Stat Comput (2012) 22:867–876
DOI 10.1007/s11222-011-9253-0

Boosted coefficient models

Joseph Sexton · Petter Laake

Received: 3 July 2010 / Accepted: 23 February 2011 / Published online: 6 April 2011

Abstract Regression methods typically construct a mapping from the covariates into the real numbers. Here, however, we consider regression problems where the task is to form a mapping from the covariates into a set of (univariate) real-valued functions. Examples are given by conditional density estimation, hazard regression and regression with a functional response. Our approach starts by modeling the function of interest using a sum of B-spline basis functions. To model dependence on the covariates, the coefficients of this expansion are each modeled as functions of the covariates. We propose to estimate these coefficient functions using boosted tree models. Algorithms are provided for the above three situations, and real data sets are used to investigate their performance. The results indicate that the proposed methodology performs well. In addition, it is both straightforward, and capable of handling a large number of covariates.

Keywords Boosting · Conditional density estimation · Functional regression · Hazard regression · Regression trees

1 Introduction

Many regression problems require the estimation of a mapping from the covariates into the real numbers. The goal is to construct this mapping such that each realization of the covariates is mapped into a good prediction of the corresponding response. Here, however, we consider regression problems where the task is to construct a mapping from the covariates into a set of (univariate) real valued functions. For instance, in conditional density estimation the object is to estimate how the density of the response depends on the covariates. A related problem occurs in survival analysis. Here the hazard function plays a key role, and estimating how this function depends on the covariates is of central interest. In other cases, for instance longitudinal data analysis, the response may be viewed as repeated measurements of an underlying curve, and the object is to determine how this curve varies with the covariates.

The purpose of this article is to describe how boosting methodology can be applied to these situations. Boosting, which originates from machine learning, has become a popular approach for fitting regression and classification models. Overviews are provided by Hastie et al. (2009) and Büehlmann and Hothorn (2007). The main application of boosting has been towards estimating models for predicting or classifying a univariate response. This article extends the application of booting to situations where the response may be thought of as a function.

To illustrate our approach, consider the problem of density estimation. Letting $f(t)$ denote the density of some variable Y, a flexible approach studied by Stone and Koo (1986) and O'Sullivan (1988), among others, is to model the logarithm of f using a B-spline basis (de Boor 1978). Specifically, one can express the density as $f(t) = e^{\eta(t)} / \int e^{\eta(u)} du$, and model $\eta(t)$ using

$$\eta(t) = \sum_{j=1}^{J} \beta_j B_j(t). \tag{1}$$

Here $B_j(t)$ is the j-th B-spline basis function, and β_j its coefficient. The number of B-spline basis functions J is governed by the number of knots employed, with a greater number of knots leading to a more flexible model. Estimation can

J. Sexton (✉) · P. Laake
Department of Biostatistics, Institute of Basic Medical Sciences, P.O. Box 1122, Blindern, 0317 Oslo, Norway
e-mail: j.a.sexton@medisin.uio.no

be carried out by maximizing the corresponding likelihood function, or some penalized variant of it.

Now suppose that interest lies in relating the density of Y to a set of covariates X. Thus, one wants to estimate $f(t|x)$, the conditional density of Y given $X = x$. In this case, for each value of x one can imagine a model of form (1). The coefficients of (1) will vary with x, implying the model

$$\eta(t|x) = \sum_{j=1}^{J} \beta_j(x) B_j(t), \tag{2}$$

with $f(t|x) = e^{\eta(t|x)} / \int e^{\eta(u|x)} du$.

The task now is to model and estimate the coefficient functions $\beta_j(x)$, $j = 1, \ldots, J$. Setting up appropriate models for the $\beta_j(x)$'s is not straightforward. The dimension of x may be large, it may contain predictors of mixed type, there may be interactions between the predictors, and non-linear relations. A powerful function approximation methodology that automatically deals with such issues is that of boosted regression trees (Hastie et al. 2009). Here we adapt this methodology to estimate the coefficient functions $\beta_j(x)$ in models of form (2).

The remainder of the article is organized as follows. In Sect. 2 we review regression trees, and the gradient boosting algorithm of Friedman (2001). In Sect. 3 we describe how gradient boosting with trees can be applied to estimate models of form (2). Section 4 considers applications to conditional density estimation, hazard regression and regression with a functional response. Section 5 investigates the performance of the methods on real data, and Sect. 6 concludes.

2 Background

2.1 Regression trees

Tree models (Breiman et al. 1984) have many desirable properties. They can handle covariates of mixed types, categorical, ordered and continuous; are invariant to monotone transformations of the predictors; they have built in variable selection, and thus can be applied in cases with numerous covariates; and they are quick to induce and to predict from. We will use regression trees for both univariate and multivariate responses, which are now described.

2.1.1 Univariate response

Consider a regression problem with a scalar response y_i with an accompanying p dimensional predictor variable x_i, $i = 1, \ldots, n$. The model fit by a regression tree takes the form:

$$T(x) = \sum_{k=1}^{K} c_k I(x \in R_k), \tag{3}$$

where $I()$ denotes the indicator function. The R_k, $k = 1, \ldots, K$, form a partition of the covariate space, and c_k is the prediction of y if $x \in R_k$.

Trees are typically constructed using recursive binary partitioning. The process starts by first partitioning the covariate space by considering all possible partitions of form $x_j < d$ and $x_j \geq d$ where j ranges from 1 to p, and d over all possible real values. The partition leading to the largest decrease in some measure of loss, for instance squared error, is selected. Next, these two regions are themselves split in the same manner and the process continues on producing the tree. Each resulting partition of the covariate space is termed a node, and the final set of partitions are called the terminal nodes. For single tree models, some manner of determining when to stop growing the tree is needed, to avoid overfitting. However, when trees are combined with boosting, one typically uses shallow trees, having a small pre-determined number of terminal nodes.

