Università degli Studi di Napoli
Federico II

Generalized Boosted Additive Models

Sonia Amodio

Tesi di Dottorato di Ricerca in
Matematica per l'Analisi Economica e
la Finanza

XXIII Ciclo

Generalized Boosted Additive Models

Napoli, 30 novembre 2011

Ringraziamenti

É difficile in poche righe ricordare e ringraziare tutte le persone che mi sono state vicino durante questo mio percorso formativo.
Desidero ringraziare il mio tutor, il Dr. Antonio D'Ambrosio, per avermi sempre supportato durante questi anni.
Un ringraziamento particolare va alla Prof. Roberta Siciliano, che mi ha sempre fornito ottimi spunti di riflessione e preziosi consigli.
Grazie al Dr. Massimo Aria per avermi tante volte ascoltato, confortato e consigliato.
Grazie a tutti i membri del Dipartimento di Matematica e Statistica ed ai colleghi dottorandi.
Un ringraziamento particolare va alla Prof. Jacqueline Meulman che mi ha insegnato cosa vuol dire avere metodo e fare ricerca.
Un sentito ringraziamento va alla Dr. Anita van der Kooij, sotto la cui guida ho sviluppato le mie abilità di programmazione e a tutti i membri dell'Istituto di Matematica dell'Università di Leiden, per avermi accolto e fatto sentire come a casa.
Ma, sopratutto, un ringraziamento va alla mia famiglia che mi ha supportato e sopportato con infinita pazienza e amore in questi anni.

ai miei genitori

Contents

List of Tables

List of Figures

Introduction

This monograph is focused on nonparametric nonlinear regression and additive modeling.

Regression analysis is a central method of statistical data analysis. Linear regression concerns the conditional distribution of a dependent variable, Y, as a function of one or more predictors, or independent variables. The main characteristics of this model are its parametric form and the hypothesis that the underlying relationship between the outcome and the predictors is linear. For this reason this method is often inappropriate to model this relationship when it is characterized by complex nonlinear patterns and it can fail to capture important features of the data.
In such cases, nonparametric regression, which allows to determine the functional form between the dependent variable, Y, and the explicative variables by the data themselves, is more suitable. Hence, nonparametric methods become increasingly popular and apply to many area of research and practical problems. These methods show a great flexibility compared to parametric ones, but they also present an important drawback known as *curse of dimensionality*, which involves

that the precision of the estimates obtained via these methods is in inverse proportion to the number of explicative variables that are included in the model.

To overcome this problem Generalized Additive Models (GAM) were introduced. GAMs are based on the assumption that the conditional value of the outcome variable can be expressed as the sum of a certain number of univariate nonlinear functions, one for each predictor that is included in the model. One major concern to the use of the GAM is, therefore, when concurvity is present in the data. Concurvity can be defined as the presence of nonlinear dependencies among transformations of the explanatory variables considered in the model. One of the most common case of concurvity directly follows from the presence of collinearity among the untransformed predictors. In the context of generalized additive models the presence of concurvity leads to biased estimates of the model parameters and of their standard errors.

For such reasons we explore an alternative class of models, *CATREG*, based on the Regression with Transformation approach, applying the optimal scaling methodology as presented in the Gifi system. When we use this class of models in the presence of collinearity among untransformed predictors, applying nonlinear transformations through optimal scaling implies that interdependence among these predictor decreases.
Moreover in the framework of nonlinear regression with optimal scaling, we follow the approach proposed by Meulman (2003) of considering models in which, applying the basic idea of a *forward stagewise boosting procedure*, we introduce in the model nonlinear prediction components in a sequential way with the aim of improving the predictive power of the model itself. We call this approach the Generalized Boosted Additive Model (GBAM).

This monograph is structured as follows.
In first chapter we explore nonparametric regression models and their methodological framework. This chapter deals also with (Generalized) Additive Models, focusing on their advantages and limitations.
The second chapter is about nonlinear regression with optimal scaling and its theoretical context. In this chapter we focus on the fact that when collinearity is present in the data, applying nonlinear transformations via optimal scaling on the predictor results in decreasing the interdependence among them and we present some illustrations through real data analysis.
The third chapter is about Generalized Boosted Additive Models. After presenting their theoretical background, we show, through the use of simulations and the analysis of a real dataset, that in case of collinearity among predictors, which implies the presence of approximate concurvity among nonlinear transformations of these explicative variables, the proposed strategy leads to a solution that is a huge improvement compared to the linear model in terms of expected prediction error. At the same time, the use of prediction components in GBAM has the advantage of reducing the computational burden by decreasing the number of iterations required in each step of the procedure significanlty.

Chapter 1

Generalized Additive Models

1.1 Introduction

Regression analysis is a central method of statistical data analysis. By extension and generalization, it provides the basis for much of applied statistics.
Regression analysis concerns the conditional distribution of a response, or dependent variable, Y, as a function of several predictors, or independent variables. The object is to estimate the the regression coefficients $\{\beta_j\}_1^p$ of the model.
The expected value of the dependent variable is, for this reason, expressed as a linear combination of the predictors and the parameters in the model.

$$E(Y|\mathbf{x}) = \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p = \mathbf{x}^T \beta \tag{1.1}$$

where $E(y|\mathbf{x})$ is the expected value of y and depends on the particular realization of the vector $\mathbf{x}^T = (x_1, x_2, ..., x_p)^T$. If we define ϵ as the difference between the dependent variable Y and its conditional

expected value $E(Y|\mathbf{x})$:

$$\epsilon = Y - E(Y|\mathbf{x}) \tag{1.2}$$

we can write the model as:

$$Y = \mathbf{x}^T\beta + \epsilon \tag{1.3}$$

The main characteristics of this model are the *parametric form* (i.e. the regression function is completely determined by the unknown parameters, β_j) and the hypothesis of a linear relationship between the dependent variable and the predictors.

Given a sample, the estimation of the parameters in the model is usually obtained by least squares.

1.2 Nonparametric Regression

If we suppose that the relationship between the dependent variable and the predictors is completely described by a generic function $m(\cdot)$, which can be either linear or nonlinear, the regression model can be expressed in the following way:

$$E(Y|x_1, x_2, ..., x_p) = m(x_1, x_2, ..., x_p) \tag{1.4}$$

The model in equation (1.4) is known as *nonlinear regression*.

The object of nonparametric regression is to estimate the regression function $m(\cdot)$ directly, rather than to estimate parameters. Most methods of nonparametric regression implicity assume that $m(\cdot)$ is a smooth, continuous function.

The precision of the estimates obtained via this kind of models is in inverse proportion to the number of predictors which are included in the model. This problem is known as *curse of dimensionality* [2, 29]. The relationship between the dependent variable and the independent

variables can be graphically represented by a surface whose dimensions depend on the number of predictors that are included into the model. In general smoothing techniques used in nonparametric regression are based on the idea of locally averaging the data to obtain an estimate of the mean response curve.
Suppose that we want to estimate the following model:

$$E[Y|(x_1, x_2)] = m(x_1, x_2)$$

and suppose that $m(\cdot)$ is a smooth function.

These smoothing techniques give an estimate of the function $m(\cdot)$ in an arbitrary point $(x_1 = s, x_2 = e)$ using a *local weighted average* of the values of the dependent variable, Y, that correspond to some values of the independent variables that are situated in a small neighborhood of the arbitrary point that has coordinates equal to (s, e). This weighted average is characterized by a proximity concept: the values of Y receive a higher weight if the correspondent couple of values of x_1 and x_2 are closer to the point (s, e), otherwise they receive a lower weight. The outcome of this non-parametric model characterized by only two predictors will be the approximation of a scatterplot in a 3-dimensional space with a surface. Formally this *local weighted average* procedure can be defined as:

$$\hat{m}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} w_i(\mathbf{x}) Y_i \tag{1.5}$$

where w denotes a sequence of weights that may depend on the complete vector $\mathbf{x}^T$. Section 1.2 describes some of the most important smoothing methods that fall into the class of *linear* smoothers.

1.2.1 Smoothing methods

According to the definition in [55], an estimator $\hat{f}_n$ of f is a *linear smoother* if, for each x, there exists a vector $l(x) = (l_1(x), ..., l_n(x))^T$

such that

$$\hat{f}_n(x) = \sum_{i=1}^{n} l_i(x) Y_i \tag{1.6}$$

If we define the vector of fitted values as $\hat{\mathbf{f}} = (\hat{f}_n(x_1), ..., \hat{f}_n(x_n))^T$, where $\mathbf{y} = (Y_1, ..., Y_n)^T$, then follows that:

$$\hat{\mathbf{f}} = \mathbf{S}\mathbf{y} \tag{1.7}$$

where $\mathbf{S}$ is the $n \times n$ *smoothing matrix*, whose i-th row is $l(x_i)^T$, the *effective Kernel* for estimating $f(x_i)$, contains the weights given to each Y_i in forming the estimate $\hat{f}_n(x_i)$. Note that the smoother matrix, $\mathbf{S}$, depends on the dependent variables, as well as on the smoother, but not on Y. The trace of the smoothing matrix represents the degrees of freedom of the linear smoother. The simplest linear smoother is the **Regressogram** [77]. Suppose that all the predictor values are included in the interval (a, b), $a \leq x_i \leq b$, $i = 1, ..., n$. If we divide this interval into m equally spaced bins, b_j, $j = 1, ..., m$, and define

$$\hat{f}_n(x) = \sum_{i: x_i \in b_j} l_i(x) Y_i, \quad for\, x \in b_j, \tag{1.8}$$

where $l_i(x) = \frac{1}{n_j}$ if $x_i \in b_j$, 0 otherwise, and n_i is the number of points included into b_j, then the estimate $\hat{f}_n$ is a step function obtained by averaging the Y_i in each bin (note that in this case we are assigning equal weights to each observation that falls into a certain bin). The binwidth $h = \frac{(b-a)}{m}$ controls the smoothness of the estimate (the higher the h, the smoother the estimate).

A way to improve the estimate obtained from the regressogram is to consider overlapping regions, instead of disjoint and exhaustive regions. This is the main idea of two other simple smoothers, the **Running-mean** smoother and the **Running-line** smoother.

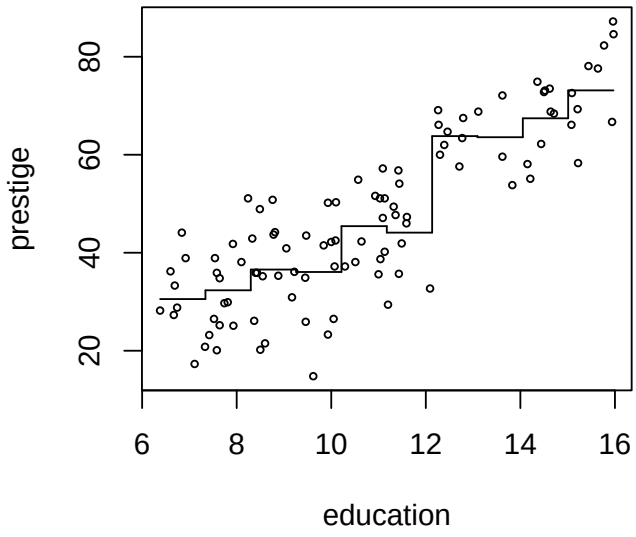

Figure 1.1: Example of regressogram: Prestige data from `R` package `car`. The number of bins is 10 and binwidth is equal to 0.96.

The **running-mean** smoother, also known as *nearest neighborhood*, produces a fit at the target point x by averaging the data points in a *neighborhood* N_i around x. Conversely to the regressogram, the width of the neighnorhood is variable and not fixed. In other words, the values of the dependent variable, Y, that are considered to calculate the mean, are those which correspond to the k values of the independent variable X that are closer to the target point. More formally,

$$\hat{f}(x_i) = \frac{1}{n} \sum_{j \in N(x_i)} Y_j. \tag{1.9}$$

The neighborhoods that are commonly used are *symmetric nearest neighborhoods* consisting of the nearest $2k + 1$ points:

$$N(x_i) = \{max(i - k, 1), ..., i - 1, i, i + 1, ..., min(i + k, n)\}. \tag{1.10}$$

Therefore the k parameters control the smoothness of the estimate: a large value of k will produce smoother curves, whereas a small value will produce more jagged estimates. We set $w = \frac{(2k+1)}{n}$, which represents the proportion of points that are included in each neighborhood. The proportion w is called the *span* and controls the smoothness of the estimate (the larger the span, the smoother the functions). Even though this smoother is simple in practice, it tends to be wiggly and to flatten out trends near the endpoints, so it can be severly biased.

A simple generalization of the running-mean smoother is the **running-line** smoother. This smoother fits a line by ordinary least squares to the data in a symmetric nearest neighborhood N_i around each x_i. The estimated smooth at x_i is the value of the fitted line at x_i:

$$\hat{f}(x_i) = \hat{\alpha}(x_i) + \hat{\beta}(x_i)x_i, \tag{1.11}$$

where $\hat{\alpha}(x_i)$ and $\hat{\beta}(x_i)$ are the coefficients obtained by ordinary least squares in the neighborhood of x_i. The parameter k that indicates

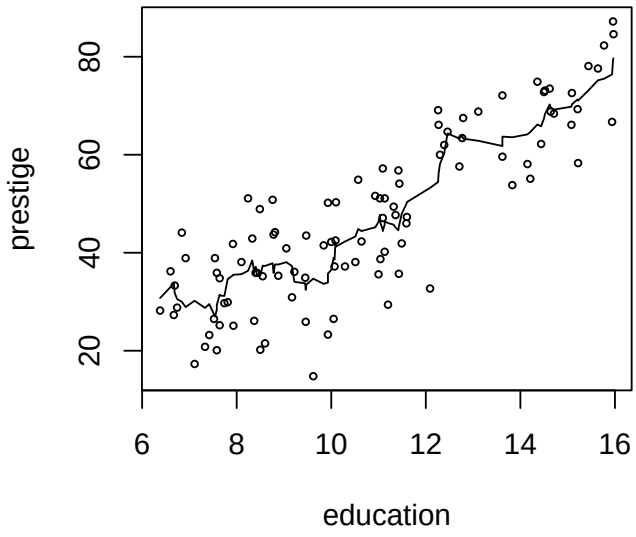

Figure 1.2: Example of symmetric running-mean: Prestige data from `R` package `car`. Span is equal to 0.088 (k = 4).

the number of points included in the neighborhood, as in the previous case, determines the shape of the estimate. Moreover, also in this case the span $w = \frac{(2k+1)}{n}$ indicates the proportion of points in each neighborhood. In the extreme case, if $w = 2$, each neighborhood contains all the data, the running-line smoother is the least square line, while if $w = \frac{1}{n}$, each neighborhood contains just one data point and the smoother interpolates the data. The running-line smoother is considered to be an improvement over the running-mean because it reduces the bias near the endpoints. But also this smoother, like the other examined up to this point, can produce jagged curves, because it assigns equal weights to all the points included in a given neighborhood and zero weight to points outside the neighborhood.

Differently from the smoothers presented up to this point, **Kernel** smoothers refine moving average smoothing through the use of a weighted average. In other word, they explicitly use a specified set of weigths W_i, defined by the *kernel*, to obtain an estimate at each target value: they describe the shape of the weight function by a density function with a scale parameter, h, that adjusts the size and the form of the weights near the target point. The kernel is a non-negative symmetric real integrable function K which satisfies:

$$\int_{-\infty}^{+\infty} K(u)du = 1.$$

More formally, the estimate given by a generic Kernel smoother can be written as:

$$\hat{f}_h(x) = \frac{\sum_i w_i y_i}{\sum_i w_i} \tag{1.12}$$

The distance from the target point is

$$d_i = \frac{x_i - x}{h}, \tag{1.13}$$

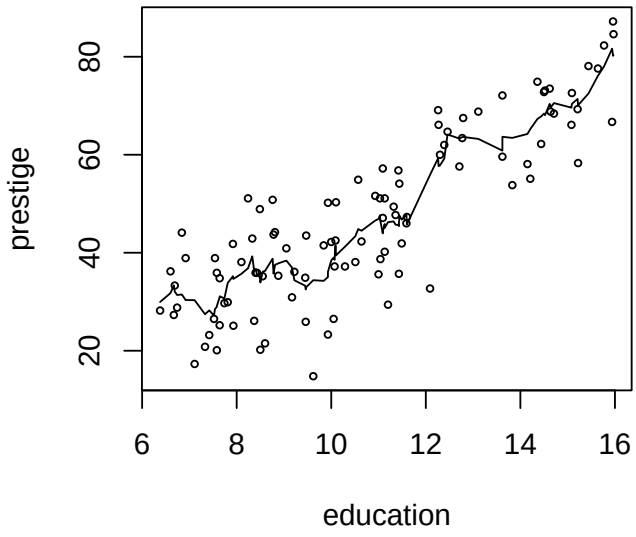

Figure 1.3: Example of symmetric running-line: Prestige data from `R` package `car`. Span is equal to 0.088 (k = 4).

and it measures the scaled and signed distance between the x-value for the i-th observation and the target point x. The scale factor h controls the bin width and, so, the smoothness of the estimate. Within each bin, the set of weights results from applying the kernel function to the distances calculated for the observations in the bin, $w_i = K\left[\frac{x_i - x}{h}\right]$. Then, these weights are used to calculate the local weighted average in equation (1.12). In Table 1.1 we show several kinds of kernel function which are commonly used [49]. The value u in Table 1.1 is equal to $u = (X - Xi)/h$ and $I(\cdot)$ is the indicator function.

