}_k^{-1}(\boldsymbol{y} - \hat{\boldsymbol{\mu}}_{(k)}) \\ &= L_{k+1}(\boldsymbol{y} - \hat{\boldsymbol{\mu}}_{(k)})\end{aligned}$$

where $L_{k+1} = \hat{D}_k Z_j (Z_j^T \hat{W}_k Z_j + \lambda\Lambda)^{-1} Z_j^T \hat{W}_k \hat{D}_k^{-1}$. Then, recursively,

$$\begin{aligned}\hat{\boldsymbol{\mu}}_{(k+1)} - \hat{\boldsymbol{\mu}}_{(k)} &\approx L_{k+1}(\boldsymbol{y} - \hat{\boldsymbol{\mu}}_{(k)}) \\ &= L_{k+1}(\boldsymbol{y} - \hat{\boldsymbol{\mu}}_{(k-1)} + (\hat{\boldsymbol{\mu}}_{(k)} - \hat{\boldsymbol{\mu}}_{(k-1)})) \\ &\approx L_{k+1}(I - L_k)(\boldsymbol{y} - \hat{\boldsymbol{\mu}}_{(k-1)}) \\ &\vdots \qquad\qquad \vdots \\ &\approx L_{k+1}(I - L_k)(I - L_{k-1}) \cdots (I - L_1)(\boldsymbol{y} - \hat{\boldsymbol{\mu}}_{(0)}).\end{aligned}$$

Further, since $\hat{\boldsymbol{\mu}}_{(0)} = (\bar{y}, \ldots, \bar{y})^T = n^{-1}\mathbf{1}\mathbf{1}^T\boldsymbol{y}$, then letting $L_0 = n^{-1}\mathbf{1}\mathbf{1}^T$ we have

$$\hat{\boldsymbol{\mu}}_{(k+1)} - \hat{\boldsymbol{\mu}}_{(k)} \approx L_{k+1}(I - L_k) \cdots (I - L_1)(I - L_0)\boldsymbol{y},$$

and then by summing differences,

$$\hat{\boldsymbol{\mu}}_{(m)} - \hat{\boldsymbol{\mu}}_{(0)} \approx \sum_{j=1}^{m} L_j(I - L_{j-1}) \cdots (I - L_0)\boldsymbol{y}$$
$$\hat{\boldsymbol{\mu}}_{(m)} \approx L_0\boldsymbol{y} + \sum_{j=1}^{m} L_j(I - L_{j-1}) \cdots (I - L_0)\boldsymbol{y}.$$

□

Proof of Lemma 3

Suppose that the j_mth covariate is selected at the mth iteration, then by Lemma 1,

$$Z_i\beta_i^{(m)} = Z_i\beta_i^{(m-1)} + R_m(\boldsymbol{y} - \hat{\boldsymbol{\mu}}_{(m-1)})\mathbb{1}_{\{i=j_m\}}$$

where $R_m = Z_j(Z_j^T \hat{W}_{m-1} Z_j + \lambda\Lambda)^{-1} Z_j^T \hat{W}_{m-1} \hat{D}_{m-1}^{-1}$. Then by Lemma 2,

$$\hat{\boldsymbol{\mu}}_{(m)} \approx H_m\boldsymbol{y},$$

so

$$\begin{aligned}Z_i\beta_i^{(m)} &\approx Z_i\beta_i^{(m-1)} + R_m(I - H_{m-1})\boldsymbol{y}\mathbb{1}_{\{i=j_m\}} \\ &\vdots \qquad\qquad \vdots \\ &\approx Z_i\beta_i^{(0)} + \sum_{k=1}^{m} R_k(I - H_{k-1})\boldsymbol{y}\mathbb{1}_{\{i=j_k\}} \\ &= 0 + \sum_{k=1}^{m} R_k(I - H_{k-1})\boldsymbol{y}\mathbb{1}_{\{i=j_k\}}\end{aligned}$$

and thus the claimed result holds with $Q_{m,j} = \sum_{k=1}^{m} R_k(I - H_{k-1})\mathbb{1}_{\{i=j_k\}}$. □

B Form of Functional Gradient Descent

This sketch derivation is mostly taken from Friedman (2001). We are interested in minimising

$$\mathbb{E}_{X,Y}[\rho(Y, f(X))],$$

and we assume that f takes the form

$$f(x) = \sum_{m=0}^{M} h^{(m)}(x)$$
$$f^{(k)}(x) = \sum_{m=0}^{k} h^{(m)}(x),$$

where each $h^{(m)}$ is an incremental step or 'boost'. For a fixed x, to obtain a steepest descent we have

$$h^{(m)}(x) = -\nu\phi^{(m)}(x)$$

where ν is a scalar and

$$\phi^{(m)}(x) = \frac{\partial}{\partial f^{(m-1)}(x)}\mathbb{E}_Y[\rho(Y, f^{(m-1)}(x))].$$

Now, assuming we may interchange the differentiation and expectation, we get

$$\phi^{(m)}(x) = \mathbb{E}_Y\left[\frac{\partial}{\partial f^{(m-1)}(x)}\rho(Y, f^{(m-1)}(x))\right].$$

Now, in general we can only observe the pair (x, Y) at a finite number of n data points, so we can use

$$-U_i = \phi^{(m)}(X_i) = \left.\frac{\partial}{\partial f^{(m-1)}(x)}\rho(Y, f^{(m-1)}(x))\right|_{(x,Y)=(X_i,Y_i)},$$

but $\phi^{(m)}$ is not defined elsewhere; we must use the strength of our observations to regress over the whole space. Thus we can choose a function from our weak learner class $\mathcal{G}$, which is most highly correlated with $\phi^{(m)}$ at the data points. But then we are just fitting the pseudo-responses U_i to the data X_i using the base procedure. We then use this n-dimensional approximation to the true function of steepest descent, which lies in a possibly infinite dimensional space of functions, and set

$$h^{(m)}(x) = \nu g^{(m)}(x).$$

C Model Code

Sample code for the binomial case with $p = 3$. Here `Y` is the vector of observations, and `X` a matrix of covariates, with columns `x1`, `x2`, `x3`.

Base. From package `stats`:

```
glm(Y ~ 1, family=binomial())
```

GLM. From package `stats`:

```
glm(Y ~ x1 + x2 + x3, data=X, family=binomial())
```

GLMse. Model fitted as GLM, then from package `MASS`:

```
stepAIC(glm.mod, direction="backward", k=2.8)
```

bfGAM. From package `gam`:

```
bfGAM.mod = gam(Y ~ s(x1) + s(x2) + s(x3), data=X, family=binomial())
step.gam(bfGAM.mod, data=X, scope = Y ~ s(x1) + s(x2) + s(x3))
```

`step.gam` was modified to give the suggested factor of 1.4 to the degrees of freedom.

wGAM. From package `mgcv`:

```
gam(Y ~ s(x1, bs="ps") + s(x2, bs="ps") + s(x3, bs="ps"), data=X,
  family=binomial(), gamma=1.4)
```

GAMBoost. From package `GAMBoost`:

```
optimGAMBoostPenalty(X, Y, family=binomial())
```

Optimal. From package `GAMBoost`:

```
cv.GAMBoost(X, Y, family=binomial())
```