2.1.2 Multivariate response

Now consider the same setting as above, but with a multivariate response, $y_i = (y_{i1}, \ldots, y_{iq})$. A tree model for this case also takes the form in (3), but now the c_k are vectors of same dimension as the response. Constructing a multivariate tree proceeds exactly as for the univariate case, using a measure of loss appropriate for multivariate responses. For our purposes, squared error loss will be used, and thus the error in a node defining a region R is

$$\sum_{i=1}^{n} \sum_{j=1}^{q} (y_{ij} - \mu_j)^2 I(x_i \in R), \tag{4}$$

where $\mu_j = \sum_{i=1}^{n} y_{ij} I(x_i \in R) / \sum_{i=1}^{n} I(x_i \in R)$.

2.2 Gradient boosting

A single tree model, typically, does not have excellent predictive accuracy. However, combining multiple trees using boosting has been shown to yield models with very good predictive performance. Here we review the gradient tree boosting algorithm of Friedman (2001), which forms the basis of our estimation approach described in Sect. 3.

For an observation (y, x), let $\beta(x)$ denote the model prediction of y. Given a loss function $L(y, \beta(x))$, for instance squared error, the task is to form a $\beta()$ with low predictive error. The gradient tree boosting algorithm builds a model for $\beta()$ in a stagewise manner. At iteration m, the model on the previous iteration, denoted $\beta^{(m-1)}()$, is used to form (generalized) residuals. A regression tree is then fit to these residuals, using the original covariates. This regression tree is then added to $\beta^{(m-1)}()$ to form the updated model.

The generalized residuals are defined as

$$r_i^{(m)} = -\frac{\partial}{\partial \beta(x_i)} L(y_i, \beta(x_i))|_{\beta=\beta^{(m-1)}}, \quad i = 1, \ldots, n, \quad (5)$$

where the partial derivative in (5) is formed treating each $\beta(x_i)$ as a separate parameter.

These residuals are then regressed on x to form a K-terminal node regression tree, the terminal node regions of which are denoted $R_k^{(m)}$ for $k = 1, \ldots, K$. Fixing these nodes, the tree

$$T^{(m)}(x; c) = \sum_{k=1}^{K} c_k I(x \in R_k^{(m)}), \quad (6)$$

can be viewed as a function of the terminal node parameters c_k, $k = 1, \ldots, K$. The next step of the algorithm is to find appropriate values of these parameters using

$$c^{(m)} = \underset{c}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \beta^{(m-1)}(x_i) + T^{(m)}(x_i; c)). \quad (7)$$

Finally, the model is updated:

$$\beta^{(m)}(x) = \beta^{(m-1)}(x) + \lambda T^{(m)}(x; c^{(m)}), \quad (8)$$

where λ is a pre-set shrinkage factor between 0 and 1. The steps (5) to (8) are repeated a number m_{stop} times, typically chosen using some form of cross-validation.

Note that the minimization in (7) may require iteration, and thus will be costly if m_{stop} is large. In such cases, the strategy in Friedman (2001) is to perform a single iteration of a Newton-Raphson algorithm.

3 Gradient boosted coefficient models

We now return to the problem of estimating the coefficient functions described in the introduction. In particular, we are interested in fitting the $\beta_j(x)$ functions in the model

$$\eta(t|x) = \sum_{j=1}^{J} \beta_j(x) B_j(t), \quad (9)$$

where $B_j(t)$, $j = 1, \ldots, J$, is a set of B-spline basis functions. In our applications we have used a cubic B-spline basis, in which case J is equal to four plus the number of knots. Furthermore, the knots and their placement are considered fixed, determined prior to the estimation. The function $\eta(t|x)$ can be thought of as the logarithm of the conditional density function, or it could be something else. Specific examples are considered in Sect. 4.

Our approach is to estimate the coefficient functions using an additive expansion of regression trees. That is

$$\beta_j(x) = \sum_{m=1}^{m_{stop}} T_j^{(m)}(x) \quad (10)$$

where each $T_j^{(m)}(x)$ is a regression tree. To estimate this model we utilize the gradient boosting algorithm.

The method in Sect. 2.2 can, however, not be directly applied. The reason is that now multiple boosted tree models need to be estimated. Instead of there being a single (generalized) residual associated with each observation, there are now J of these, one for each of the β_j functions. At the m-th iteration, these residuals are given by

$$r_{ij}^{(m)} = -\frac{\partial}{\partial \beta_j(x_i)} L(y_i, \eta(t|x_i))|_{\beta_j=\beta_j^{(m-1)}}, \quad j = 1, \ldots, J, \ i = 1, \ldots, n, \quad (11)$$

where $L(y, \eta(t|x))$ denotes the loss of predicting y using $\eta(t|x)$. Evaluating (11) is done similarly to (5).

We consider two approaches to generalizing the gradient boosting algorithm such that it can be applied to multivariate residuals. The first utilizes multivariate regression trees, while the second employs multiple univariate trees.

3.1 Coefficient boosting with multivariate trees

In this approach, a multivariate regression tree is grown at each iteration of the boosting algorithm. The response is the residual vector $r_i^{(m)} = (r_{i1}^{(m)}, \ldots, r_{iJ}^{(m)})$, and the tree is grown regressing $r_i^{(m)}$ on x. Following the prescription in Friedman (2001), we want to grow this tree such that it is most correlated with the gradient, i.e. the multivariate residual. To accomplish this we use squared error loss as the node splitting criteria in the tree growing, given in (4).

Having fit the tree, it is used to update the model via

$$\eta^{(m)}(t|x) = \eta^{(m-1)}(t|x) + B(t)' T^{(m)}(x; c) = \eta^{(m-1)}(t|x) + \sum_{k=1}^{K} B(t)' c_k \cdot I(x \in R_k^{(m)}), \quad (12)$$

where $B(t)' = (B_1(t), \ldots, B_J(t))'$ and $c = (c_1, \ldots, c_K)$ with $c_k = (c_{1k}, \ldots, c_{Jk})$. Note that this updating step in effect adds, for each k, a function $\sum_j c_{kj} B_j(t)$ to the region $R_k^{(m)}$. However, before updating the model with $T^{(m)}$, an appropriate value of the c vector needs to be determined. One approach is to use

$$c^{(m)} = \underset{c}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \eta^{(m-1)}(t|x_i) + B(t)' T^{(m)}(x_i; c)), \quad (13)$$

similar to (7) in the gradient boosting algorithm.