The kernel specifies how the observations in the neighborhood of the

Kernel	$k(u)$
Uniform	$\frac{1}{2}\ I(\lvert u\rvert \leq 1)$
Triangular	$(1 - \lvert u\rvert)I(\lvert u\rvert \leq 1)$
Epanechnikov	$\frac{3}{4}(1 - u^2)I(\lvert u\rvert \leq 1)$
Quartic	$\frac{15}{16}\ (1 - u^2)I(\lvert u\rvert \leq 1)$
Triweight	$\frac{35}{32}\ (1 - u^2)^2 I(\lvert u\rvert \leq 1)$
Gaussian	$\frac{1}{\sqrt{2\pi}}\ \exp\left(-\frac{1}{2}u^2\right)$
Cosinus	$\frac{\pi}{4}\ \cos\left(\frac{\pi}{2}u\right)I(\lvert u\rvert \leq 1)$

Table 1.1: Kernel functions

target point, x, contribute to the estimate in that point. Whatever the weighting function is, the weights must have certain properties: (1) they must be symmetric with respect to the target point; (2) they must be positive, and (3) they must decrease from the target point to the bin boundaries.

Even though the kernel smoother represents an improvement with respect to the simple moving average smoother, it has a drawback: the

mean cannot be considered as an optimal local estimator, and using a local regression estimate instead of a local mean produces a better fit.

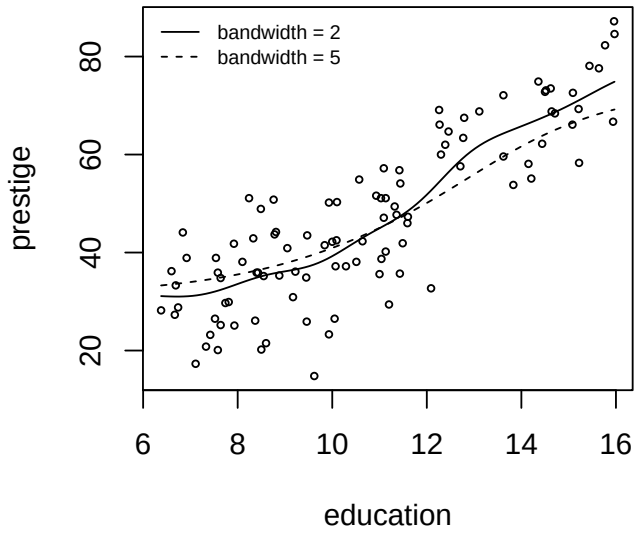

Figure 1.4: Example of Gaussian Kernel estimator: Prestige data from `R` package `car`.

Spline smoothers represent the estimate as a piecewise polynomial of a fixed order. Regions that define the pieces are separated by a sequence of knots (or breakpoints) and the piecewise polynomials are forced to joint smoothly in correspondence to these knots. For a given set of knots, the estimate is computed by multiple regression on a set of basis vectors which are the basis functions representing the

particular family of piecewise polynomials, evaluated at the observed values of the predictors. There are several types of spline smoothers: regression splines, cubic splines, B-splines, P-splines, natural splinse and smoothing splines, and many others. In the simplest regression splines, the piecewise functions are linear. In practice, we fit separate regression lines within the regions between the knots, and the knots tie together the piecewise regression fits. Also in this case, splines are a local model with local fits between the knots instead of within bins, and allow us to estimate the functional form from the data. Like in other smoothing functions we must make several decisions: we need to decide the degree of the polynomial for the piecewise function, the number of knots, and their placement. The evaluation of all the different spline smoothers is far beyond the aim of this monograph, for a wider coverage refer to [4, 17, 20, 55].

The **locally weighted running-line smoother** (LOWESS) [12] combines the local nature of running-line smoother and the smooth weights of the kernel smoother. The idea is to start with a local polynomial least-squares fit computed in different steps. In the first step the k nearest neighbors of the target point, x, are identified. Then, we compute the distance of the furthest near-neighbor from x, $\Delta(x)$. The weights w_i are assigned to each point in the neighborhood using the *tri-cube* weight function

$$W\left(\frac{\|x - x_i\|}{\Delta(x)}\right), \tag{1.14}$$

where

$$W(u) = \begin{cases} (1-u^3)^3 & \text{for} \quad 0 \leq u < 1; \\ 0 & \text{otherwise.} \end{cases} \tag{1.15}$$

Then, the estimate is the fitted value at x from the weighted least-squares fit of y to x in the neighborhood using the computed weights.

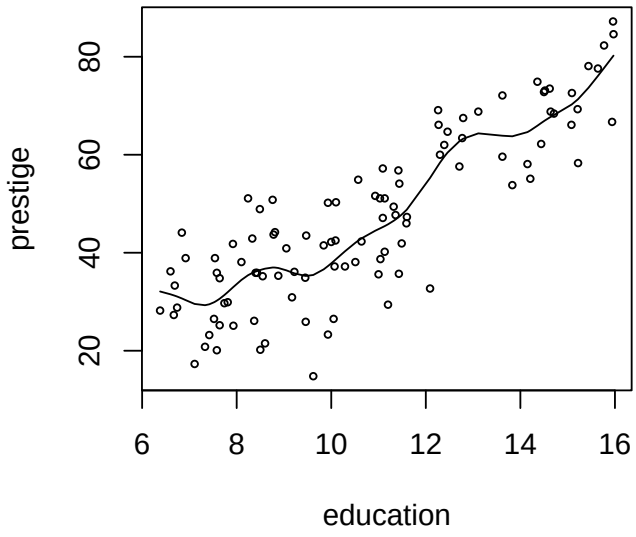

Figure 1.5: Example of cubic spline estimator: Prestige data from `R` package `car`. Smoothing parameter is 0.86

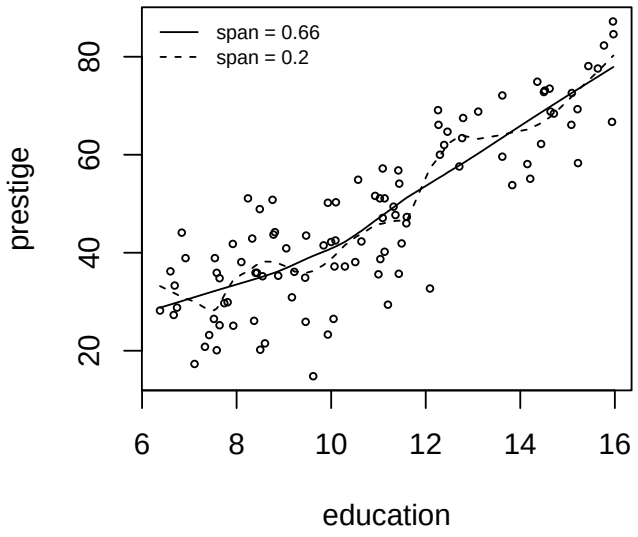

Figure 1.6: Example of Lowess estimator: Prestige data from `R` package `car`.

1.2.2 Span selection and the Bias-Variance Trade-off

A smoother is considered effective if it produces a small prediction error, usually measured by the Mean Squared Error (MSE).

$$MSE(\hat{f}_h(x)) = E\left(\hat{f}_h(x) - f(x)\right)^2 = Bias^2(\hat{f}_h(x)) + Var(\hat{f}_h(x)), \tag{1.16}$$

where

$$Bias = E\left[\hat{f}_h(x) - f(x)\right] \tag{1.17}$$

and

$$Var(\hat{f}_h(x)) = E\left(\hat{f}_h(x) - E\hat{f}_h(x)\right)^2. \tag{1.18}$$

We can see that the bandwidth controls the trade-off between the squared bias and the variance: when h is large, the squared bias is large but the variance is small and viceversa. Intuitively, when the size of the local neighborhood is small this neighborhood contains few observations and so the estimate closely approximates f. This results in a small bias of the $\hat{f}_h(x)$. However, since there are only few observations in the neighborhood, the variance of the estimate is large.

The smoothing parameter, from this point on indicated as λ, can be chosen by visual trial and error, picking a value that balances smoothness against the fit of the model.
Intuitively, a good smoothing parameter should produce a small average mean squared error:

$$ASE(\hat{f}_\lambda) = N^{-1}\sum_{i=1}^{N} MSE(\hat{f}_\lambda(x_i)) = N^{-1}\sum_{i=1}^{N}\left(\hat{f}_\lambda(x_i) - f(x_i)\right)^2, \tag{1.19}$$

but this quantity is not defined because f is an unknown function which has to be estimated. A good estimator for the ASE is the cross-validation score:

$$CV(\lambda) = N^{-1} \sum_{i=1}^{N} \left(\hat{f}_{\lambda}^{-i}(x_i) - y_i \right)^2, \tag{1.20}$$

where $\hat{f}_{\lambda}^{-i}(x_i)$ is the fitted value at point x_i if we use all the data points except (x_i, y_i).

The cross-validation procedure selects the λ that minimizes the score in equation (1.20). Moreover, for linear smoother the CV score can be simplified as,

$$CV(\lambda) = N^{-1} \sum_{i=1}^{N} \left(\frac{\hat{f}_{\lambda}(x_i) - y_i}{1 - S_{ii}} \right)^2 \tag{1.21}$$

where S_{ii} is the i-th diagonal element of the smoother matrix $\mathbf{S}$, see [55] for details. Repeating this procedure for different values of the smoothing parameter will suggest a value that minimizes the cross-validation estimate. Often, Generalized cross-validation, first proposed by [86] is used, where S_{ii} is replaced by its average value $tr(\mathbf{S})/N$ and, for this reason, easier to compute.

1.2.3 Curse of dimensionality

With increasing dimensionality p, these techniques suffer under the curse of dimensionality [2]. The relationship between the dependent variable and the independent variables can be graphically represented by a surface whose dimension depends on the number of predictors included in the model. The use of nonparametric estimators, which are for sure more flexible compared to those used in parametric models, is for this reason often used with complementary methods for the

reduction of dimensionality. This means that if we define a local neighborhood over which we want to average the data to obtain an estimate, then this neighborhood is most likely empty (i.e. has no observations in it). Vice versa, if we choose the neighbourhood such that it is not empty, then the estimate will be no longer local. But even if we could estimate the smooth function reliably, it is not clear how we can visualise the response curve (surface) for large number of predictors to gain the insight that we are looking for.
To overcome these two problems a class of models have been proposed known as (generalized) additive models [50, 76]. Here, we do not assume that the response curve f is a smooth p-variate function. Rather, the assumption is that f can be written as the sum of p univariate functions each of which has one predictor as argument.

1.3 Additive Models

A very useful generalization of the ordinary multiple regression

$$y_i = \alpha + \beta \mathbf{x}_i + \epsilon_i,$$

is the class of additive models:

$$y_i = \alpha + f(x_{1,i}) + ... + f(x_{p,i}) + \epsilon_i. \tag{1.22}$$

The form of the multiple regression model is relaxed: as in linear regression, the additive regression model specifies the expected value of Y as the sum of separate terms for each predictor, but now these terms are assumed to be smooth functions of the independent variables. Even in this case the model might have component functions with one or more dimensions, as well as categorical variable terms and their interactions with continuous variables. Obviously we assume that

$$E\{f_j(x_j)\} = 0,$$

otherwise there will be a free constant in each of the functions.
A substantial advantage of the additive regression model respect to nonparametric regression is that it eliminates the *curse of dimensionality*, as it reduces to a series of two-dimensional partial regression problems.
Moreover, since each variable is represented in a separate way the model has another important interpretative feature which is common to the linear model: the variation of the fitted response surface, holding all predictors constant except one, does not depend on the values of the other predictors. In other words, as each partial regression problem is a two-dimensional problem, we can estimate separately the partial relationship between the dependent variable and each predictor.

The model is fitted by iteratively smoothing partial residuals in a process known as **backfitting**, which is a block[1] Gauss-Seidel procedure for solving a system of equations. The idea of the backfitting algorithm goes back to Friedman and Stuetzle [38], who used it for projection pursuit regression, Breiman and Friedman [6], who employed it in their Alternating Conditional Expectation algorithm (ACE) and Young, De Leeuw and Takane [92], who used it in their alternating least squares algorithm (CORALS).
We can notice that in the additive model,

$$E\left[Y - \alpha - \sum_{j \neq k}^{p} f_j(x_j)|x_k\right] = f_k(x_k) \tag{1.23}$$

holds for any k, $1 < k < p$. This suggests the use of an iterative algorithm to calculate the f_j.
Given a set of initial estimates $\{\hat{\alpha}, \hat{f}_j\}$, we can improve these estimates iteratively (i.e. looping over $j = 1, ..., p$) by calculating the partial

[1]Backfitting constitutes a block Galuss-Seidel procedure for the fact that instead of solving for one element at each step we solve for n elements simultaneously

residuals from the observations $\{\mathbf{x}_j, Y_i\}$. Considering the partial residuals

$$r_i^{[1]} = y_i - \hat{\alpha} - \sum_{l \neq k}^{p} f_l(x_{il}), \tag{1.24}$$

and smoothing $r^{[1]}$ against x_j to update the estimate $\hat{f}_i$.
The backfitting algorithm for Additive model is sketched in Table 1.2.

---

1. Set the counter k to zero. Initialise $\hat{\alpha}$ and $\hat{f}_i$ as
   $\hat{\alpha} = \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$
   $\hat{f}_j(x_j) = 0 \quad \text{for} \quad j = 1, ..., p \quad \text{and} \quad \text{for } i = 1, ..., n$
2. For $j = 1, ..., p$ do:
   Calculate partial residuals: $r_i = y_i - \hat{\alpha} - \sum_{\substack{l=1 \\ l \neq j}}^{p} \hat{f}_l(x_{il}), \qquad i = 1, ..., n$
   Update the jth smooth function: $\hat{f}_j(\cdot) = S_j(w_i, x_{ij})r$
3. Check for convergence.
   If the algorithm has not converged yet, set $k = k + 1$ and go to 2.
   Else return.

Table 1.2: Estimating an additive model using the backfitting algorithm

The f_js are arbitrary univariate and smooth functions, one for each predictor.
A two-dimensional plot is sufficient to examine the estimated partial regression function $\hat{f}_j$ relating y to x_j holding the other explanatory variables constant. This means that interpretation of additive regression models is relatively simple, assuming that the additive model adequately captures the dependence of Y on the independent variables.

In other words, the backfitting algorithm solves the following system of estimating equations:

$$\begin{pmatrix} \mathbf{I} & \mathbf{S}_1 & \mathbf{S}_1 & \cdots & \mathbf{S}_1 \\ \mathbf{S}_2 & \mathbf{I} & \mathbf{S}_2 & \cdots & \mathbf{S}_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_p & \mathbf{S}_p & \mathbf{S}_p & \cdots & \mathbf{I} \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_p \end{pmatrix} = \begin{pmatrix} \mathbf{S}_1 Y \\ \mathbf{S}_2 Y \\ \vdots \\ \mathbf{S}_p Y \end{pmatrix} \tag{1.25}$$

where $\mathbf{I}$ is a $n \times n$ unit matrix. In a short form we can write:

$$\mathbf{Pf} = \mathbf{Q}Y \tag{1.26}$$

1.4 Generalized Additive Models

Generalized additive models represent a flexible extension of generalized linear models [59], allowing non-parametric smoothers in addition to parametric forms combined with a range of link functions and provide one way to extend the additive model. More specifically, the effects of the predictors are assumed to be linear in the parameters, but the distribution of the response variable, as well as the link between the predictors and this distribution, can be quite general.
At least two other estensions have been proposed: Friedman and Stuezle [38] introduced *Projection Pursuit Regression* and Breiman and Friedman [6] introduced *Alternating Conditional Expectation.*

A generalized linear models is specified by three components:

- a random component: we specify the distribution of the response variable and we assume that it comes from exponential family density,

$$f(Y, \theta, \phi) = exp\left\{\frac{Y\theta - b(\theta)}{a(\theta)} + c(Y, \phi)\right\}, \tag{1.27}$$

which includes many distributions that are useful for practival modelling, such as the Poisson, Binomial, Gamma and Normal distribution. The canonical parameter θ represents the location, while the dispersion parameter ϕ represents the scale of the exponential distribution taken into account. Moreover $a_i(\phi)$, (θ_i) and $c(y_i, \phi)$ are known functions. Generally $a_i(\phi)$ has the form $a_i(\phi) = \frac{\phi}{w_0}$, where w_0 is a known prior weight, usually equal to 1.

- a systematic component: we assume that the expected value of the response variable is related to the set of covariates by a linear predictor,
  $\eta = \beta\mathbf{X}$.