Table 1 Coefficient boosting with multivariate trees algorithm

For $m = 1$ to m_{stop}
$r_{ij}^{(m)} = -\partial/(\partial\beta_j(x_i))L(y_i, \eta(t|x_i))|_{\beta_j=\beta_j^{(m-1)}}$, $j = 1,\ldots,J$ and $i = 1,\ldots,n$
Set $r_i^{(m)} = (r_{i1}^{(m)},\ldots,r_{iJ}^{(m)})$, for $i = 1,\ldots,n$
Regress $r_i^{(m)}$ on x_i, giving:
$T^{(m)}(x;c) = \sum_{k=1}^{K} c_k I(x \in R_j^{(m)})$, $c_k = (c_{k1},\ldots,c_{kJ})$ for $k = 1,\ldots,K$
$c^{(m)} = \text{argmin}_c \sum_i L(y_i, \eta^{(m-1)}(t|x_i) + B(t)'T^{(m)}(x_i;c)) + \gamma\text{Pen}(c)$
$\eta^{(m)}(t|x) = \eta^{(m-1)}(t|x) + \lambda \cdot B(t)'T^{(m)}(x;c^{(m)})$
EndFor

The problem with (13) is that it does not impose smoothness on the resulting model update. For each k, we would like to ensure that the added function $\sum_j c_{kj}B_j(t)$ is smooth. This can be accomplished by adding a penalty to the loss function, penalizing irregularity in the update. A convenient approach is to use a difference penalty on each c_k, as advocated in Eilers and Marx (1996). We use a second order difference penalty, giving $\text{Pen}(c) = \sum_{k=1}^{K}\sum_{j=3}^{J}(c_{kj} - 2c_{k,j-1} + c_{k,j-2})^2$. The resulting update of the terminal node parameters takes the form

$$c^{(m)} = \underset{c}{\text{argmin}} \sum_{i=1}^{n} L(y_i, \eta^{(m-1)}(t|x_i) + B(t)'T^{(m)}(x_i;c)) + \gamma\text{Pen}(c) \tag{14}$$

where γ is a pre-set penalty parameter. Note that $\text{Pen}(c)$ is a quadratic penalty, and can be written $\text{Pen}(c) = c'Mc$, where M is a band-diagonal matrix with bandwidth equal to 2. Furthermore, solving this minimization problem may be difficult, and in such cases shortcuts must be taken to save computational effort, see Sect. 4 for further details.

The algorithm is summarized in Table 1. Note, importantly, that the only regularization parameter to be estimated is the number of boosting iterations, m_{stop}, which is typically done by cross-validation. The penalty parameters λ and γ are set beforehand and fixed throughout the estimation.

Our method does not directly enforce smoothness on $\eta(t|x)$. Instead, this function is updated at each iteration by adding a function, namely $B(t)'T^{(m)}(x;c)$, which is smoothed across t according to the penalty. The approach of applying regularization to the base-learner, but not directly to the model itself is common to boosting algorithms, see for instance Büehlmann and Hothorn (2007). Regularization of the boosting model is applied by selecting an appropriate stopping iteration, m_{stop}.

The above approach imposes smoothness after having obtained the terminal node regions, i.e. (14). However, the regions themselves are obtained using (unpenalized) least squares node splitting, with no regard to smoothness. It could be, as suggested by a referee, that smoothness should be considered when building the regression tree. This can be accomplished by adding a difference penalty to the squared error used in the node splitting, i.e. to (4). The two approaches are compared in Sect. 5.

3.2 Coefficient boosting with univariate trees

Instead of building a multivariate tree as above, one can consider the residuals corresponding each coefficient function separately, and construct separate univariate trees to each of these. Thus, at the m-th boosting iteration one loops through the basis functions, at each j ($j = 1,\ldots,J$) fitting a regression tree $T_j^{(m)}$ to the residuals $r_{ij}^{(m)}$ ($i = 1,\ldots,n$).

Having fit the J trees, the coefficients of the trees are determined using a step similar to (14). One modification needs to be made, however, since the regions of the trees are no longer common as in the multivariate tree approach. Thus if we want to impose smoothness on the function updates we need to smooth or penalize across trees corresponding neighboring basis functions. In particular, consider imposing smoothness at a point x^*. At x^* the function update is $\sum_j T_j(x^*)B_j(t)$. To impose smoothness we apply a difference penalty to the coefficients of this function. Now consider a set of points x_s^* ($s = 1,\ldots,S$) at which the function is constrained to be smooth. Using a second order difference penalty, this implies using the penalty

$$\text{Pen}(c) = \sum_{j=3}^{J}\sum_{k_1,k_2,k_3=1}^{K}(c_{k_1,j} - 2c_{k_2,j-1} + c_{k_3,j-2})^2 \times n_j(k_1,k_2,k_3), \tag{15}$$

where $n_j(k_1,k_2,k_3) = \sum_{s=1}^{S} I(x_s^* \in R_{j,k_1} \cap R_{j-1,k_2} \cap R_{j-2,k_3})$. This penalty can be written in matrix form as $\text{Pen}(c) = c'Mc$, where M is a banded diagonal matrix with bandwidth equal to $3 \cdot K$, K being the number of terminal nodes in the trees. Finding $n_j(k_1,k_2,k_3)$ can be done by dropping the x^*'s down the trees, and counting the number x_s^*'s that end up in $R_{j,k_1} \cap R_{j-1,k_2} \cap R_{j-2,k_3}$. The algorithm in summarized in Table 2.