- a link function that describes how the expected value of the response variable is linked to covariates through linear predictor,
  $g(\mu) = \eta$.

Estimation and inference with generalized linear models is based on the theory of maximum likelihood estimation. For a single observation the log-likelihood is:

$$\log L(\theta_i, \phi, Y_i) = \left[\frac{Y_i\theta_i - b(\theta_i)}{a_i(\phi)}\right] + c(Y_i, \phi) \tag{1.28}$$

So for independent observations, the log-likelihood will be $\sum_i \log L(\theta_i, \phi, Y_i)$. We can maximize this analytically and find an exact solution for the MLE, $\hat{\beta}$, only if the response variable has a Gaussian density function, otherwise numerical optimization is required. McCullagh and Nelder [59] showed that the optimization is equivalent to **iteratively reweighted least squares** (IRWLS), which turns out to be equivalent to *Fisher's method of scoring*, which is simply the Newton-Raphson method with the Hessian replaced by its expected value.

Given a starting estimate of the parameters $\hat{\beta}$, we calculate the estimated linear predictor $\hat{\eta}_i = x_i'\hat{\beta}$ and use that to obtain the fitted values $\hat{\mu}_i = g^{-1}(\hat{\eta}_i)$. Then we calculate, using these quantities, the working dependent variable as

$$z_i = \hat{\eta}_i + (y_i - \hat{\mu}_i)\left(\frac{\partial \eta_i}{\partial \mu_i}\right) \tag{1.29}$$

The rightmost term in (1.29) is the derivative of the link function evaluated at the initial estimate. Next we calculate the iterative weights as:

$$w_i = \frac{1}{b''(\theta_i)\left(\frac{\partial \eta_i}{\partial \mu_i}\right)^2} \tag{1.30}$$

where $b''(\theta_i)$ is the second derivative of $b(\theta_i)$ evaluated at the starting estimate assuming $a_i(\phi) = \phi$. This weight is inversely proportional to the variance of the working dependent variable, given the current estimate of the parameters, with proportionality factor equal to ϕ. From this point on we obtain an updated estimate of the β, regressing the working dependent variable z_i on the predictors, using the weights w_i. In other words, we calculate the weighted least-square estimate

$$\hat{\beta} = (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}z \tag{1.31}$$

where $\mathbf{X}$ is the model matrix, $\mathbf{W}$ is a diagonal matrix of weights (with entries w_i) and z is the response vector, with entries z_i. This procedure is repeated until two successive estimates change less than a pre-specified small amount.

The linear predictor in GLM, $\eta = \beta\mathbf{X}$, specifies that the independent variables act in a linear way onto the response. According to [50], a more general model can be:

$$\eta = \alpha + \sum_{i}^{p} f_j(X_j) \tag{1.32}$$

where $f(\cdot)$s are smooth functions. As in the GLM the estimates are found by regressing repeatedly the adjusted dependent variable z on the predictors. In this 'smooth' version of the model in (1.32) it is possible to estimate the $f(\cdot)$ by repeatedly smoothing the adjusted dependent variable on X. The authors called this procedure **local scoring**. A general local scoring algorithm, as reported in [50], is sketched in Table 1.3. As shown in Buja, Hastie and Tibshirani [8], the backfitting algorithm will always converge. Being the local scoring simply a Newton-Raphson step, if the step size optimization is performed, it will converge as well. Note moreover that the convergence is monitored by a change in the fitted functions rather than in the deviance. Because the deviance is penalized, as analitically shown in [55], it can increase during the iterations, especially when the starting functions are too rough.

1.5 Degeneracy in GAMs: concurvity

As the term *collinearity* refers to linear dependencies among the independent variables, the term *concurvity* [8] describes the nonlinear dependencies among the predictor variables. In this sense, as collinearity results in inflated variance of the estimated regression coefficients, the result of the presence of concurvity will lead to instability of the estimated coefficients in GAM. As mentioned in [8]:

> "Concurvity boils down to collinearity of (nonlinear) transforms of predictors". [8], p484

Exact concurvity is defined as the existence of a nonzero solution, $\mathbf{g} = (g_1, ..., g_p)^T$, of the corresponding homogeneous system of equa-

1. Initialise $\hat{\alpha}$ and $\hat{f}_i$ as
   $\hat{\alpha} = G^{-1}(\bar{y}) = G^{-1}(\frac{1}{n}\sum_{i=1}^{n} y_i)$
   $\hat{f}_j^{(0)}(\cdot) = 0$

2. Set the counter k to zero. Iterate: $k = k + 1$

   - Form the adjusted dependent variable

     $$Z = \eta^{m-1} + (Y - \mu^{m-1})(\partial\eta/\partial\mu^{m-1}),$$

     where:
     $\eta^{m-1} = \alpha + \sum_{j=1}^{p} f_j^{m-1}(X_j)$ and
     $\eta^{m-1} = g(\mu^{m-1})$
   - Form the weights $W = (\partial\mu/\partial\eta^{m-1})^2 V^{-1}$.
   - Fit an additive model to Z usign the backfitting algorithm with weights W, so to obtain estimated functions $f_j^m(\cdot)$ and model η^m.
   - Compute the convergence criterion
     $\Delta(\eta^m, \eta^{m-1}) = \frac{\sum_{j=1}^{p} \|f_j^m - f_j^{m-1}\|}{\sum_{j=1}^{p} \|f_j^{m-1}\|}$

   Repeat step 2 until the convergence criterion is below some small threshold.

Table 1.3: General local scoring algorithm.

tions:

$$\begin{pmatrix} \mathbf{I} & \mathbf{S}_1 & \mathbf{S}_1 & \cdots & \mathbf{S}_1 \\ \mathbf{S}_2 & \mathbf{I} & \mathbf{S}_2 & \cdots & \mathbf{S}_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_p & \mathbf{S}_p & \mathbf{S}_p & \cdots & \mathbf{I} \end{pmatrix} \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_p \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{1.33}$$

If such $\mathbf{g}$ exists and $\mathbf{f} = (f_1, f_2, ..., f_p)^T$ is a solution of the system of normal equations in (1.25), then the system will have an infinite number of solution because for any c also $\mathbf{f} + c\mathbf{g}$ will be a solution. In other words, the concurvity space of (1.26) is the set of additive functions $\mathbf{g}(x) = \sum g_k(x_k)$ such that $\mathbf{Pg} = 0$.
As demonstrated in [55], concurvity is present when the spaces spanned by the eigenvectors of the smoothing matrices are linearly dependent. As demonstrated in [8], if we consider symmetric smoothers, for example cubic spline smoothers, exact concurvity will be present only in case of a perfect collinearity among the untransformed predictors.

Note that, in contrast to the linear regression framework where collinearity implies that the solution of the equation system cannot be found unless the data matrix is transformed in a full rank matrix or a generalized inverse is defined, the presence of concurvity does not imply that the backfitting algorithm will not converge. It has been demonstrated that backfitting algorithm will always converge to a solution. In case of concurvity the starting functions will determine which solution of (1.25) will be the final solution. While exact concurvity is highly unlikely, except in the case of symmetric smoothers with eigenvalues $[0, 1]$, since it can only derive from an exact collinearity among the original predictors, approximate concurvity is of practical concern, because it can lead to upwardly biased estimates of the parameters and to the underestimation of their standard errors.

1.6 An illustration

In this section we try to illustrate the effects of strong concurvity in fitting a generalized additive model with a simulated example. As mentioned before, concurvity is present in the data when the predictors are collinear. We simulate a total of $n = 200$ observations. We simulate three predictors (x, z and t) independently from a uniform distribution $U[0, 1]$. The forth predictor, g is generated as:

$$g = 3 \times x^3 + N(0, 0.0001)$$

to show strong concurvity with predictor x.

	Eigenvalues	Tolerance
x	1.93871	0.14917
t	1.07470	0.97570
z	0.90923	0.98721
g	0.07736	0.14856

Table 1.4: Eigenvalues and tolerance of independent variables

In Table 1.4 we report the eigenvalues of the correlation matrix of the four predictors and corresponding values of the tolerance. Obviously, there is a strong relation between x and g. Furthermore, we generate the outcome variable, y, as:

$$y = 3 \times e^{-x} + 1.3 \times x^3 + t + N(0, 0.01),$$

So, y is function only of x and t. Now we fit a generalized additive model on these data using the `R` package `gam` created by Hastie and Tibshirani (smoothing parameters estimates are determined via cross validation).

```
Call: gam(formula = y ~ -1 + s(x, df = 5) + s(t, df = 1) + s(z, df = 2) +
    s(g, df = 1), family = gaussian, data = datas)
Deviance Residuals:
     Min       1Q   Median       3Q      Max
-2.64432 -0.66797  0.02713  0.68381  2.38037

(Dispersion Parameter for gaussian family taken to be 0.9417)

    Null Deviance: 2166.579 on 200 degrees of freedom
Residual Deviance: 177.984 on 189.0097 degrees of freedom
AIC: 568.2313

Number of Local Scoring Iterations: 4

DF for Terms and F-values for Nonparametric Effects

             Df Npar Df  Npar F     Pr(F)
s(x, df = 5)  1       4 19.2320 2.743e-13 ***
s(t, df = 1)  1       0  0.2410   0.04080 *
s(z, df = 2)  1       1  1.5496   0.21474
s(g, df = 1)  1       2  6.5943   0.00178 **
---
Signif. codes:  0 Ś***Š 0.001 Ś**Š 0.01 Ś*Š 0.05 Ś.Š 0.1 Ś Š 1
```

The results obtained above clearly show that the nonparametric effects for variables x and g are significantly different from zero. This arises only from the concurvity between predictors x and g. In Figure 1.7 we show the graphical representation of the model we fit.

Moreover, in Figure 1.8 we show the effect of the true function $f(x) = 3 \times e^{-x} + 1.3 \times x^3$ on $U[0, 1]$.

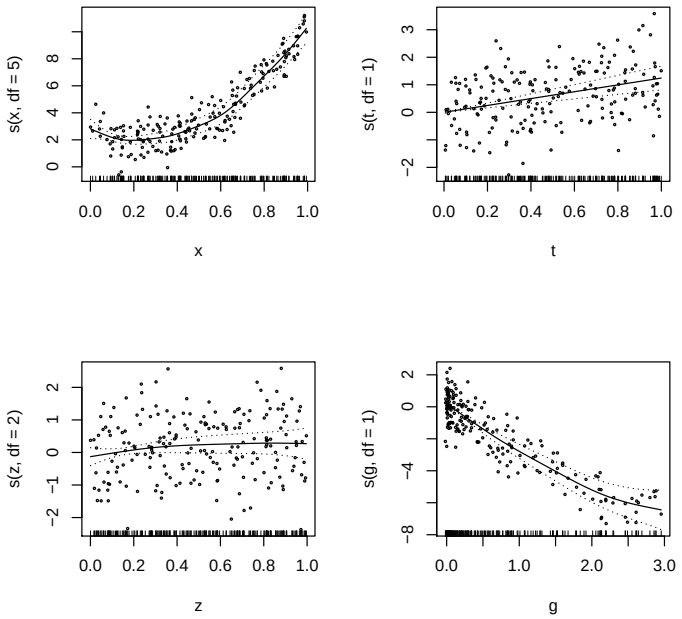

Figure 1.7: Additive model that relates the outcome variable to the predictors. Each plot represents the contribution of a term to the additive predictor. The 'y-axis' label represents the expression used to specify the corresponding contribution in the model formula.

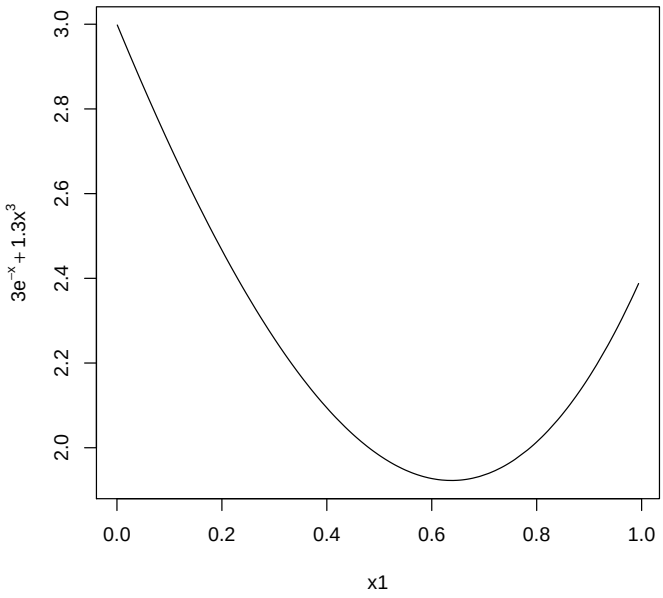

Figure 1.8: True function of the effect of predictor x on the outcome variable

Chapter 2

Nonlinear categorical regression

2.1 Introduction

In this chapter we will focus on the nonlinear categorical regression method, *CATREG*, which follows the Regression with Transformation approach, applying the *optimal scaling* methodology as presented in the so called *ALSOS* system [91] and in the Gifi system [39]. The ideas presented in this section are inspired by [64], in which the author argues that :

> 'the particular transformations resulting from the regression problem would have a particular influence on the structure of the correlation matrix between the predictors after optimal scaling' ([64]; pp 498)

As we know, if the predictors in the linear regression model are independent, their correlation matrix is equal to the identity matrix with all eigenvalues equal to 1. In contrast, if collinearity is present, the

predictors are highly linearly related and this influences the size of the distribution of the eigenvalues and the value of the small eigenvalues. When the correlation among the predictors increases, the value of the smaller eigenvalues decreases.

In the presence of multicollinearity, nominal and ordinal transformations obtained via optimal scaling linearize the relationship between the dependent variable and the predictors. In other words, the effect of these transformation is to decrease the interdependence among the predictors. This effect is stronger or weaker depending on the smoothness of the transformation itself (the smoother the transformation, the smaller the effect).

2.2 Optimal scaling

The optimal assignment of quantitative values to qualitative scales is an important development in multidimensional data analysis.
Optimal scaling represents a method to find an optimal transformation to convert categorical variables into numeric data. This transformation process is known as 'quantification' [91]. In a regression framework, quantifications of the categorical variables are estimated in parallel with the estimation of the regression coefficients, via an alternating least squares procedure that maximizes the correlation between transformations of the dependent variable and the set of predictors. The result of this procedure is that the optimal scaling transformations linearize the relationship between the dependent variable and the predictors. Numerical variables in this framework are treated as categorical ones, with a number of categories equal to the number of distinct values that the numeric variable presents. Of course optimality must be interpreted in a wide sense because it is obtained always with respect to the particular data set and to a particular criterion that is optimized.

In the quantification process it is possible to preserve properties of the data in the transformations by choosing an appropriate optimal scaling level for the variables. Note that, when we use the term optimal scaling level, we refer to the level on which the variable is analyzed and not to the measurement level of the original variable, which can be different.
The properties of the data that can be preserved are grouping, ordering and equal relative spacing.
According to its measurement level a variable can have one, two or all of these properties. We can distinguish:

- Variables with nominal measurement level: only grouping , i.e. the category values code the observations into the different classes
- Variables with ordinal measurement level: grouping and ordering, i.e. the category values code observations into the different classes and these classes are ordered .
- Numeric variable: grouping, ordering and equal relative spacing

To choose the scaling level, independent of the measurement level, we make the following distinction:

- We use a *nominal* scaling level when we want just to maintain the class membership information, i.e. objects in the same group according to variable j obtain the same quantification in the transformed variable $\varphi_j(\mathbf{x}_j)$.

$$x_{ij} = x_{i'j} \;\Rightarrow\; \varphi_j(x_{ij}) = \varphi_j(x_{i'j})$$

- We use an *ordinal* scaling level if a (categorical) variable contains order information on the objects and we want to preserve it in

the transformation. In this case x_j and $\varphi_j(x_j)$ are related by a monotonic function.

$$x_{ij} < x_{i'j} \;\Rightarrow\; \varphi_j(x_{ij}) \leq \varphi_j(x_{i'j})$$

- We use a *numeric* scaling level when we want to preserve all the properties. Note that if we use the numeric scaling level for a variable that is measured on a categorical level we treat the category values as numeric values, whereas if we use this scaling level for a numeric variable, this will result in a linear transformation to standard scores.

The scaling level is also related to the degrees of freedom of the transformation and to the fit of the model: transformations with less degrees of freedom will result in smoother transformations and worse fit and vice versa.
The transformation based on the nominal scaling level has the maximum number of degrees of freedom, which is equal to the number of categories minus one. Otherwise, the transformation which derives from choosing the ordinal scaling level implies one more restriction on the quantification of the order of thee categories, so the number of degrees of freedom is equal to the number of categories with different quantified values minus one.
Among the different transformations two approaches are available: **step functions** and **spline functions**.
Step functions are generally associated with categorical data, while *spline functions* refer to continuous data. Moreover, a continuous variable can also be considered as a categorical variable with a number of categories equal to the number of the objects. For this reason we need to limit the number of parameters we want to fit. For splines, the number of parameters is determined by the degree of the spline that we choose and the number of interior knots, thus we have to limit

both of them. In spline transformation case we consider monotonic and nonmonotonic spline.
Obviously the shape of the transformation is related to the number of degrees of freedom of the transformation itself. Transformations with more freedom will results in less smooth transformations and in a better fit and vice versa. In other words, this means that if we choose to preserve more properties of the data, using more restrictive transformations, we lose something in terms of fit of the model.