3.3 Interpretation

To gauge the overall contribution of the various covariates on a boosted tree model, variable importance measures are

Table 2 Coefficient boosting with univariate trees algorithm

For $m = 1$ to m_{stop}
  For $j = 1$ to J
    $r_{ij}^{(m)} = -\partial/(\partial\beta_j(x_i))L(y_i, \eta(t|x_i))|_{\beta_j=\beta_j^{(m-1)}}$ for $i = 1, \ldots, n$
    Regress $r_{ij}^{(m)}$ on x_i, giving $T_j^{(m)}(x; c) = \sum_{k=1}^{K} c_{kj} I(x \in R_{kj})$
  EndFor
  $c^{(m)} = \text{argmin}_c \sum_i L(y_i, \eta^{(m-1)}(t|x_i) + \sum_j T_j^{(m)}(x_i; c_j)B_j(t)) + \gamma \text{Pen}(c)$
  $\eta^{(m)}(t|x) = \eta^{(m-1)}(t|x) + \lambda \cdot \sum_j T_j^{(m)}(x; c_j^{(m)})B_j(t)$
EndFor

available, see Hastie et al. (2009). Such measures can also be used here. For instance, let $VI_j(l)$ denote the variable importance of covariate x_l on the model for $\beta_j(x)$, then an overall measure of the importance of x_l can be formed averaging across j, i.e. $VI(l) = \sum_{j=1}^{J} VI_j(l)/J$.

To graphically depict how a covariate influences a boosted tree model, Friedman (2001) introduced plots of the partial dependence functions. The partial dependence of covariate x_l, is a function of x_l summarizing how it influences the boosted tree model $\beta(x)$, and is denoted $\bar{\beta}_l(x_l)$. It is formed by averaging $\beta(x)$ over all observed values of the other covariates, holding x_l fixed, i.e. $\bar{\beta}_l(x_l) = \sum_i \beta((x_{i1}, \ldots, x_{ip}))|_{x_{il}=x_l}/n$. Extending partial dependence functions to boosted coefficient models is straightforward. In particular, let $\bar{\beta}_{jl}(x_l)$ be the partial dependence of covariate x_l on $\beta_j(x)$, then the partial dependence of $\eta(t|x)$ on this covariate is the function $\bar{\eta}_l(t|x_l) = \sum_j \bar{\beta}_{jl}(x_l)B_j(t)$.

4 Applications

This section considers applications of the boosted coefficient algorithms. Three applications are considered, namely conditional density estimation, survival distribution estimation and regression with a functional response.

4.1 Conditional density estimation

In conditional density estimation the task is to estimate the density of a real-valued variable Y conditional on given values of the covariates X. We denote the conditional density by $f(t|x)$, and assume that independent observations, (y_i, x_i) for $i = 1, \ldots, n$, are available. Our approach is to model the logarithm of $f(t|x)$ using a B-spline basis. That is, we set $f(t|x) = e^{\eta(t|x)}/\int e^{\eta(u|x)}du$ and use the model $\eta(t|x) = \sum_j \beta_j(x)B_j(t)$. The negative log-likelihood loss is given by

$$L(y, \eta(t|x)) = -\eta(y|x) + \log\left\{\int e^{\eta(t|x)}dt\right\} \tag{16}$$

with the negative log-likelihood being $\sum_{i=1}^{n} L(y_i, \eta(t|x_i))$.

In this setting, the residuals at the m-th boosting iteration take the form:

$$r_{ij}^{(m)} = B_j(y_i) - E(B_j(Y)|x_i; \eta^{(m-1)}), \tag{17}$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, J$.

Here, solving the optimization problem (14) requires iteration. To avoid this, the approach in Friedman (2001) is followed, and only a single iteration of the Newton-Raphson algorithm is carried out. This leads to setting

$$c^{(m)} = (H + \gamma M)^{-1}\nabla, \tag{18}$$

where $\nabla = -\partial/\partial c \sum_i L(y_i, \eta^{(m-1)}(t|x_i) + B(t)' \times T^{(m)}(x_i; c))|_{c=0}$, whose elements are of form $\sum_i r_{ij}^{(m)} I(x_i \in R_{kj}^{(m)})$ for $j = 1, \ldots, J$ and $k = 1, \ldots, K$. In a straightforward Newton-Raphson iteration, H is the matrix of second order derivatives with respect to the vector c. However to simply, we approximate this matrix by using only its diagonal elements, a short-cut also utilized by Friedman (2001). Thus, H in (18) is a diagonal matrix whose elements are of form:

$$\sum_{i=1}^{n} E(B_j(Y)^2|x_i; \eta_{m-1}) \cdot I(x_i \in R_{jk}^{(m)}), \tag{19}$$

for $j = 1, \ldots, J$ and $k = 1, \ldots, K$. The matrix M in (18) corresponds the quadratic penalty, which is band-diagonal with bandwidth equal to 2 for the multivariate tree approach and $3 \cdot K$ for the univariate tree approach, using a second order difference penalty. Solving (18) can be done quickly using Cholesky factorization for band-diagonal matrices.

To summarize. At each iteration, the one-dimensional integrals $E(B_j(Y)|x_i; \eta^{(m-1)})$ and $E(B_j(Y)^2|x_i; \eta^{(m-1)})$, for $i = 1, \ldots, n$, need to be computed. Next, the residuals in (17) are formed, and regressed on the covariates, using either a multivariate regression tree, or J univariate regression trees. The terminal node coefficients are formed solving (18) using band-diagonal Cholesky factorization, upon which the model is updated, and the steps iterated.

Having estimated the conditional density, it is straightforward to find corresponding estimates of the conditional

distribution and quantile functions. In particular, the conditional distribution function is $F(t|x) = \int_{-\infty}^{t} f(s|x)ds$ and the conditional quantile function is $F^{-1}(t|x)$.

4.2 Survival analysis

In survival studies the statistical analysis is often complicated by right-censoring. This means that for some individuals, all that is known is that their survival time exceeded some known duration, the censoring time. This might be due to termination of the study, or from subjects being lost to follow-up. Here we consider applying coefficient boosting to such data.

Let T_i be the true survival time, R_i the right-censoring time and x_i the covariates, for $i = 1, \ldots, n$. The observed data are (y_i, δ_i, x_i) for $i = 1, \ldots, n$, where $y_i = \min(T_i, R_i)$ and $\delta_i = I(T_i \leq R_i)$ is the censoring indicator. Interest is on estimating how the covariates influence the survival distribution, defined as $S(t|x) = 1 - F(t|x)$ where $F(t|x)$ is the conditional distribution function, i.e. $F(t|x) = P(T \leq t|x)$.