2.2.1 Monotonic splines

Following [69], *monotonic splines* are a class of piecewise polynomials. A polynomial spline is a piecewise polynomial function defined on an interval $[a, b]$ which is divided in a mesh consisting of points $a = \xi_1 < ... < \xi_q = b$. This mesh is also divided in subintervals $[\xi_j, \xi_{j+1})$ within which the function is a polynomial piece of specified order k.
Adjacent polynomial are required to join with a specified degree of smoothness at the boundaries of the subintervals. Smoothness is defined as the equality of the derivatives of the polynomial pieces at the joining points. In the common case, all orders of continuity, υ_j, are specified as the degree $k-1$ of the polynomial. For example, if $k = 2$ the spline consists of straightline segments that are required to match at the boundaries, whereas $k = 3$ the spline is a piecewise quadratic with matching first derivatives.
The domain and the continuity conditions are incorporated into the knot sequence, $t = \{t_1, ..., t_{n+k}\}$, where n represents the number of free parameters (total degrees of freedom) that specify the spline function. This sequence has the following properties:

1.
$$t_1 \leq ... \leq t_{n+k}$$

2. For all i there exists a j such that $t_i = \xi_j$.

3. The continuity characteristics are determined by:

   - $t_1 = ... = t_k = a$ and $b = t_{n+1} = ... = t_{n+k}$;
   - $t_i < t_{i+k}$ for all i;
   - if $t_i = \xi_j$ and $t_{i-1} < \xi_j$ then $t_i = ... = t_{i+k-\upsilon_j-1}$.

This means that the sequence of knots, t, is derived from the mesh by placing the number of knots at the boundary value according to the order of continuity desired. A spline of order $k-1$ is a polynomial at any point ξ and so it is determined by k free coefficients in the subinterval containing that point. But the continuity conditions impose υ_j linear equality constraints on the coefficients which define adjacent polynomials. So the total degrees of freedom is equal to the value n.
The spline transformation can be computed by defining a set of basis splines, $M_i(\cdot|k,t), i = 1, ..., n$ such that any piecewise polynomial, f, of order k and associated sequence of knots t, can be represented as a linear combination $f = \sum b_i M_i$.
In the Monotonic spline family the set of basis spline is defined to be positive in (t_i, t_{i+k}) and zero elsewhere, and must comply with the normalization property $\int M_i(x)dx = 1$ [16]. So, each M_i has the properties of a probability density function over the interval $[t_i, t_{i+k}]$. The monotonicity is assured by the nonnegativity of b_i.
Because monotonic splines are nonnegative we can define *integrated spline* as:
$$I_i(x|k,t) = \int_L^x M_i(u|k,t)du, \tag{2.1}$$

where L is the lower limit of the domain of the spline. Since each M_i is a piecewise polynomial of degree $k-1$, each I_i is a piecewise polynomial of degree k.
For a simple knot sequence, for which $t_j \leq x < t_j + k$ for all x, the I-spline I_i can be computed as:

$$I_i(x|k,t) = \begin{cases} 0, \ i > j, \\ \sum_{m=1}^{j} (t_{m+k+1} - tm) M_m(x|k+1,t)/(k+1), \ j-k+1 \leq i \leq j, \\ 1, \ i < j-k+1 \end{cases} \tag{2.2}$$

2.3 Nonlinear regression with optimal scaling

As mentioned before, in linear regression a dependent variable Y is predicted from a set of p independent variables $\mathbf{X}$. The aim of the regression is to find a linear combination of $\mathbf{X}$ that is maximally correlated with the dependent variable.
In terms of a least squares loss function we write:

$$L(\beta) = \|\mathbf{y} - \beta\mathbf{X}\|^2 = \|\mathbf{y} - \sum_{j=1}^{J} \beta_j \mathbf{x}_j\|^2 \tag{2.3}$$

We assume that the predictors are normalized to have zero mean and sum of squares equal to one, so we do not need to fit an intercept. The analytic solution of this problem is given by:

$$\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \tag{2.4}$$

where $(\mathbf{X}'\mathbf{X})^{-1}$ denotes the inverse of the correlation matrix between the independent variables.

If we include *optimal scaling* of the variables we substitute the independent variables with the one-to-one nonlinear transformations of the original variables. The nonlinear transformation of the j-*th* predictor is indicated as $\varphi_j(\mathbf{x}_j)$.

The loss function in (1) then can be rewritten as:

$$L(\beta, x) = \|\mathbf{y} - \sum_{j=1}^{p} \beta_j \varphi_j(\mathbf{x}_j)\|^2 \tag{2.5}$$

where $L(\beta, \mathbf{x})$ indicates that the arguments over which the function is to be minimized are the regression coefficients and the set of nonlinear transformations $x = \{\varphi_j(\mathbf{x}_j), j = 1, ..., p\}$.

If the independent variables are correlated in the regression problem, the optimal transformations $\varphi_j(\mathbf{x}_j)$ are also interdependent. To solve this problem we use a backfitting approach which separates each transformed variable and its weight from the rest of the weighted predictors, isolating the current target part $\beta_k \varphi_k(\mathbf{x}_k)$ from the rest, denoted as $\sum_{l \neq k} \beta_l \varphi_l(\mathbf{X}_l)$.
The loss function can be rewritten as:

$$L(\beta, \varphi(\mathbf{X}), \mathbf{y}) = \|\mathbf{y} - \sum_{l \neq k} \beta_l \varphi_l(\mathbf{x}_l) - \beta_k \varphi_k(\mathbf{x}_k)\|^2 \tag{2.6}$$

This means that we are changing the original multivariate problem into a univariate one.
If we define as auxiliary variable u_k:

$$\mathbf{u}_k = \mathbf{y} - \sum_{l \neq k} \beta_l \varphi_l(\mathbf{x}_l), \tag{2.7}$$

we have to minimize:

$$L(\beta_k, \varphi_k(\mathbf{x}_k)) = \|\mathbf{u}_k - \beta_k \varphi_k(\mathbf{x}_k)\|^2 \tag{2.8}$$

which is a function of β_k and $\varphi_k(\mathbf{x}_k)$ only.
Using **alternating least squares**, we minimize over β_k and $\varphi_k(\mathbf{x}_k)$ consecutively.
As the variable $\varphi_k(\mathbf{x}_k)$ is standardized, we can compute the regression weight β_k separately from the transformation.
The new value for the regression weight β_k is found as:

$$\hat{\beta}_k = \mathbf{u}_k' \varphi_k(\mathbf{x}_k) \tag{2.9}$$

After having fixed the new weight β_k with respect to the fixed values $\mathbf{u}_k$ and $\varphi_k(\mathbf{x}_k)$, we minimize the loss function over $\varphi_k(\mathbf{x}_k)$ with respect to the fixed $\mathbf{u}_k$ and $\hat{\beta}_k$. Using the new value of β_k, we minimize the loss function over all $\varphi_k(\mathbf{x}_k)$ over the cone that contains all admissible transformations of the variable $\mathbf{x}_k$, $\mathbb{C}_k$.

For each categorical variable $\mathbf{x}_k$ we search a vector of quantification $\mathbf{v}_k$ which minimizes the overall value of the associated loss function,

$$L(\beta, \varphi_k(\mathbf{x}_k)) = \|\mathbf{u}_k - \beta_k \mathbf{G}_k \mathbf{v}_k\|^2 \tag{2.10}$$

where $\mathbf{G}_k$ represents the indicator matrix associated with the k-th predictor. The number of different categories in the variable x_k are associated with the columns of this matrix and for each of these column a $0-1$ coding registers the presence-absence of the object in that particular category. In case of a spline transformation we construct an I-spline basis matrix $\mathbf{S}_k$ of $\mathbf{x}_k$ and we minimize

$$L(b_k) = \|\mathbf{u}_k - \beta_k \mathbf{S}_k(\mathbf{x}_k b_k)\|^2, \tag{2.11}$$

where $b_k = \{b_t^k, t = 1, ..., T\}$ is the T-vector with spline coefficients that have to be estimated, and T depends on the degree of the spline and the number of interior knots. In this last case the problem is further partitioned by separating the t-th column of the spline basis matrix $\mathbf{S}_k$, denoted by $\mathbf{s}_t^k$, from the other columns $\{\mathbf{s}_r^k, r \neq t\}$ and the

t-th element (b_t^k) of the spline coefficient vector $\mathbf{b}_k$ from the remaining elements $\{b_r^k, r \neq t\}$. If the I-spline transformation is required to be monotonic we have also to include this restriction in the model which implies that the the spline coefficients must be non-negative. Then we minimize iteratively:

$$L(b_t^k) = \|(\mathbf{u}_k - \beta_k \sum_{r \neq t} b_r^k \mathbf{s}_r^k) - \beta_k b_t^k \mathbf{s}_t^k\|^2 \tag{2.12}$$

When both β_k and $\varphi_k(\mathbf{x}_k)$ are updated, we move to the next regression weight and variable to be transformed. When all coefficients and variable transformations have been updated, we move to the outcome variable for which we may apply a similar set of transformation options as the ones described above for the predictor variables. A sketch of the resulting algorithm in the case of a spline transformation is presented in Table 2.1.

2.4 An illustration

The dataset analyzed in this section is called the Boston Housing dataset. It was collected by Harrison and Rubingeld (1978) an it concerns housing values in suburbs of Boston. This dataset contains 506 instances on 14 variables (13 continuous variable and a binary one) and there are no missing values. These variables are reported in Table 2.2. The dependent variable is MEDV which indicates the median value of owner-occupied homes in $1000's.

As a first step, we start with a standard linear regression analysis. When we use categorical regression and we decide that all the transformations have to be linear we obtain the same result of a linear multivariate regression model. This first step is useful to have an idea about the expected result when we consider the linearity hypothesis and also to inspect the plots of the residuals against the predictors and versus each predictor in turn. Moreover, when we look at the plots of the standardized partial residuals against each predictor, this shall be indicative of the most appropriate non-linear transformation to apply in a next step. In Table 2.3 we show the eigenvalues of the correlation matrix of the predictors and the values of the tolerance for each predictor. Tolerance, in the linear regression framework, is a measure for detecting collinearity. It can be formally expressed as

$$t_j = 1/r_{jj}^{-1} \tag{2.14}$$

where r_{jj} is the jth diagonal element of the inverse of the correlation matrix of the predictors, $\mathbf{R}$. Since the standard errors of the estimated parameters of the predictors depend in inverse proportion on the tolerance, small values of the tolerance cause large standard errors of the estimates of the regression coefficients, large confidence intervals and, likely, not significant test results. Note also that the Pearson correlation coefficient (or zero-order correlation coefficient) has low power in detecting collinearity, because it is sensitive to outliers presence and it

1. Normalize response variable and predictor variables to obtain
   $Y \Rightarrow \vartheta(Y)$
   $\mathbf{X} \Rightarrow \varphi(\mathbf{X})$

2. Initialize regression coefficients $\beta_1, ..., \beta_p$

3. Initialize spline coefficients $b_1, ..., b_T$

4. for k in $1:p$ minimize the loss function:

$$L(\beta_k, \varphi_k(\mathbf{x}_k)) = \|\mathbf{u}_k - \beta_k \varphi_k(\mathbf{x}_k)\|^2,$$

   - for t in $1:T$ minimize the loss function

$$L(b_k^t) = \|\mathbf{u}_k - \beta_k \left(\sum_{r \neq t} S_r^k(\mathbf{x}_k) b_r^k - S_k(\mathbf{x}_k) b_k \right) \|^2,$$

     to obtain the estimates of the spline coefficients, imposing the normalization condition $b_k' S_k' S_k b_k = N$, until the decrease in the loss function $L(b_k^t)$ is smaller than some pre-specified value.

5. Update $\varphi_k(x_k)$ as

$$\hat{\varphi}_k(x_k) = S_k(\mathbf{x}_k)\hat{b}_k \tag{2.13}$$

6. Minimize $L(\beta_k) = \|\mathbf{u}_k - \beta_k \hat{\varphi}_k(\mathbf{x}_k)\|^2$ to obtain the estimates of the regression coefficients.

7. Repeat step 4 until the decrease in the loss function $L(\beta, \varphi(x))$ is smaller than some pre-specified value.

Table 2.1: Backfitting algorithm for nonlinear regression with optimal scaling

Label	Description
CRIM	per capita crime rate by town
ZN	proportion of residential land zoned for lots over 25,000 sq.ft.
INDUS	proportion of non-retail business acres per town
CHAS	Charles River adjacency
NOX	nitric oxides concentration (parts per 10 million)
RM	average number of rooms per dwelling
AGE	proportion of owner-occupied units built prior to 1940
DIST	weighted distances to five Boston employment centres
RAD	index of accessibility to radial highways
TAX	full-value property-tax rate per \$10,000
PTRATIO	pupil-teacher ratio by town
B	$(Bk - 0.63)^2$, where Bk is the proportion of blacks by town
LSTAT	% lower status
MEDV	median value of owner-occupied homes in \$1000's

Table 2.2: Boston Housing dataset

also cannot detect collinearity due to the presence of a high correlation between a predictor and a combination of other predictors. It is well known that, when all the explanatory variables are independent, the correlation matrix is equal to the identity matrix, with all eigenvalues equal to one. Departures from independence thus can be indicated by the largest eigenvalues greater than one.
We notice that the Boston Housing dataset seems to be slightly affected by collinearity, for example if we look at the values of the tolerance for predictor RAD and TAX that are just greater than 0.1. Collinearity us also indicated by the largest eigenvalue being 6.13 and the smallest eigenvalue being 0.06.

	Eigenvalues	Tolerance
CRIM	6.12685	0.55798
ZN	1.43328	0.43502
INDUS	1.24262	0.25053
CHAS	0.85758	0.93110
NOX	0.83482	0.22760
RM	0.65741	0.51713
AGE	0.53536	0.32249
DIS	0.39610	0.25278
RAD	0.27694	0.13361
TAX	0.22024	0.11101
PTRATIO	0.18601	0.55584
B	0.16930	0.74155
LSTAT	0.06351	0.33996

Table 2.3: Eigenvalues of the correlation matrix and tolerance of the predictors

The estimates of the regression coefficients, the β_j's, and their standard errors (calculated via bootstrap resampling (1000)), are reported

	Beta	Estimate of Std Error	F	Sig.
CRIM	-0.098	0.036	7.540	0.009
ZN	0.116	0.036	10.742	0.001
INDUS	0.018	0.037	0.226	0.746
CHAS	0.073	0.035	4.327	0.030
NOX	-0.223	0.046	23.314	0.000
RM	0.299	0.062	23.467	0.000
AGE	-0.002	0.050	0.001	0.931
DIS	-0.335	0.045	56.484	0.000
RAD	0.287	0.062	21.431	0.000
TAX	-0.229	0.055	17.610	0.000
PTRATIO	-0.224	0.028	65.552	0.000
B	0.093	0.029	10.203	0.001
LSTAT	-0.402	0.072	30.879	0.000

Table 2.4: Regression coefficients of the model which considers only numerical transformations of all predictors

in Table 2.4. The Apparent Prediction Error (APE) for the training set and the Expected Prediction Error (EPE) for the test set, obtained by evaluating the Mean Square Error estimates with 10-fold cross-validation, for this first model are respectively equal to

$$APE = 0.259 \quad EPE = 0.283.$$

Note that using standardized coefficients, their interpretation is based on the standard deviation of the variables.
Each coefficient indicates the number of standard deviations that the predicted response changes for a one standard deviation change in a predictor, all the other predictors remaining constant. For example, a one standard deviation change in ZN leads to an increase in predicted MEDV of 0.116 standard deviations. The standard deviation of raw values of ZN is 23.322, so our outcome variable increases by $0.116 \times 23.322 = 2.745$.
We note that the regression coefficient for the predictor AGE is quite close to zero and it is not significantly different from zero according to the F test. For this reason we can try to consider another type of transformation for that predictor, different from the linear one. For example, if we consider a nominal transformation, or more precisely a nonmonotonic spline transformation of order two with two interior knots, for the explicative variable AGE, we obtain different results that are summarized in Table 2.5.

In this way the regression coefficient for the independent variable AGE becomes significantly different from zero. In Figure 2.1, we can see the standardized partial residuals, obtained as the difference between the standardized outcome and the fitted values calculated considering all the predictors except AGE, plotted against the standardized values of the variable AGE. The line in Figure 2.1 represents the standardized nominal transformation of predictor AGE. In Figure 2.2, the same standardized linear partial residuals are plotted against the standardized nominal transformation of AGE. In this figure the line

	Beta	Estimate of Std Error	df	F	Sig.
CRIM	-0.110	0.034	1	10.467	0.001
ZN	0.094	0.037	1	6.454	0.012
INDUS	-0.014	0.038	1	0.136	0.716
CHAS	0.078	0.034	1	5.263	0.022
NOX	-0.235	0.048	1	23.969	0.000
RM	0.291	0.061	1	22.758	0.000
AGE	0.082	0.042	4	3.812	0.005
DIS	-0.332	0.046	1	52.091	0.000
RAD	0.304	0.063	1	23.284	0.000
TAX	-0.237	0.053	1	19.996	0.000
PTRATIO	-0.215	0.027	1	63.409	0.000
B	0.081	0.029	1	7.801	0.005
LSTAT	-0.429	0.070	1	37.559	0.000

Table 2.5: Coefficients of the model considering a nominal spline transformation (2,2) for predictor AGE and numerical transformation for all the other predictors

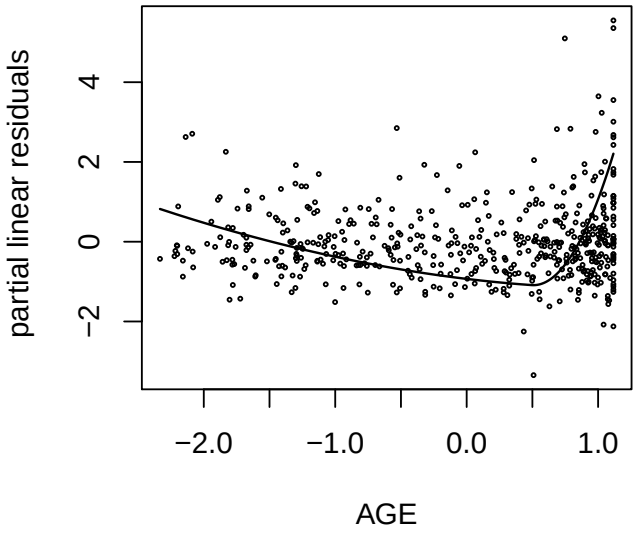

Figure 2.1: Standardized partial linear residuals versus standardized predictor AGE. The line represents the standardized nominal transformation of predictor AGE

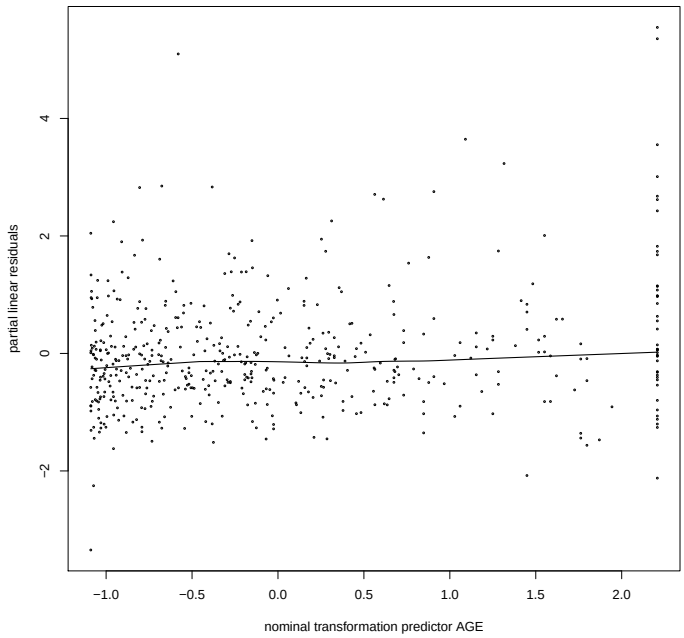

Figure 2.2: Standardized partial linear residuals versus standardized transformation of predictor AGE.

represents a lowess smoother that is used in order to detect any departure from linearity.
As we can see from Figure 2.2 the relationship between the partial linear residuals and the transformed predictor AGE seems to be linearized by using a nominal transformation. The APE and EPE for the second model are

$$APE = 0.254 \quad EPE = 0.280,$$

thus both values decrease with respect to a linear transformation of AGE. APE decreases per definition, but it is important that EPE decreases as well. Then we consider a nominal transformation also for the predictor INDUS, because its regression coefficient also was not significantly different from zero in the first two models we considered.

In Table 2.6 we report the regression coefficients of the model in which we consider for both AGE and INDUS a nominal transformation and linear transformations for all the other predictors. For this model we have that

$$APE = 0.245 \quad EPE = 0.272,$$

which is satisfactory.

In Figure 2.3 we show the standardized partial residuals plotted against the standardized predictors AGE and INDUS, while in Figure 2.4 the standardized partial residuals are plotted against the standardized transformations of these predictors. We can deduce from the latter figure that both relationships have been linearized.
Moreover, in Table 2.7 we can notice that by applying these nominal transformations we improve the values of the tolerance, not only for AGE and INDUS, but also for others predictors, even if to a lesser extent. Besides, the smaller eigenvalue of the correlation matrix of the transformed predictors (0.073) is slightly greater than the smallest eigenvalue of the correlation matrix before applying transformations

	Beta	Estimate of Std Error	df	F	Sig.
CRIM	-0.113	0.033	1	11.725	0.001
ZN	0.023	0.041	1	0.315	0.538
INDUS	0.129	0.034	4	14.395	0.000
CHAS	0.087	0.034	1	6.548	0.012
NOX	-0.249	0.048	1	26.910	0.000
RM	0.272	0.063	1	18.640	0.000
AGE	0.085	0.039	4	4.750	0.004
DIS	-0.352	0.045	1	61.187	0.000
RAD	0.372	0.082	1	20.581	0.000
TAX	-0.290	0.062	1	21.878	0.000
PTRATIO	-0.220	0.030	1	53.778	0.000
B	0.081	0.027	1	9.000	0.002
LSTAT	-0.428	0.073	1	34.375	0.000

Table 2.6: Coefficients of the model considering a nominal transformation for predictor AGE and INDUS and numerical transformation for all the others predictors

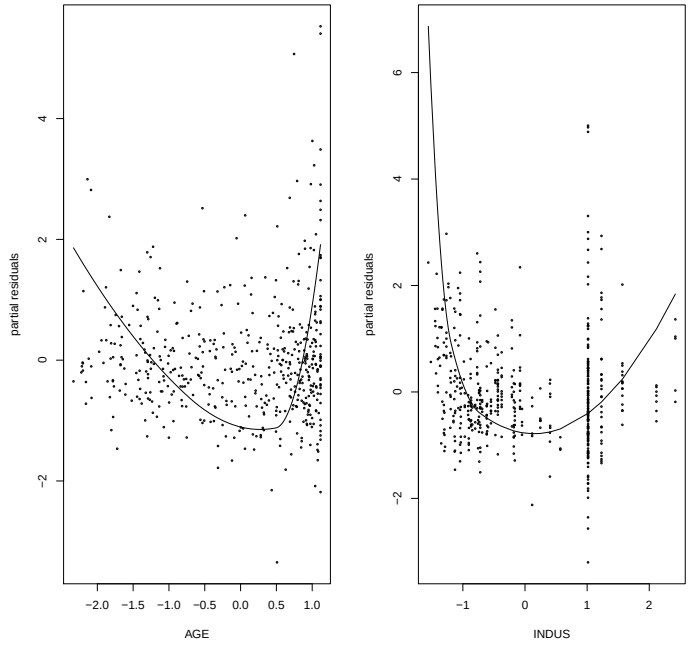

Figure 2.3: Standardized partial linear residuals versus standardized predictors, AGE and INDUS. The lines in these plots represent the standardized nominal transformations of AGE and INDUS, respectively.

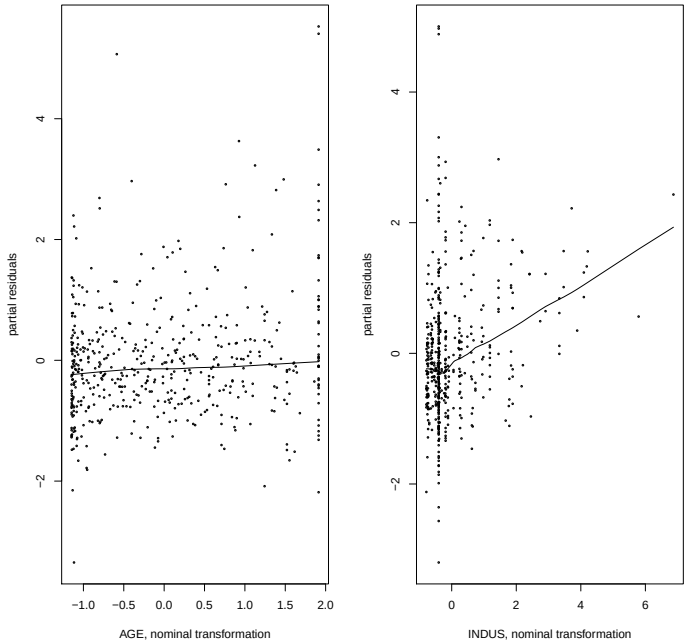

Figure 2.4: Standardized partial linear residuals versus standardized transformations of predictors AGE and INDUS.

Before nominal transformations			After nominal transformations	
	Eigenvalues	Tolerance	Eigenvalues	Tolerance
CRIM	6.12685	0.55798	5.16203	0.54209
ZN	1.43328	0.43502	1.56822	0.34876
INDUS	1.24262	0.25053	1.27068	0.57163
CHAS	0.85758	0.93110	1.06598	0.93637
NOX	0.83482	0.22760	0.83460	0.25961
RM	0.65741	0.51713	0.76746	0.52848
AGE	0.53536	0.32249	0.62888	0.79491
DIS	0.39610	0.25278	0.52107	0.28968
RAD	0.27694	0.13361	0.45042	0.13515
TAX	0.22024	0.11101	0.27649	0.13066
PTRATIO	0.18601	0.55584	0.21113	0.56868
B	0.16930	0.74155	0.17046	0.73647
LSTAT	0.06351	0.33996	0.07259	0.38028

Table 2.7: Eigenvalues of the correlation matrix and tolerance values before and after applying nominal transformations on AGE and INDUS

(0.063). Also the largest eigenvalue after applying these nominal transformations (5.16) is smaller than the one we had before, (6.12). Just for illustrative purposes, we see what happens if we consider an additional ordinal transformation for the predictor LSTAT, which presents the highest absolute value of the regression coefficient (-0.428).

	Beta	Estimate of Std Error	df	F	Sig.
CRIM	-0.137	0.022	1	38.779	0.000
ZN	-0.012	0.040	1	0.090	0.742
INDUS	0.105	0.031	4	11.472	0.000
CHAS	0.058	0.030	1	3.738	0.036
NOX	-0.213	0.041	1	26.989	0.000
RM	0.172	0.053	1	10.532	0.001
AGE	0.065	0.043	4	2.285	0.020
DIS	-0.280	0.042	1	44.444	0.000
RAD	0.371	0.069	1	28.910	0.000
TAX	-0.250	0.053	1	22.250	0.000
PTRATIO	-0.214	0.028	1	58.413	0.000
B	0.077	0.023	1	11.208	0.001
LSTAT	-0.587	0.064	3	84.123	0.000

Table 2.8: Coefficients of the model considering a nominal transformation for AGE and INDUS, and an ordinal transformation for LSTAT (numerical transformation for all others predictors)

As for the previous cases, we show in Table 2.8 the estimated coefficients for this model. In Figure 2.5 the standardized partial residuals are plotted against the standardized predictors and in Figure 2.6 against the standardized transformations of the predictors.
For this model we have that

$$APE = 0.202 \quad EPE = 0.221.$$

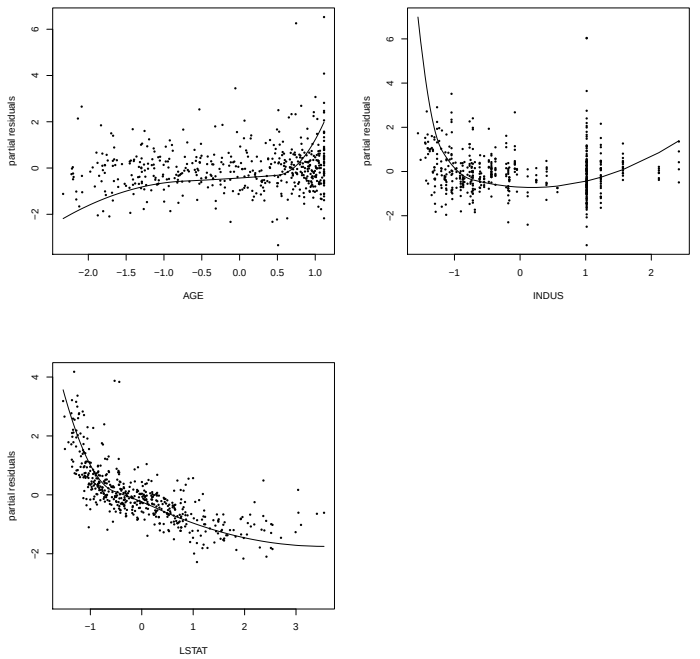

Figure 2.5: Standardized partial residuals versus standardized predictors, AGE, INDUS and LSTAT. The lines in these plots represent, respectively, the standardized nominal transformations for predictors AGE and INDUS, and standardized ordinal transformation for predictor LSTAT.

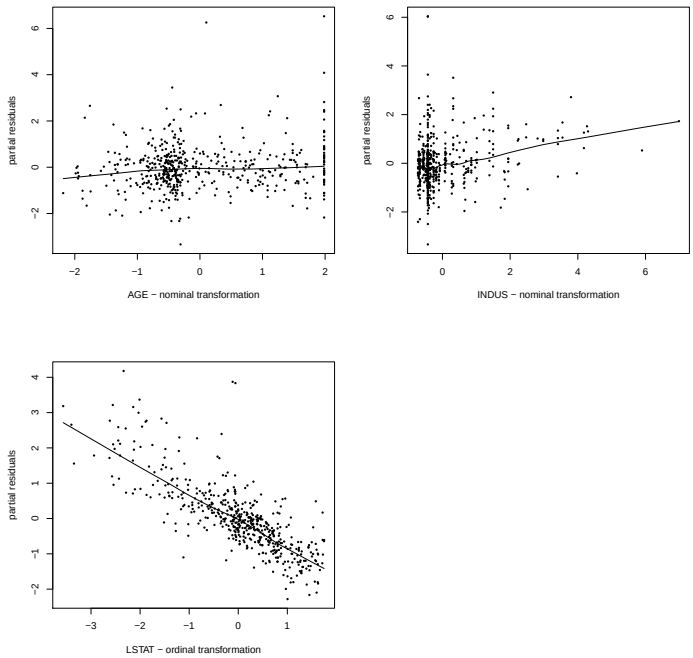

Figure 2.6: Standardized partial residuals versus standardized transformations of AGE(nominal), INDUS(nominal) and LSTAT(ordinal).

This clearly demonstrates that by choosing transformations different from the linear ones we can improve the expected prediction error considerably.

Chapter 3

Generalized Boosted Additive Models

3.1 Introduction

The ideas presented in this chapter are inspired by [64], in which the author, following the ideas of a *forward stagewise boosting* procedure [26, 34], proposed to consider the fit of prediction component in a sequential way with the aim of improving the predictive power of the mode.

3.2 Generalized Boosted Additive Models

What we will call a *generalized boosted additive model* is based on prediction components that consists of different nonlinear transforma-

tions of the predictors. We define a *prediction component* as a linear combination of optimally transformed predictors. More formally,

$$f = \sum_{k=1}^{p} \beta_k \varphi_k(\mathbf{x}_k). \tag{3.1}$$

We can express a *forward stagewise additive model* as

$$L(X) = \|y - f_m(X)\|^2 \tag{3.2}$$

where f_m can be defined as

$$f_m(X) = f_{m-1} + \sum_{k=1}^{p} \beta_{km} \varphi_{km}(\mathbf{x}_k), \quad m = 1, ..., M. \tag{3.3}$$

This is in line with [53], pp 304-305. In other words, we consider M prediction components that are computed sequentially to predict the outcome variable, y, minimizing the loss function in M consecutive steps.
Suppose that we are interested in fitting three different kinds of prediction components sequentially, a linear, an ordinal and a nominal one. Each prediction component is considered as the weighted sum of the selected transformations of the predictor variables.
So, when we first fit the linear prediction component, $\sum_{k=1}^{p} \beta_k^l \varphi_k^l(x_k)$, we can write the loss function as

$$L(\beta, X)_l = \|y - \sum_{k=1}^{p} \beta_k^l \varphi_k^l(\mathbf{x}_k)\|^2 \tag{3.4}$$

where the subscript l denotes a linear transformation. We thus define the residual from the linear component, r_{lin}, as

$$r_{lin} = y - \sum_{j=1}^{J} \beta_j^l \varphi_j^l(\mathbf{x}_j). \tag{3.5}$$

Next, we minimize the loss function with respect to this residual vector, r_{lin}, by fitting the second prediction component, which we choose to be ordinal, $\sum_{k=1}^{p} \beta_k^o \varphi_k^o(\mathbf{x}_k)$. More formally,

$$L(\beta, x)_{lin+ord} = \|r_{lin} - \sum_{k=1}^{p} \beta_k^o \varphi_k^o(\mathbf{x}_j)\|^2. \tag{3.6}$$

We consider the residual vector r_{ord}, which results in

$$r_{ord} = r_{lin} - \sum_{k=1}^{p} \beta_k^o \varphi_k^o(\mathbf{x}_j).$$

In the next step we consider the third prediction component, nominal, $\sum_{k=1}^{p} \beta_k^n \varphi_k^n(\mathbf{x}_k)$, and we can express the loss function as

$$L(\beta, X)_{lin+ord+nom} = \|r_{ord} - \sum_{k=1}^{p} \beta_k^n \varphi_k^n(x_k)\|^2 \tag{3.7}$$

If we consider the observed outcome, then this sequential procedure results to be equivalent in fitting a single prediction component (in this example a nominal one) if and only if the transformations considered in each prediction component are less restrictive in each step and if the optimal solution has been obtained in the last step with respect to both the weights and the transformations.