Estimation of $S(t|x)$ is often done by forming a model for the hazard function, which is defined as $\alpha(t|x) = -\partial \log S(t|x)/\partial t$, and is non-negative. Here we apply coefficient boosting to the log-hazard function, i.e. $\log \alpha(t|x) = \eta(t|x)$. Note that $S(t|x) = \exp(-\int_0^t e^{\eta(s|x)}ds)$.

Here the negative log-likelihood loss is

$$L(y_i, \eta(t|x_i)) = -\delta_i \eta(y_i|x_i) + \int_0^{y_i} e^{\eta(t|x_i)}dt. \tag{20}$$

The generalized residuals at the m-th boosting iteration are:

$$r_{ij}^{(m)} = \delta_i B_j(y_i) - \int_0^{y_i} B_j(t) e^{\eta^{(m-1)}(t|x_i)}dt, \tag{21}$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, J$. Furthermore, the diagonal elements of the H matrix in (18) are of form $\sum_i \int_0^{y_i} B_j(t)^2 e^{\eta^{(m-1)}(t|x_i)}dt \cdot I(x_i \in R_{jk}^{(m)})$. With these modifications, the estimation proceeds as with the conditional density estimation.

4.3 Functional regression

In some longitudinal regression problems, the response can be viewed as being repeated measurements of an underlying curve (Ramsay and Silverman 2005), possibly measured with considerable noise. The task is to estimate how the curve depends on the observed covariates. Letting $y(t)$ denote the response at time t, the data take the form of (y_i, x_i) for $i = 1, \ldots, n$, where $y_i = (y_i(t_{i1}), \ldots, y_i(t_{in_i}))$, with t_{ij}, $j = 1, \ldots, n_i$ denoting the measurement time points of subject i. We model the mean of $y(t)$, conditional on x, as:

$$E(y(t)|x) = \eta(t|x) = \sum_{j=1}^{J} \beta_j(x) B_j(t). \tag{22}$$

Here we consider applying coefficient boosting to estimate model (22). For simplicity, we consider using squared error loss:

$$L(y_i, \eta(t|x_i)) = \sum_{h=1}^{n_i} (y_i(t_{ih}) - \eta(t_{ih}|x_i))^2. \tag{23}$$

In this case, the generalized residuals are $r_{ij}^{(m)} = \sum_{h=1}^{n_i} (y_i(t_{ih}) - \eta^{(m-1)}(t_{ih}|x_i)) B_j(t_{ih})$, and the diagonal elements of H in (18) are of form $\sum_i \sum_{h=1}^{n_i} B_j(t_{ih})^2 \cdot I(x_i \in R_{jk}^{(m)})$. Beyond this estimation proceeds as in Sect. 4.1.

Loss functions besides (23) are possible, and more robust loss functions may be preferred if there is considerable noise in the measurements of y. Friedman (2001) provides details for implementing absolute deviation and the Huber loss function, which both could be straightforwardly adapted to our setting.

5 Experiments

5.1 Conditional density estimation

In this section, we report on applications our approach to conditional density estimation, as well as a comparison with an alternative method. The alternative method was kernel conditional density estimation, which has been studied by Bashtannyk and Hyndman (2001) and Hall et al. (2004), among others. All the computations were carried out using the R software (R Development Core Team 2009). The kernel conditional density method of Hall et al. (2004) is implemented in the R-package *np* (Hayfield and Racine 2008), which was used here. Our code implementing the coefficient boosting algorithms is available from the website: http://folk.uio.no/josephse.

Four data sets were considered. These were BostonHousing ($n = 506$, $p = 13$), Ozone ($n = 366$, $p = 12$), BigMac ($n = 69$, $p = 9$), and Fuel ($n = 51$, $p = 5$). The first two were taken from the R-package *mlbench* (Leisch and Dimitriadou 2010) and the last two from the R-package *alr3* (Weisberg 2009).

The coefficient boosting method has four meta-parameters. These are m_{stop}, λ, γ and the number of knots defining the cubic B-spline basis. An additional meta-parameter is the number of terminal nodes in the tree models, which was set to four in all experiments. The two parameters m_{stop} and λ are common to most boosting algorithms. Typically, m_{stop} is estimated using cross-validation, and we have used 10-fold cross-validation. The λ parameter was set to 0.1. Concerning γ and the number of knots, we have found that the results can be sensitive to the latter, but fairly insensitive former, provided it is not too small. Figures 1(a) and 1(b) illustrate these points using the Boston Housing data. Figure 1(a)