When our goal is to predict the outcome values for future observations iterating until the optimal solution is found is often not the best choice, because this optimal fit of the data can lead to poor predictions. For this reason we assume that obtaining a suboptimal solution through a boosting approach leads to an improvement in predictions. So, as described before, we follow this stagewise approach, but we do not iterate until convergence in each step that fits a prediction component. Thus, in each step of the procedure we stop before reaching convergence, and we move to the next step in which we consider the

residuals of the previous step and fit a new prediction component. In this way the number of total iterations strongly decreases.

3.3 Simulations

In this section we show some of the results we obtain with simulated datasets. The aim of our experiments is to prove that, in case of collinearity among predictors, which implies the presence of approximate concurvity, our strategy leads to an improvement in predictions.

In the first simulation the explicative variables have been simulated in such a way to show a moderate collinearity. We simulated a total of $n = 100$ observations.

	Eigenvalues	Tolerance
x_1	3.02435	0.42799
x_2	0.86006	0.33777
x_3	0.54180	0.34852
x_4	0.40262	0.44627
x_5	0.17118	0.79523

Table 3.1: Eigenvalues and tolerance for simulated dataset, 5 predictors

In Table 3.1 eigenvalues of the correlation matrix and values of tolerance for these five predictors are shown. As expected none of the explicative variables has a small tolerance value. Moreover, we simulate the outcome variable as:

$$y = 0.65 \times \sin X_1 - 1.4 \times \cos X_2 + 0.05 \times \sqrt{X_3} + \\ + 0.04 \times \sin \pi * X_4 - 0.6 \times \log X_5 + U[0, 1]$$

to impose a nonlinear relationship among the outcome variable and the explanatory variables.

	EPE	Std Error	APE	Std Error
lin(c)	1.987	0.052	1.545	0.020
ord(c)	1.225	0.034	0.918	0.023
nom(c)	1.206	0.036	0.834	0.015
lin(1)ord(1)nom(1)	1.587	0.054	1.173	0.041
lin(5)ord(5)nom(5)	1.241	0.046	0.863	0.014
ord(5)nom(5)	1.231	0.046	0.861	0.015
ord(1)nom(5)	1.228	0.053	0.862	0.015
ord(5)nom(1)	1.319	0.039	0.985	0.036
ord(1)nom(4)	1.232	0.055	0.865	0.015
ord(4)nom(1)	1.317	0.040	0.989	0.036
lin(5)ord(5)nom(c)	1.225	0.051	0.858	0.015

Table 3.2: EPE and APE and respective standard errors for different boosted models, simulation with five predictors

In Table 3.2 the Expected Prediction Error (EPE) for the test set and the Apparent Prediction Error (APE) for the training set, and their standard errors, obtained by evaluating the Mean Square Error estimates through a 10 fold cross validation, are shown. In the first column of Table 3.2, *lin* stands for linear transformation, *ord* for ordinal and *nom* for nominal. The number within parentheses indicates the number of iterations performed, while letter c indicates that the algorithm is performed until convergence.

We can note that the model that presents the lowest value of the EPE is the model in which there is a single nominal prediction component and iterations are carried on until convergence is reached. Applying the 1-SE rule we think that also the model $(ord(1)nom(4))$,

with two prediction components, respectively an ordinal and a nominal one, limiting the number of iterations to one for the first prediction component and four for the second one, performs quite good.

	Eigenvalues			Tolerance		
	before	ord(1)	nom(4)	before	ord(1)	nom(4)
x_1	3.02435	2.88019	1.75070	0.42799	0.60892	0.64019
x_2	0.86006	0.85247	1.06486	0.33777	0.42834	0.61942
x_3	0.54180	0.55294	1.00708	0.34852	0.42002	0.96363
x_4	0.40262	0.46068	0.78952	0.44627	0.53646	0.93456
x_5	0.17118	0.25371	0.38785	0.79523	0.82019	0.93987

Table 3.3: Eigenvalues and tolerance for simulated dataset, 5 predictors

In Table 3.3 we show the eigenvalues of the correlation matrix and the tolerance values of the predictors before and after each transformation. We notice that the value of the greatest eigenvalue after each step decreases, as well as the value of the smallest eigenvalues increases. Tolerance values were not extreme before transformations, but after transformations are applied they increase for all predictors. In other words, applying transformations, also through prediction components, linearizes the relationship among predictors.

In the second simulation, we simulated a total of $n = 250$ observations. The explanatory variables were simulated to be strongly related. Also the outcome variable, as before, was calculated in such a way to create a nonlinear relationship with the predictors,

$$y = \sin x_1 + \exp x_2 + x_3^3 \times 0.7 \times \cos x_4 + \exp -x_5 + \sin x_6 + \\ + 0.4 * x_7 \times \sin \pi x_8 + x_9^{4/3} + \exp x_{10} * x_{12}^2 + \cos 2x_{11} + \\ + x_{13}^3 + N(0, 0.01).$$

	Eigenvalues	Tolerance
x_1	2.56988	0.19800
x_2	2.20573	0.39154
x_3	1.74454	0.44654
x_4	1.55113	0.17037
x_5	1.23456	0.35718
x_6	0.92680	0.16947
x_7	0.77038	0.27817
x_8	0.63082	0.64637
x_9	0.53694	0.41468
x_{10}	0.42723	0.69113
x_{11}	0.20180	0.40436
x_{12}	0.15447	0.15163
x_{13}	0.04573	0.53992

Table 3.4: Eigenvalues and tolerance for simulated dataset, 13 predictors

We notice in Table 3.4 that some of these explanatory variables have low values of tolerance, especially x_6 and x_{12}. We apply again our strategy, which consists in considering different combinations of predictor components.

From Table 3.5 we see that the model that has the lowest value of EPE is the model in which a single ordinal component is used and iterations continue until convergence. Applying the 1-SE rule we choose

	EPE	Std Error	APE	Std Error
lin(c)	0.56041	0.59416	0.34893	0.02929
ord(c)	0.04217	0.02276	0.01603	0.00364
nom(c)	0.04813	0.02370	0.01185	0.00294
lin(1)ord(1)nom(1)	0.14676	0.08505	0.07710	0.01606
lin(5)ord(5)nom(5)	0.05854	0.04465	0.01594	0.00313
ord(5)nom(5)	0.04645	0.02753	0.01286	0.00321
ord(1)nom(5)	0.05169	0.03170	0.01510	0.00364
ord(5)nom(1)	0.04576	0.02717	0.01802	0.00424
ord(1)nom(4)	0.05491	0.02922	0.01683	0.00394
lin(5)ord(5)nom(c)	0.04811	0.02371	0.01185	0.00294

Table 3.5: EPE and APE and respective standard deviation for different boosted models, simulation with thirteen predictors

model $lin(5)ord(5)nom(5)$, in which we use sequentially three prediction components and we limit, in each step, the number of iterations to be equal to 5. As before we are also interested in evaluating the effect of transformations on the relationship that exists among predictors.

In Table 3.6 the eigenvalues of the correlation matrix of the transformed predictors and tolerance values are shown, before and after each prediction component is added in the model. As in the previous example, tolerance values, which before transformations were applied were quite small have become larger after transformations. Moreover, as before, applying nonlinear transformations causes a decrease in the greatest eigenvalue and an increase of the smallest one.

We performed several other simulations, varying the degree of correlation among predictors and obtained similar results. So, we can conclude that applying nonlinear transformation through optimal scaling and following the stagewise boosting approach we allow a suboptimal solution in terms of Apparent Prediction Error since this sub-

	Eigenvalues			Tolerance		
	lin(5)	ord(5)	nom(5)	lin(5)	ord(5)	nom(5)
1	2.56988	2.13813	2.20731	0.19800	0.45973	0.68303
2	2.20573	1.82896	1.87613	0.39154	0.75709	0.66788
3	1.74454	1.57988	1.59536	0.44654	0.65316	0.85693
4	1.55113	1.23352	1.34432	0.17037	0.83399	0.52448
5	1.23456	1.15684	1.13957	0.35718	0.71724	0.52127
6	0.92680	1.00128	1.07829	0.16947	0.40519	0.49769
7	0.77038	0.89638	0.90838	0.27817	0.58184	0.50014
8	0.63082	0.77422	0.69998	0.64637	0.82333	0.74650
9	0.53694	0.71800	0.59479	0.41468	0.77424	0.79848
10	0.42723	0.65483	0.53507	0.69113	0.85926	0.82001
11	0.20180	0.48634	0.39100	0.40436	0.84799	0.87686
12	0.15447	0.35378	0.29915	0.15163	0.45315	0.52522
13	0.04573	0.17784	0.20066	0.53992	0.77442	0.60992

Table 3.6: Eigenvalues and tolerance for simulated dataset, 13 predictors

optimality is compensated by a smaller Expected Prediciton Error and a drastic reduction of the number of iterations.

3.4 Real data analysis

The real dataset that we use in this section was collected by an Italian financial istitution, which allowed its use only for academic purposes under a non-disclosure agreement. These data contains 3568 observations measured on 22 variables and were sampled from a bigger dataset. All the variables are reported in Table 3.7. RBT is the outcome variable and all the other variables are predictors. These predictors are mostly binary, only four of them are quantitative. This feature makes a multiple linear regression approach not so easily interpretable for these data. Also in this section, according to the *stagewise approach* mentioned earlier, we apply a boosted additive model to these data, choosing different combinations of prediction components.

In Table 3.8 the eigenvalues of the correlation matrix of the independent variables and the values of the tolerance are shown. We can notice that this dataset is quite affected by collinearity, especially if we look at the values of the tolerance for predictors NMC, NFC, DD and RB. In Table 3.9 we report the Expected Prediction Error (EPE) and the Apparent Prediction Error (APE), and their standard errors, obtained by evaluating Mean Square Error estimates obtained with 10-fold cross validation.

In the first column of Table 3.9, *lin* stands for linear transformation, *ord* for ordinal and *nom* for nominal. No transformation are available for binary predictors, on the other hand, we use spline transformations (order two and two interior knots) for numerical ones. The number within parentheses indicates the number of iterations performed, while letter *c* indicates that the algorithm is performed until convergence. The model that presents the lowest EPE is the one in which we use

Label	Description
NMC	number of marketing campaigns the customer was involved
NC	number of contacts for maketing campaigns
NFC	total of financial products owned by the customer
AGE	customer age
type	customer type (family = 1, personal= 2)
LI	life insurance contracts and pension funds (NO = 0, YES = 1)
AM	asset management (NO = 0, YES = 1)
MF	mutual funds (NO = 0, YES = 1)
BOND	bonds (NO = 0, YES = 1)
DCER	deposit certificates (NO = 0, YES = 1)
IIT	innovative investment tools (NO = 0, YES = 1)
BAC	bank account (NO = 0, YES = 1)
DEP	deposits (NO = 0, YES = 1)
DD	direct debit (NO = 0, YES = 1)
DCARD	debit card (NO = 0, YES = 1)
CCARD	credit card (NO = 0, YES = 1)
RB	remote banking (NO = 0, YES = 1)
CMS	cash management services (NO = 0, YES = 1)
CSP	credited salary/pension (NO = 0, YES = 1)
SML	short-medium terms loans (NO = 0, YES = 1)
SEX	sex (male = 1, female = 2)
RBT	ratio between bank account assets and total assets

Table 3.7: Real dataset, data description

	Eigenvalues	Tolerance
type	4.28003	0.58901
NMC	2.27561	0.19961
NC	1.71962	0.57521
NFC	1.19563	0.04554
LI	1.05413	0.16440
AM	1.02155	0.64343
MF	0.99441	0.51688
BOND	0.97581	0.46291
DCER	0.91783	0.61994
IIT	0.87875	0.52290
BAC	0.80573	0.49839
DEP	0.72073	0.67561
DD	0.69244	0.19499
DCARD	0.63130	0.36871
CCARD	0.57363	0.34895
RB	0.53954	0.15789
CMS	0.51741	0.42515
CSP	0.46427	0.30518
SML	0.44926	0.52625
SEX	0.26036	0.97293
AGE	0.03196	0.51357

Table 3.8: Eigenvalues of the correlation matrix and tolerance values, real dataset

	EPE	Std.Error	APE	Std.Error
lin(c)	1.247	0.067	1.034	0.002
ord(c)	0.807	0.063	0.798	0.016
nom(c)	0.759	0.064	0.694	0.002
lin(1)ord(1)nom(1)	0.918	0.081	0.904	0.002
lin(5)ord(5)nom(5)	0.819	0.069	0.805	0.002
ord(5)nom(5)	0.822	0.068	0.809	0.003
ord(1)nom(5)	0.827	0.072	0.813	0.002
ord(5)nom(1)	0.868	0.076	0.855	0.007
ord(1)nom(4)	0.830	0.071	0.817	0.002
ord(4)nom(1)	0.868	0.076	0.855	0.007
nom(1)nom(4)	0.830	0.073	0.817	0.003
nom(4)nom(1)	0.868	0.082	0.854	0.002
lin(5)ord(5)nom(c)	0.759	0.064	0.694	0.002

Table 3.9: EPE and APE and respective standard deviation for different boosted models, real data

only a nominal prediction component and we iterate until convergence. If we follow the 1-SE rule the model $ord(5)nom(5)$ is chosen.

In Table 3.10 we report the eigenvalues of the correlation matrix and the values of tolerance for each explanatory variable before and after transformation. We notice that the greatest eigenvalue after transformations is smaller than the one before transformations. Similarly the smallest eigenvalue after transformation is greater than the one before transformation. Moreover, the values of the tolerance for all predictors are improved after applying transformation, especially if we look at explanatory variables NMC, LI, DD and RB. It is remarkable that the tolerance for the binary variables increases dramatically through the transformation of the numeric variables. As in previous examples even dealing with real data our optimal scaling approach seems to work very well: by transformation, we gain in reduction of expected prediction error compared to the linear solution and by using different prediction components we considerably reduce the computational time restricting the number of iterations in each step.

	Eigenvalues			Tolerance		
	before	ord(5)	nom(5)	before	ord(5)	nom(5)
type	4.28003	4.05841	3.59375	0.568901	0.71882	0.75685
NMC	2.27561	2.27530	2.05339	0.19961	0.45246	0.88101
NC	1.71962	1.68988	1.51919	0.57521	0.85648	0.93033
NFC	1.19563	1.22539	1.23375	0.04554	0.14546	0.28403
LI	1.05413	1.03825	1.07931	0.16440	0.63068	0.71889
AM	1.02155	1.00795	1.00314	0.64343	0.95913	0.96083
MF	0.99441	0.98443	0.99753	0.51688	0.64030	0.73346
BOND	0.97581	0.96892	0.95816	0.46291	0.54922	0.61852
DCER	0.91783	0.89574	0.95507	0.61994	0.99448	0.99573
IIT	0.87875	0.89130	0.89167	0.52290	0.83463	0.89846
BAC	0.80573	0.82076	0.85685	0.49839	0.64936	0.61051
DEP	0.72073	0.72975	0.84327	0.67561	0.76157	0.85092
DD	0.69244	0.70190	0.75288	0.19499	0.60088	0.60039
DCARD	0.63130	0.67971	0.70361	0.36871	0.67227	0.72441
CCARD	0.57363	0.63077	0.65662	0.34895	0.59075	0.72561
RB	0.53954	0.57081	0.62842	0.15789	0.37701	0.67148
CMS	0.51741	0.52056	0.57089	0.42515	0.64557	0.76638
CSP	0.46427	0.49549	0.52090	0.30518	0.61741	0.72159
SML	0.44926	0.45615	0.51609	0.52625	0.77485	0.91051
SEX	0.26036	0.26030	0.45887	0.97293	0.97217	0.97368
AGE	0.03196	0.09822	0.20664	0.51357	0.61883	0.91517

Table 3.10: Eigenvalues and tolerance values before and after each prediction component is added into the model, real data

Conclusions

The results from the simulated and the real data analysis suggest that through the combination of nonlinear regression with optimal scaling transformations and of the forward stagewise boosting approach, Generalized Boosted Additive Models can deal with data affected by collinearity and concurvity.