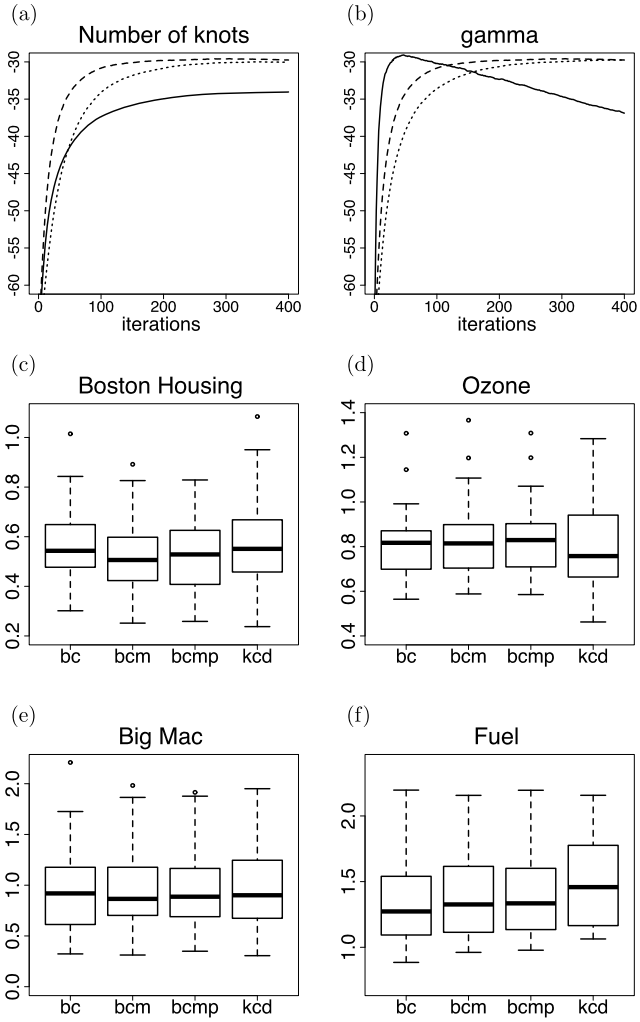

Fig. 1 Number of knots, γ and test set error. Plot (**a**) shows the cross–validated log-likelihood using 4 (*solid line*), 10 (*long-dashed line*) and 20 (*short-dashed line*) knots in boosting approach to conditional density estimation to Boston Housing data. Plot (**b**) is similar to (**a**) but shows the cross-validated log-likelihood corresponding $\gamma = .05$ (*solid line*), $\gamma = .1$ (*long-dashed line*) and $\gamma = .2$ (*short-dashed line*), with 10 knots. Plots (**c**)–(**f**) show boxplots of test set error (negative log-likelihood) for coefficient boosting with univariate trees (bc), multivariate trees with unpenalized splitting (bcm) and penalized splitting (bcmp), and kernel conditional density estimation (kcd)

shows the cross-validated log-likelihood using 4, 10, and 20 knots. It is seen that using 10 and 20 knots gives similar results, while 4 knots are insufficient. Figure 1(b) also shows the cross-validated log-likelihood, but for different values of γ, namely 0.05, 0.1 and 0.2, with the number of knots fixed at 10. Here a similar cross-validated log-likelihood is obtained for each of the γ values, provided a sufficient number of iterations are carried out. Similar conclusions were obtained using the other data sets. Based on these observations, it seems sensible to run the coefficient boosting algorithm using a different number of knots, and selecting the number giving the smallest cross-validated log-likelihood.

Three versions of the boosting approach to conditional density estimation were investigated. The multiple univariate tree approach, and two versions of the multivariate tree approach, one using un-penalized node splitting in the regression tree algorithm, the other employing penalized node splitting as described at the end of Sect. 4.1. For each, cross-validation was used to select the number of knots as well as the m_{stop} parameter. The kernel conditional density routine was run using likelihood based cross-validation of the bandwidth parameters. Unfortunately, kernel conditional density estimation is computationally demanding, and we found that applying the routine in *np* to problems with more than two predictor variables was not practical. It took too long. Therefore, the comparisons were run using only two predictor variables from each data set. From each data set, we selected the two most influential predictors, which were determined by using the *randomForest*-package (Liaw and Wiener 2002), selecting the two predictors with the highest variable importance. Note, however, that the boosting approach to conditional density estimation described here is practical in problems with a larger number of predictors, but for sake of comparison only the two predictors were used.

The comparison was done as follows. For each data set, a test set consisting of 10% of the data was randomly selected, and the methods estimated using the remaining cases. The prediction error of each method was then estimated by computing the negative log-likelihood on the test set data. This was repeated 50 times for each data set, and Figs. 1(c)–(f) give the boxplots of the results. On these data the boosting and kernel conditional density estimation approaches gave similar results. The multiple univariate tree boosting approach was similar to the two multivariate tree approaches, and the penalized and un-penalized multivariate tree methods likewise resulted in similar test set error.

5.2 Survival analysis

In this section, we report on a comparison of the coefficient boosting approach to hazard regression with two other methods. The first of these was random survival forests, introduced by Ishwaran et al. (2008) and implemented in the *randomSurvivalForest*-package (Ishwaran and Kogalur 2010). The second was gradient boosting applied to estimating a proportional hazards model, as described in Ridgeway (1999), and implemented in the *gbm*-package (Ridgeway 2007).

Four data sets were used in the comparison. Three of these are provided in *randomSurvivalForest*-package, and are *recid*, *veteran*, and *pbc*. The fourth was the *GBSG-2* data, taken from the R-package *ipred* (Peters and Hothorn 2009). The number of iterations for the boosting methods was determined using 10-fold cross-validation, and random survival forests was run using default settings. Initial runs

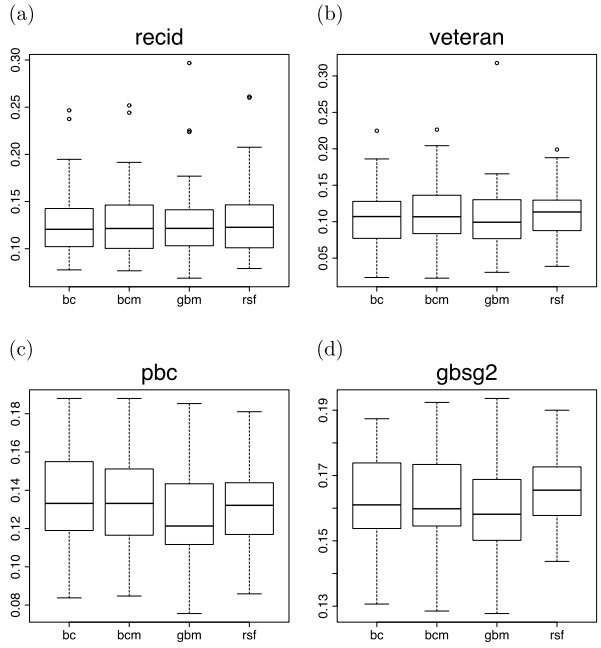

Fig. 2 Prediction error on survival data. Plots give the boxplots of the prediction error (integrated Brier score) on randomly selected test sets. "bc" is coefficient boosting applied to the hazard function using univariate trees, "bcm" is similar using multivariate trees, "gbm" is gradient boosting applied to the proportional hazards model, "rsf" is random survival forests

of the coefficient boosting method indicated that a smaller number of knots was appropriate for these data, and four knots were used. Evaluation was done using the integrated Brier score for censored data, described in Graf et al. (1999), and implemented in the *ipred*-package. The lower integration limit for the Brier score was set to 0 and the upper limit was set to the maximum of the observed times. The comparison was performed similarly to that for conditional density estimation. For each data set we repeated the following 50 times: 90% of the observations were randomly selected and used as a training set, and the Brier score was evaluated on the remaining 10% of the data.