Moreover, when our goal is to predict the outcome values for future observations from a set of explanatory variables, the boosting approach, which is the building block of the sequential fitting procedure of these models, allows us to reach a good solution in terms of Expected Prediction Error, with the additional benefit of a considerable decrease in the number of iterations that are needed. In other words, if we want to iterate the procedure until (full) convergence, almost 100 iterations are tipically required. By contrast, the proposed models require less than 20 iterations, thus we obtain a remarkable saving in execution time.

The scientific results of this work can be extended in various directions: the hope is that they have induced curiosity in the reader.

Appendix A

R codes

A.1 Catreg

A.1.1 Algorithm for data normalization

```
NORM <- function(XX, norm) {
N <- dim(XX)[1]
M <- dim(XX)[2]
A <- colSums(as.matrix(XX))/N
E <- apply(XX, 2, function(x) sqrt(sum((x - mean(x))^2)))
YY <- apply(XX, 1, function(x) (x - A)/E)
YY <- YY * sqrt(norm)
return(t(YY))
}
```

A.1.2 Algorithm to compute knots from data

```
KNOTS <- function(y, #vector of data
norder, # spline order
nknots  # number of interior knots
) {
nobs <- length(y)
ncoef <- norder + nknots
temp <- y[order(y)]
m0 <- norder
```

```
m1 <- m0 + 1
m2 <- norder + nknots
m3 <- m2 + 1
m4 <- m3 + norder - 1
T <- rep(0,m4)
T[1:m0] <- temp[1]
T[m3:m4] <-temp[nobs]
xorder <- m3 - m0
prop <- (((m1- 1) + 1:(m2-(m1-1)))- m0)/xorder
xpos <- (nobs+1) * prop
npos <- floor(xpos)
xval <- ((temp[npos + 1] - temp[npos]) * (xpos - npos))+ temp[npos]
T[(m1-1) + 1:(m2 - (m1 - 1))] <- xval
return(T)
}
```

A.1.3 Algorithm to compute integrated m-splines

```
IMSPLN <- function(T,
 y,
 left,
 ival,
 datum,
 norder
 ){
j <- 1
isw <- 0

splm <- rep(1, length = 2)
spli <- rep(1, length = 2)
dr <- rep(1, length = 2)
dl <- rep(1, length = 2)
wk <- rep(1, length = 2)

K = 1
A = 1

while (A != 7) {
A = A

if(K == 1) {
if(norder == 1){
 A = 6
 } else {
 A = A
```

```
 }
}else{

#print(A)
if (A == 1) {
# step 1
jp1 <- j + 1
dr[j] <- T[left + j] - datum
dl[j] <- datum - T[(left - j) + 1]
s <- 0
i <- 1
A = 2
}
#print(A)
if (A == 2) {
# step 2
z <- wk[i]/(dr[i] + dl[jp1- i])
wk[i] <- s + (dr[i] * z)
s <- dl[jp1-i] * z

    if(j >= i + 1) {
          A = 2
          i <- i + 1
    }  else {
        wk[jp1] <- s

              if (isw == 1) {
                  A = 3
                  } else {
        j <- jp1
              if (j < norder) {
                  A = 1
                } else {
                  A = 6
                }
                }
      }}

#print(A)
if (A == 6) {
# HUP
xorder <- norder
i <- 1
A = 4
}
#print(A)
if(A == 4) {
# step 4:
```

```
splm[i] <- (wk[i]*xorder)/(T[left + i]- T[left + (i - norder)])
 if (norder >= i + 1) {
 A = 4
 i <- i + 1
 } else {
 isw <- 1
 A = 1
 }  }
#print(A)
if (A == 3) {
# step 3:
SUM <- 0
i <- 1
A = 5
}
#print(A)
if (A == 5) {
# step 5:
n <- norder + 1 - i
SUM <- SUM + wk[n + 1]
spli[n] <- SUM
 if(norder >= i +1) {
 A = 5
 i <- i + 1
 } else {
 A = 7
 }
 }
#print(A)
}
K <- K + 1
}
return(spli)
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

JSPLINE <- function(T,
y,
NORDER,
NKNOTS
){

ncoef <- NORDER + NKNOTS
nobs <- length(y)
ispline <- matrix(0, nobs, ncoef)

ival <- 1
```

```
KK <- 1
A = 11
while(A !=71) {

if(A == 11) {
# step 1:
mj <- 0
datum <- y[ival]
left <- NORDER
if(ncoef -1 < left) {
A = 31
} else {
A = 21
}}
#print(A)
if(A == 21) {
# step 2
  if (T[left+1] > datum) {
   A = 31
   } else {
   if((ncoef - 1) >= left + 1) {
      A = 21
      left <- left + 1
  } else {
  A = 31
  left <- left + 1
  }}
  }
#print(A)
if(A == 31) {
# step 3:
spli <- IMSPLN(T, datum, left, ival, datum, norder = NORDER)
k <- 1
 if((left - NORDER) < k) {
    A = 61
    } else {
    A = 41
    }
    }
#print(A)
if (A == 41) {
# step 4
  ispline[ival, mj +k] <- ispline[ival, mj + k] + 1
   if ((left - NORDER) >= k + 1) {
        A = 41
        k = k+1
        } else {
```

```
        A = 61
        }
        }
#print(A)
if (A == 61) {
  m <- 0
  k <- (left - NORDER) + 1
  A = 51 }
#print(A)
if(A == 51) {
m <- m + 1
ispline[ival, mj + k] <- ispline[ival, mj +k] + spli[m]
    if (left >= k+1) {
    A = 51
    k = k+1
    } else {
        if (nobs >= ival + 1) {
        A = 11
        ival = ival + 1
        } else{
        A = 71}
        }}
#print(A)
KK <- KK +1
}
DEVFMN(ispline)
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
DEVFMN <- function(x # ISPLINE matrix
) {
a <- matrix(colSums(x), ncol = ncol(x), nrow = nrow(x), byrow =TRUE)/nrow(x)
w <- x - a
return(w)
}
```

A.1.4 Backfitting – inner loop

```
inner <- function(u,      # unrestricted quantification
                  inb,    # splines coefficients
                  S,      # I-splines basis
                  ssbase, # sum of squares I-splines basis
                  ncoef,  #
                  type,   # 4 = nominal spline 5 = ordinal spline
                  j,      # variable number
                  n.inits # number it for splines
```

```
                 ) {
u <- as.matrix(u)
S <- as.matrix(S)
iter <- 1
maxinit <- 2
ready <- FALSE
eps <- 1e-6

inner.res <- sum((u - (S %*% inb))^2)
bb <- cbind(inb, NULL)
R <- cbind(inner.res,NULL)

###############################################################################
#
###############################################################################
while(!ready) {
   for(T in 1:ncoef) {
      Z <- as.matrix(S[, -T]) %*% as.matrix(inb[-T])
      z <- u - Z
inb[T] <- S[,T] %*% z
      if (type == 3 | type == 5) {
        if (inb[T] >= 0)
           { inb[T] <- inb[T] }
        else
          {inb[T] <- 0 }
      }
      if (ssbase[T] > 0) { inb[T] <- inb[T] / ssbase[T] }
   }
   res.iter <-  sum((u - (S %*% inb))^2)
   R <- cbind(R, res.iter)
   bb <- cbind(bb, inb)
#   ready <- abs((R[iter+1] - R[iter]) / R[iter+1]) <=  eps
   if(iter == 1) {
      ready <- (R[iter+1] <= R[iter]) & (iter >= 5) }
   else {
      ready <- (R[iter+1] <= R[iter]) & (iter >= maxinit)
   }
#   ready <- iter == 1
   iter <- iter + 1
#   print(iter)
}

W <- list ("inb" = inb)
return(W)
}
```

A.1.5 Catreg function

```
catreg <- function(y,  # dependent variable
                   x,  # independent variables
                   NORDER, # spline order
                   NKNOTS, # number of interior knots
                   TYPE,   # transformation type
                   n.inits,  # number of iteration in the inner loop
                   critit,   # convergence criterion outer
                   maxit     # maximum number it outer
                   ) {

source("NORM.r")
source("KNOTS.r")
source("DEVFMN.r")
source("IMSPLN.r")
source("JSPLINE.r")
source("INNER.r")

nn <- dim(x)[1]

ssdat <- colSums(x^2)
varord <- order(ssdat, decreasing = FALSE)
#varord <- c(1:nvar)

norm <<- 1
Q <- t(NORM(y ,norm))    # normalized response variable
PHI <- NORM(x, norm)     # normalized indep variables

betas <- as.matrix(rep(1/nvar, nvar))       # initialize beta

bbetas <- cbind(betas, NULL)                # store beta in each iteration
RES.ini <- sum((Q - (PHI %*% betas))^2)     # Initial RSS
RES.T <- cbind(RES.ini, NULL)               # Store RSS in each iteration

for(j in 1:nvar) {
   if (TYPE[j] == 1) {
      NKNOTS[j] <- 0
      NORDER[j] <- 0
   }
   if (TYPE[j] == 2 | TYPE[j] == 3){
      NKNOTS[j] <- length(unique(x[,j])) - 2
      NORDER[j] <- 1
   }
   if (TYPE[j] >= 4) {
###   check number of specified knots
      maxnknots <- length(unique(x[,j])) - 2
      if (NKNOTS[j] > maxnknots) {
```

```
         NKNOTS[j] <- maxnknots
###      issue a warning
         cat("The number for knots of variable",j,"is set to the maximum of",
         maxnknots,"\n")
      }
   }
}

sumncoef <- sum(NORDER) + sum(NKNOTS)
ISPLINE <- array(0,dim=c(nn, sumncoef))
ssbase <- array(0,dim=sumncoef)
splb <- array(0,dim=sumncoef)
ncoef <- rep(0, nvar)
ntotknots <<- rep(0, nvar)
for(j in 1:nvar) {
   ncoef[j] <- NKNOTS[j] + NORDER[j]
   ntotknots[j] <<- NKNOTS[j] + 2*NORDER[j]
}
maxknots <- max(NKNOTS) + 2*max(NORDER)
knot <<- array(0,dim=c(maxknots, nvar))

jw <- 1
jb <- ncoef[1]
for(j in 1:nvar) {
   if (TYPE[j] != 1) {
      splb[jw:jb] <- 1/ncoef[j]               # initialize spline coefficients
      nknots <- NKNOTS[j]
      norder <- NORDER[j]
      if (TYPE[j] == 2 | TYPE[j] == 3) {
###      Knots for nom/ord level ###
         cat <- unique(x[,j])
         ord <- order(cat)
         knot[1:ntotknots[j],j] <<- unique(cat[ord])
      }
      if (TYPE[j] >= 4) {
         knot[1:ntotknots[j],j] <<- KNOTS(unique(x[,j]), norder, nknots)
      }
      ISPLINE[,jw:jb]<- JSPLINE(knot[1:ntotknots[j],j], x[,j], norder, nknots)
for (k in jw:jb) { ssbase[k] <- sum(ISPLINE[,k]^2) }
   }
   jw = jb + 1
   if (j < nvar) { jb = jb + ncoef[j+1] }
}
### Set pointers to bases and spline weights ###
pntrbase <- rep(0, nvar)
k <- 1
for (j in 1:nvar) {
   pntrbase[j] <- k
```

```
   k <- k + ncoef[j]
}
################################################
it <- 1
ready2 <- FALSE

while(!ready2){
for(jj in 1:nvar) {
   j <- varord[jj]
   jw <- pntrbase[j]
   jb <- jw + ncoef[j] - 1
   H <- as.matrix(PHI[,-j]) %*% betas[-j]
   u <- Q - H
   if(TYPE[j] == 1)
      { betas[j] <- t(PHI[,j]) %*% u }
   else {
      result <- inner(u, splb[jw:jb], ISPLINE[,jw:jb], ssbase[jw:jb],
      ncoef[j], TYPE[j], j, n.inits)

      if (TYPE[j] == 3 | TYPE[j] == 5) {
         if (all (result$inb == 0)) {
            reflu <- -1 * u
            result <- inner(reflu, splb[jw:jb], ISPLINE[,jw:jb],
            ssbase[jw:jb], ncoef[j], TYPE[j], j, n.inits)
         }
      }
      splb[jw:jb] <- result$inb
      PHI[,j] <- as.matrix(ISPLINE[,jw:jb]) %*% as.matrix(splb[jw:jb])
      ssq <- PHI[,j] %*% PHI[,j]
      if (ssq > 0) { PHI[,j] <- (PHI[,j] / sqrt(ssq)) * sqrt(norm) }
      betas[j] <- (t(u) %*% PHI[,j]) / norm
   }
}
bbetas <- cbind(bbetas,betas)
RES.it <- sum((Q - (PHI %*% betas))^2)

RES.T <- cbind(RES.T, RES.it)
ready2 <- (abs(RES.T[it+1] - RES.T[it]) <= critit) | (it == maxit)
it <- it + 1
}

### Relative Importance measure ###
imp <- as.matrix(rep(0, nvar))
zcor <- array(0,dim=nvar)
for (j in 1:nvar) {
   zcor[j] <- (t(Q)  %*% PHI[,j]) / norm
   imp[j] <- zcor[j] * betas[j]
}
```

```
tmp <- sum(imp)
imp <- imp / tmp

### Tolerance ##########
R_p <- cor(PHI)
IR_p <- diag(solve(R_p))
tol <- rep(0, nvar)
for(i in 1:nvar) {
  if(abs(betas[i]) > 0){
      tol[i] <- 1/IR_p[i]
      } else {
      tol[i] <- 0
      }
      }

W = list(betas = betas,
         importance = imp,
         transf = PHI,
         Q = Q,
         hist.RSS = RES.T,
         hist.beta = bbetas,
         RSS = RES.T[length(RES.T)])
return(W)
}
```

A.1.6 Algorithm for plotting transformations

```
OutputTrans <- function(x,      # original predictor values
                        transf, # transformed variables
                        VNAME,  # variables names
                        TYPE,   # transformation type
                        prints, # TRUE or FALSE
                        plots   # TRUE or FALSE
                       ) {

npred <- dim(x)[2]
nn <- dim(x)[1]
nvar <- length(VNAME)
dim1 <- 4
dim2 <- ceiling(nvar / 4)
if (plots) {
windows()
par(mfrow=c(dim1, dim2))
```

```
for(j in 1:npred){
   ord <- order(x[,j])
   if (TYPE[j] == 1) {
      plot(x[ord,j], transf[ord,j], type = 'b', xlab = VNAME[j],
      ylab = 'phi', main = 'NUM TRANSF')
    }
   if (TYPE[j] == 2) {
      plot(x[ord,j], transf[ord,j], type = 'b', xlab = VNAME[j],
      ylab = 'phi', main = 'NOM TRANSF')
    }
   if (TYPE[j] == 3) {
      plot(x[ord,j], transf[ord,j], type = 'b', xlab = VNAME[j],
      ylab = 'phi', main = 'ORD TRANSF')
    }
   if (TYPE[j] == 4) {
      plot(x[ord,j], transf[ord,j], type = 'b', xlab = VNAME[j],
      ylab = 'phi', main = 'SPLINE NOM TRANSF')
    }
   if (TYPE[j] == 5) {
      plot(x[ord,j], transf[ord,j], type = 'b', xlab = VNAME[j],
      ylab = 'phi', main = 'SPLINE ORD TRANSF')
 }
}
}

if (prints) {
W <- list(rep(0))
for(j in 1:npred){
   ord <- order(x[,j])
   W[[j]] <- unique(transf[ord,j]) * sqrt(nn)
   print(W)
}
}

}
```