The boxplots of the Brier score evaluated on the test sets are given in Fig. 2. Here 'bc' denotes coefficient boosting using multiple univariate regression trees, 'bcm' denotes coefficient boosting using multivariate regression trees, 'gbm' gradient boosting applied to the proportional hazards model, and 'rsf' random survival forests. Overall, the methods performed similarly on the four data sets, with the 'gbm' method being slightly better on the pbc data. The 'rsf' method and the coefficient boosting approaches resulted in similar prediction error. The coefficient boosting approach using univariate regression trees gave similar results to the approach using multivariate trees. Relatively few iterations were required by the boosting methods, possibly reflecting a fairly low signal in these data.

5.3 Functional regression

In this section, we apply coefficient boosting to the problem of regression with a functional response, and consider an interesting data set described in Faraway (1997). The data concern the movement trajectories of individuals reaching for different targets, while sitting in a car. The trajectories were captured by four cameras, and markers on the individuals were sampled at a rate of 25 Hertz. The purpose of the study was to develop a predictive model for the trajectories.

The data considered by Faraway (1997), available at http://www.maths.bath.ac.uk/~jjf23/fda, are of an angle between the shoulder, elbow and wrist. These are from a single individual making 3 reaches to each of 20 different targets. To remove the effect of movement speed, the time points associated with each curve were scaled from 0 to 1. The data thus consist of 60 different curves, with time ranging from 0 to 1 on each, and associated with each curve are the coordinates of the target.

The curves were sampled at different times, and Faraway (1997) first smoothed the curves, and then obtained predictions on an equally spaced grid, using 20 grid points. This was done, as his strategy was to fit a separate model at each time point. Letting $y_i(t)$ denote the i-th curve at time t, the data are $(y_i(t_j), x_i)$, $i = 1, \ldots, 60$ and $t_j = 1/20, 2/20, \ldots, 1$.

After the initial smoothing step, Faraway (1997) first fit a linear model to each time point, which was not satisfactory. Following this, a quadratic model, including all second order interactions, was fit to each time point. Plots of its fitted curves showed good agreement with the data, as did a formal test. Here we apply the coefficient boosting algorithm of Sect. 3.3 to this data. We note that the coefficient boosting method does not require data to be sampled at the same time points, but for sake of comparison, this data was used.

As the aim was to predict the trajectory given a new target point, we did the following. For each target point, the corresponding trajectory data was removed, and the coefficient boosting model and the quadratic model of Faraway (1997) was trained on the data corresponding the other targets. These models were then used to predict the trajectory at the left-out target point. Figures 3(a)–(f) show the results for the first six trajectories. The solid line corresponds the boosted coefficient estimate, the long dashed the quadratic model, and the short dashed lines give the data corresponding the three reaches. The quadratic model and the coefficient boosting model appear to give fairly similar results. In particular, the predicted trajectories on plots (a), (b), (e) and (f) are in good agreement with the observed trajectories. Both methods fail to give an accurate prediction of the trajectories plotted in Fig. 3(c). For the trajectory in plot (d)

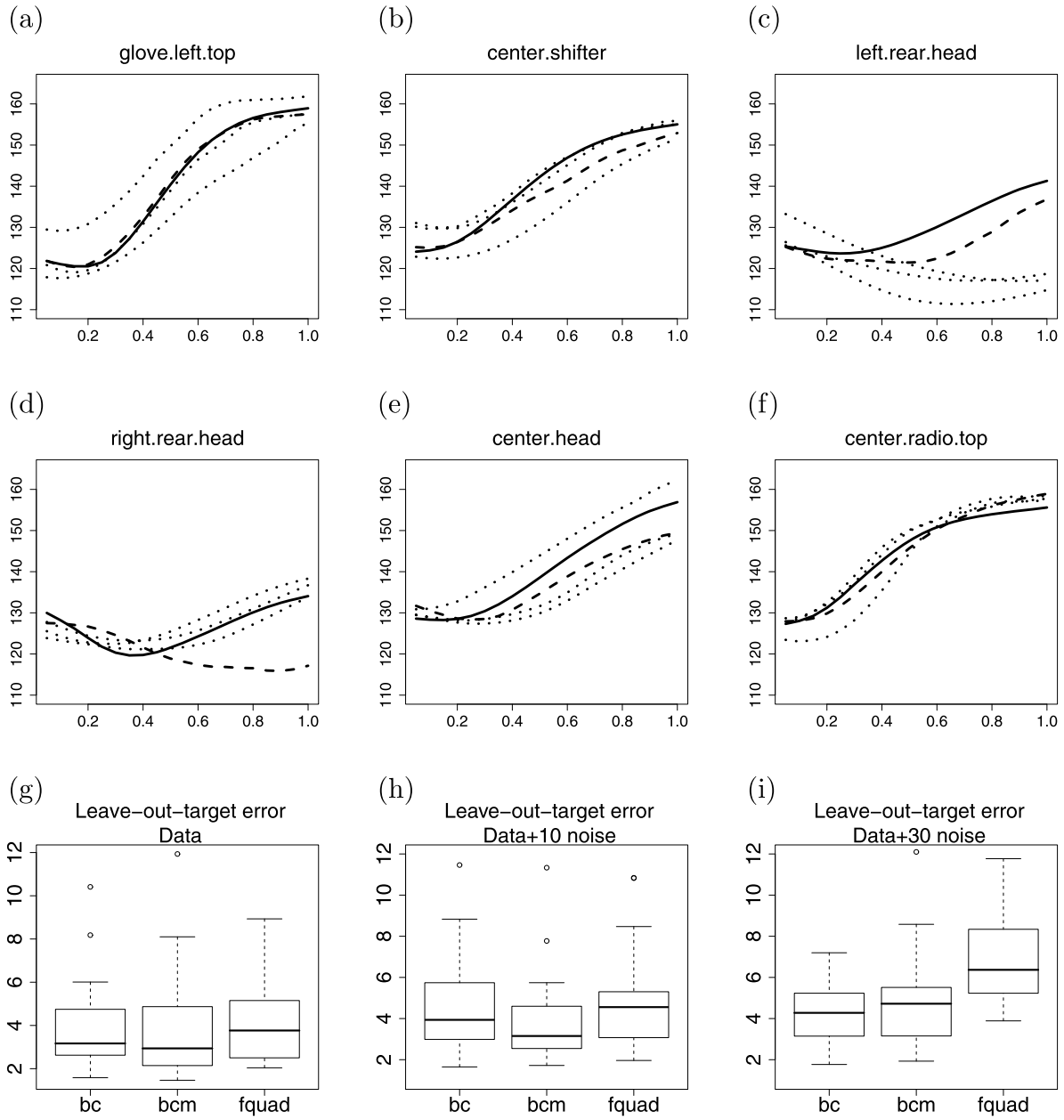

Fig. 3 Trajectory data, prediction and prediction error. Plots (**a**)–(**f**) show the trajectory data (*short-dashed lines*), the predictions from coefficient boosting model (*solid lines*), and the predictions from the quadratic model (*long-dashed lines*). Plots (**g**)–(**i**) show prediction error (mean absolute error) for leave-out-target validation

the prediction from the boosting algorithm is better than the quadratic model. Figure 3(g) shows the boxplot of the mean absolute error (averaged across the 20 sampled points) for the 20 leave-out-target validation. Here "bc" denotes coefficient boosting with univariate trees, "bcm" coefficient boosting with multivariate trees, and "fquad" the quadratic model of Faraway (1997). The models give similar results, with the boosting approaches giving somewhat lower error.

Next, we investigated the effects of adding redundant variables. Two data sets were formed, one by augmenting the original data with 10 normally distributed noise variables, the other with 30 normally distributed noise variables. These variables were entered as linear terms in the quadratic model since, adding them as second-order terms would lead to more parameters than observations at each time point. Note that this type of problem is not an issue when boosting tree models. Figure 3(h) and (i) show boxplots of the mean absolute error for the 20 leave-out-target validation. Not surprisingly, adding a larger number of noise variables considerably degrades performance of the quadratic model. The

error corresponding the coefficient boosting models also increases, but to a lower extent.

6 Discussion

Boosting in combination with tree models is a highly attractive methodology. The attractiveness stems from the ability of trees to cope with possibly high dimensional data of mixed types, and the ability of boosting to combine multiple trees to improve prediction error. This article has extended the gradient boosting algorithm of Friedman (2001) to situations where interest is on predicting a univariate function given a set of covariates. The experiments indicate that the approach performs well.

Two methods were proposed, one based on multivariate regression trees, the other on multiple univariate regression trees. The investigations suggest that the two give similar results, at least on the data sets considered. The multivariate tree approach is however more computationally efficient. At each iteration only a single regression tree is grown, as opposed to J of these with the multiple univariate tree approach. Furthermore, updating the terminal nodes of the regression tree requires only inverting a band 2 diagonal matrix, as opposed to one with bandwidth equal to $3 \cdot K$.

Coefficient boosting seems particularly attractive for modeling longitudinal data, viewing the response as repeated (possibly noisy) measurements of an underlying function. In many such situations, the mean response as a function of time is complicated and not easily modeled using linear methods. Having a more or less automated approach for estimating how the mean response over time depends on covariates is useful. However, we have only considered situations where the covariate vector is fixed. In many studies this is not the case, and the covariates vary with time. The manner in which such covariates influence the response may be complex. Extending our approach to such problems would be valuable.

Acknowledgements The authors thank the Associate Editor and two referees whose detailed comments greatly improved the manuscript. J.S. was supported by the Norwegian Cancer Society, grant HS02-2007-0154.

References

Bashtannyk, D.M., Hyndman, R.J.: Bandwidth selection for kernel conditional density estimation. Comput. Stat. Data Anal. **36**, 279–298 (2001)

Breiman, L., Friedman, J.H., Ohlsen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth, Belmont (1984)

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

de Boor, C.: A Practical Guide to Splines. Springer, New York (1978)

Eilers, P.H.C., Marx, B.D.: Flexible smoothing with B-splines and penalties. Stat. Sci. **11**, 89–121 (1996)

Faraway, J.J.: Regression with a functional response. Technometrics **39**, 254–261 (1997)

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

Graf, E., Schmoor, C., Sauerbrei, W., Schumacher, M.: Assessment and comparison of prognostic classification schemes for survival data. Stat. Med. **18**, 2529–2545 (1999)

Hall, P., Racine, J.S., Li, Q.: Cross-validation and the estimation of conditional probability densities. J. Am. Stat. Assoc. **99**, 1015–1026 (2004)

Hastie, T., Tibshirani, T., Friedman, J.: The Elements of Statistical Learning, 2nd edn. Springer, New York (2009)

Hayfield, T., Racine, J.S.: Nonparametric econometrics: the np package. J. Stat. Softw. **27**(5), 1–32 (2008)

Ishwaran, H., Kogalur, U.B.: randomSurvivalForest: Ishwaran and Kogalur's Random Survival Forest. R package version 3.6.1 (2010)

Ishwaran, H., Kogalur, U.B., Blackstone, E.H., Lauer, M.S.: Random survival forests. Ann. Appl. Stat. **2**, 841–860 (2008)

Leisch, F., Dimitriadou, E.: mlbench: Machine Learning Benchmark Problems. R package version 2.0-0 (2010)

Liaw, A., Wiener, M.: Classification and regression by randomForest. R News **2**, 18–22 (2002)

O'Sullivan, F.: Fast computation of fully automated log-density and log-hazard estimators. SIAM J. Sci. Stat. Comput. **9**, 363–379 (1988)

Peters, A., Hothorn, T.: ipred: Improved Predictors. R package version 0.8-8 (2009)

R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2009)

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

Ridgeway, G.: The state of boosting. Comput. Sci. Stat. **31**, 172–181 (1999)

Ridgeway, G.: gbm: Generalized Boosted Regression Models. R package version 1.6-3 (2007)

Stone, C.J., Koo, K.-Y.: Logspline density estimation. Contemp. Math. **29**, 1–15 (1986)

Weisberg, S.: alr3: Methods and data to accompany Applied Linear Regression, 3rd edn. R package version 1.1.12 (2009)