A.2 Boosted Additive Models

A.2.1 Algorithms for implementing boosted additive models when cross validation is required

```
CatFreq <- function(y, x){

nobs <- dim(x)[1]

ncat <- array(0,dim=nvar)
for (j in 1:nvar) {
   ncat[j] <- length(unique(x[,j]))
}
ncatdep <- length(unique(y))

maxncat <- max(ncat)
cat <- array(0,dim=c(maxncat,nvar))
for (j in 1:nvar) {
   ord <- order(x[,j])
   cat[1:ncat[j],j] <- unique(x[ord,j])
}
ord <- order(y)
catdep <- unique(y[ord])


freq <- array(0,dim=c(maxncat,nvar))
for (j in 1:nvar) {
   for (k in 1:ncat[j]) {
      ix <- which((x[,j] == cat[k,j]) == TRUE)
      freq[k,j] <- length(ix)
   }
}
freqdep <- array(0,dim=ncatdep)
for (k in 1:ncatdep) {
   ix <- which((y == catdep[k]) == TRUE)
   freqdep[k] <- length(ix)
}

return(list(nobs=nobs, ncat=ncat, cat=cat, freq=freq,
        maxncat=maxncat, ncatdep=ncatdep, catdep=catdep,
        freqdep=freqdep))
}
################################################################

expandFrame <- function (tab, clean = TRUE, zero = TRUE,
```

```
                        returnFrame = TRUE){
    n <- dim(tab)[1]
    m <- dim(tab)[2]
    g <- matrix(0, n, 0)
    l <- rep("", 0)
    lab1 <- labels(tab)[[1]]
    lab2 <- labels(tab)[[2]]
    for (j in 1:m) {
        y <- as.factor(tab[, j])
        h <- levels(y)
        g <- cbind(g, ifelse(outer(y, h, "=="), 1, 0))
        l <- c(l, paste(lab2[j], "_", h, sep = ""))
    }
    if (zero)
        g <- ifelse(is.na(g), 0, as.matrix(g))
    if (clean) {
        g <- g[which(rowSums(g) > 0), which(colSums(g) > 0)]
        g <- g[, which(colSums(g) < n)]
    }
    if (!returnFrame)
        return(g)
    g <- as.data.frame(g, row.names = lab1)
    names(g) <- l
    return(g)
}
#################################################################

CreateIndmat <- function(y,  # dependent variable
                         x,  # independent variables
                         ncat,
                         ncatdep,
                         nobs #number of observations
                         ){
source("expandFrame.r")

### Set pointers to Indmat ###
pntrixIndmat <<- rep(0, nvar)
k <- 1
for (j in 1:nvar) {
   pntrixIndmat[j] <<- k
   k <- k + ncat[j]
}
sumncat <- sum(ncat)
Indmat <<- array(0,dim=c(nobs,sumncat))
for (j in 1:nvar) {
   var <- as.matrix(x[,j])
   a <- pntrixIndmat[j]
   b <- a + ncat[j] - 1
```

```
   Indmat[,a:b] <<- as.matrix(expandFrame(var,
                    zero = FALSE, clean = FALSE))
}

Indmatdep <<- array(0,dim=c(nobs,ncatdep))
var <- as.matrix(y)
Indmatdep <<- as.matrix(expandFrame(var,
                         zero = FALSE, clean = FALSE))
}

DepInclTest <- function(y, QlinTrain, nobsAll, ncatdep,
                        ncatdepAll, freqdep, catdep,
                        catdepAll,NORDER, NKNOTS, knot,
                        TYPE, nv){

QlinAll <- array(0,dim=nobsAll)
maxncat <- max(ncatdepAll)
quantlin <- array(0,dim=maxncat)
tmp <- array(0,dim=ncatdep)
for (k in 1:ncatdep) {
   ix <- which((y == catdep[k]) == TRUE)
   tmp[k] <- QlinTrain[ix[1]]
}
kk <- 1
for (k in 1:ncatdepAll) {
   if (freqdep[k] != 0) {
      quantlin[k] <- tmp[kk]
      kk <- kk + 1
   } else {
      quantlin[k] <- 0
   }
}

### Check if interpolation required ###

if (ncatdepAll != ncatdep) {
   for (k in 1:ncatdepAll) {
      if (freqdep[k] == 0) {

        quantlin[k] <- interpol(1, catdepAll[k], y, quantlin,
                          catdepAll, ncatdepAll, freqdep, 0, 0, 0, 1)
      }
   }
}
QlinAll <- Indmatdep %*% as.matrix(quantlin[1:ncatdepAll])
return(QlinAll)
}
#########################################################################
```

```
TransInclTest <- function(x, transTrain, nobsAll, ncat,
                          ncatAll, freq, cat, catAll,
                          NORDER, NKNOTS, knot, TYPE, nv){

transAll <- array(0,dim=c(nobsAll, nv))
maxncat <- max(ncatAll)
quant <- array(0,dim=c(maxncat,nv))
for (j in 1:nv) {
   tmp <- array(0,dim=ncat[j])
   for (k in 1:ncat[j]) {
      ix <- which((x[,j] == cat[k,j]) == TRUE)
      tmp[k] <- transTrain[ix[1],j]
   }
   kk <- 1
   for (k in 1:ncatAll[j]) {
      if (freq[k,j] != 0) {
         quant[k,j] <- tmp[kk]
         kk <- kk + 1
      } else {
         quant[k,j] <- 0
      }
   }
}

### Check if interpolation required ###
for (j in 1:nv) {
   if (ncatAll[j] != ncat[j]) {
      #for (k in 1:ncatAll[j]) {
         ind <-  which(quant[,j] == 0)

           for(z in ind){
           quant[z,j] <- interpol(j, catAll[z,j], x[,j],
                                  quant[1:ncatAll[j],j],
                                  catAll, ncatAll, freq,
                                  NORDER[j], NKNOTS[j],
                                  knot[1:ntotknots[j],j],
                                  TYPE[j])
            }
      }

   a <- pntrixIndmat[j]
   b <- a + ncatAll[j] - 1
   transAll[,j] <- Indmat[,a:b] %*% as.matrix(quant[1:ncatAll[j],j])
   ix <- which((transAll[,j] == 99999) == TRUE)
   transAll[ix,j] <- NA
}

return(transAll)
```

```
}
#########################################################################

interpol <- function(j, catip, dattrain, quant, catAll, ncatAll,
                     freq, NORDER, NKNOTS, knot,LEVEL) {



if (LEVEL == 1) {
   mean <- sum(dattrain) / nobstrain
   ss <- sum((dattrain - mean)^2)
   ipval1 <- (catip - mean) / sqrt(ss)
} else {
   interpol <- TRUE
   if (LEVEL == 2 | LEVEL == 3) {
#      goto 5000
      ipval1 <- 99999
      interpol <- FALSE
   }
   if (interpol) {
### CHECK IF EXTRAPOLATION IS REQUIRED
      catneih <- FALSE
      catneil <- FALSE
      for (k in 1:ncatAll[j]) {
         if (freq[k,j] != 0) {
            if (catAll[k,j] > catip) { catneih <- TRUE }
            if (catAll[k,j] < catip) { catneil <- TRUE }
         }
      }
      extrapol <- FALSE
      if ((!catneih) | (!catneil)) {
         extrapol <- TRUE
         interpol <- FALSE
         ipval1 <- 99999
      }
   }

   if (interpol) {

   ipval1 <- 0
   dimtmp1 <- (NORDER + 1)
   tmpa1 <- array(0,dim=c(dimtmp1, dimtmp1))
   tmpaa1 <- array(0,dim=dimtmp1)
   tmpb1 <- array(0,dim=dimtmp1)
   tmp21 <- array(0,dim=c(dimtmp1, dimtmp1))
   tmp31 <- array(0,dim=dimtmp1)
      tmpa1[1:dimtmp1,1] <- 1
      tmpaa1[1] <- 1
```

```
tknot <- trunc(knot)
indk <- which(tknot > catip)[1]
categ <- tknot[indk]

   for(tt in 1:dimtmp1) {
      for (t in 1:(dimtmp1-1)) {
               catix1 <- 0
               ready <- FALSE
               while(!ready) {
               id <- which(catAll[,j] == categ)
                if(length(id) != 0 && freq[id, j] != 0){
                catix1 <- id
                ready = TRUE
                } else {
                catix1 <- 0
                categ <- categ - 1
                }
                }
          if (is.na(ipval1) == TRUE) { break }
         tmpa1[tt,t+1] <- catAll[catix1,j]^t
         tmpaa1[t+1] <- catip^t
      } # end t loop

      if (is.na(ipval1) == TRUE) { break }
      tmpb1[tt] <- quant[catix1]
      if (tt < dimtmp1) {
         categ <- categ - 1
         if (categ == 0) {
            ipval1 <- 99999
         }
      }
   } # end  tt loop

    if (is.na(ipval1) != TRUE) {
      tmp21 <- solve(tmpa1)
      for (i in 1:dimtmp1) {
         tmp31[i] <- 0
         for (l in 1:dimtmp1) {
            tmp31[i] <- tmp31[i] + tmp21[i,l] * tmpb1[l]
         }
      }

      ipval1 <- 0
      for (k in 1:dimtmp1) {
         ipval1 <- ipval1 + tmpaa1[k] * tmp31[k]
      }
   }
```

```
  }
} ### end else
return(ipval1)
}
#####################################################################
```

A.2.2 Algorithm for boosted additive model

```
boosting <- function(y,  # outcome variable
                     x,  # predictor variables
                     NORDER, # spline order
                     NKNOTS, # number of interior knots
                     TYPE,   # type of transformation
                     maxit){ # max number of iterations

source("catreg.r")
y <- as.matrix(y)
x <- as.matrix(x)
# pointer of the number of additive components
if(dim(TYPE)[2] == 1){
A = 100
}
if (dim(TYPE)[2] == 2) {
   A = 200
 }
if(dim(TYPE)[2] == 3){
   A = 300
}

nobs <- length(y)

norm=1
TYPE1 <- TYPE[,1]
mod.1<- catreg(y, x, NORDER, NKNOTS, TYPE[,1], n.inits, critit, maxit[1])
betas.M1 <- mod.1$betas
Q.M1 <- mod.1$Q
trans.M1 <-mod.1$transf

pred.M1 <- trans.M1 %*% betas.M1
res.M1 <- as.matrix(Q.M1 - pred.M1)

if(A == 100){
pred.M1 <- (pred.M1)*sqrt(nobs)
dep.M1 <- Q.M1 * sqrt(nobs)
res <- (dep.M1 - pred.M1)
betas.M2 = NA
```

```
betas.M3 = NA
PHI <- trans.M1

} else {

norm=1
TYPE2 <- TYPE[,2]
if (length(maxit) < 2) {
stop("you have to specify a number of max iteration per type")
}

mod.2 <- catreg(res.M1, x, NORDER, NKNOTS, TYPE2, n.inits, critit, maxit[2])
betas.M2 <- mod.2$betas
Q.M2 <- mod.2$Q
trans.M2 <- mod.2$trans

MSSres.M1 <- mean((res.M1*sqrt(nobs))^2, na.rm = TRUE)
predraw.M2 <- trans.M2 %*% (betas.M2 * sqrt(MSSres.M1))
res.M2 <- res.M1 - predraw.M2

if(A == 200){
pred.M2 <- (pred.M1 + predraw.M2)*sqrt(nobs)
dep.M2 <- Q.M1 * sqrt(nobs)
res <- (dep.M2 - pred.M2)
betas.M3 <- NA
PHI <- trans.M2
} else {

TYPE3 <- TYPE[,3]
if (length(maxit) < 3) {
stop("you have to specify a number of max iteration per type")
}

mod.3 <- catreg(res.M2, x, NORDER, NKNOTS, TYPE3, n.inits, critit, maxit[3])
betas.M3 <- mod.3$betas
Q.M3 <- mod.3$Q
trans.M3 <- mod.3$transf

MSSres.M2 <- mean((res.M2*sqrt(nobs))^2, na.rm = TRUE)
predraw.M3 <- trans.M3 %*% (betas.M3 * sqrt(MSSres.M2))

pred.M3 <- (pred.M1 + predraw.M2 + predraw.M3)*sqrt(nobs)
dep.M3 <- QAll.M1 * sqrt(nobs)
res <- (dep.M3 - pred.M3)
PHI <- trans.M3
  }
}
```

```
MSE <- mean(res^2, na.rm = TRUE)

if(all(!is.na(betas.M1))){
      b1 <- betas.M1
      } else {
      b1 <- NA
      }

if(all(!is.na(betas.M2))){
      b2 <- betas.M2
      } else {
      b2 <- rep(NA, length(b1))
      }

if(all(!is.na(betas.M3))){
b3 <- betas.M3
} else {
b3 <- rep(NA, length(b1))
}

eigenvals <- svd(cor(PHI))$d
tolerance <- 1/diag(solve(cor(PHI)))


W <- list(MSE=MSE, beta1 = b1, beta2 = b2, beta3 = b3, eig = eigenvals, tol = tolerance)
return(W)
}
```

Bibliography

[1] R.E. Barlow, D.J. Bartholomew, J.M. Bremner, and H.D. Brunk. *Statistical inference under order restrictions: the theory and application of isotonic regression*. J. Wiley, 1972.

[2] R. E. Bellman. *Adaptive control processes - A guided tour*. Princeton University Press, Princeton, New Jersey, U.S.A., 1961.

[3] R.D. Bock. Methods and applications of optimal scaling (tech. rep. 25). Technical report, Chapel Hill, NC: University of North Carolina, L.L. Thurstone Psychometric Laboratory, 1960.

[4] A.W. Bowman and A. Azzalini. *Applied smoothing techniques for data analysis: the kernel approach with S-Plus illustrations*, volume 18. Oxford University Press, USA, 1997.

[5] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[6] L. Breiman and J.H. Friedman. Estimating optimal transformations for multiple regression and correlation. *Journal of the American Statistical Association*, 80:580–598, 1985.

[7] A. Buja. Remarks on functional canonical variates, alternating least squares methods and ace. *The Annals of Statistics*, 18(3):1032–1069, 1990.

[8] A. Buja, T. Hastie, and R. Tibshirani. Linear smoothers and additive models. *The Annals of Statistics*, 17:453–510, 1989.

[9] J.M. Chambers, T. Hastie, et al. *Statistical models in S*. Chapman & Hall London, 1992.

[10] Z. Chen, C. Gu, and G. Wahba. Linear smoothers and additive models: Discussion. *The Annals of Statistics*, 17(2):515–522, 1989.

[11] B. Clarke, E. Fokoue, and H.H. Zhang. *Principles and Theory for Data Mining and Machine Learning (Springer Series in Statistics)*. Springer, 1 edition, July 2009.

[12] W.S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American statistical association*, 74:829–836, 1979.

[13] W.S. Cleveland and S.J. Devlin. Locally weighted regression: an approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83:596–610, 1988.

[14] C. Conversano, R. Siciliano, and F. Mola. Supervised classifier combination through generalized additive multi-model. In Josef Kittler and Fabio Roli, editors, *Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science*, pages 167–176. Springer, 2000.

[15] C. Conversano, R. Siciliano, and F. Mola. Generalized additive multi-mixture model for data mining. *Comput. Stat. Data Anal.*, 38:487–500, 2002.

[16] H.B. Curry and I.J. Schoenberg. On pólya frequency functions iv: the fundamental spline functions and their limits. *Journal d'analyse mathématique*, 17(1):71–107, 1966.

[17] C. de Boor. *A practical guide to splines*. Springer Verlag., 1978.

[18] J. De Leeuw and W.J. Heiser. Multidimensional scaling with restrictions on the configuration. *Multivariate analysis*, 5:501–522, 1980.

[19] J. De Leeuw, F.W. Young, and Y. Takane. Additive structure in qualitative data: An alternating least squares method with optimal scaling features. *Psychometrika*, 41(4):471–503, 1976.

[20] P. Dierckx. *Curve and surface fitting with splines*. Oxford University Press, USA, 1995.

[21] B. Efron. Bootstrap methods: another look at the jackknife. *The annals of Statistics*, 7(1):1–26, 1979.

[22] B. Efron. Estimating the error rate of a prediction rule: improvement on cross-validation. *Journal of the American Statistical Association*, 78:316–331, 1983.

[23] B. Efron and R. Tibshirani. *An introduction to the bootstrap*, volume 57. Chapman & Hall/CRC, 1993.

[24] J. Fan and I. Gijbels. *Local polynomial modelling and its applications*, volume 66. Chapman & Hall/CRC, 1996.

[25] J. Fox. *Multiple and generalized nonparametric regression*. Sage Publications Thousand Oaks, CA, 2000.

[26] J. H. Friedman. Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 38:367–378, 1999.

[27] J.H. Friedman. A variable span smoother. Technical report, Laboratory for Computational Statistics, Standford University, 1984.

[28] J.H. Friedman. Multivariate adaptive regression splines (with discussion). *The annals of statistics*, 19:1–141, 1991.

[29] J.H. Friedman. On bias, variance, 0/1 loss, and the curse-of-dimensionality. *Data mining and knowledge discovery*, 1(1):55–77, 1997.

[30] J.H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

[31] J.H. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[32] J.H. Friedman. Recent advances in predictive (machine) learning. *Journal of classification*, 23(2):175–197, 2006.

[33] J.H. Friedman and P. Hall. On bagging and nonlinear estimation. *Journal of statistical planning and inference*, 137(3):669–683, 2007.

[34] J.H. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting (with discussion). *The annals of statistics*, 28(2):337–407, 2000.

[35] J.H. Friedman and B.E. Popescu. Gradient directed regularization for linear regression and classification. Technical report, Laboratory for Computational Statistics, Standford University, 2003.

[36] J.H. Friedman and B.E. Popescu. Importance sampled learning ensembles, 2003.

[37] J.H. Friedman and B.W. Silverman. Flexible parsimonious smoothing and additive modeling (with discussion). *Technometrics*, 31:3–39, 1989.

[38] J.H. Friedman and W. Stuetzle. Projection pursuit regression. *Journal of the American statistical Association*, 76:817–823, 1981.

[39] A. Gifi. *Nonlinear multivariate analysis*. New York: John Wiley & Sons, 1990.

[40] G.H. Golub, M. Heath, and G. Wahba. Generalized Cross-Validation as a Method for Choosing a Good Ridge Parameter. *Technometrics*, 21(2):215–223, 1979.

[41] P. J. Green. Iteratively reweighted least squares for maximum likelihood estimation, and some robust and resistant alternatives (with discussion). *Journal of the Royal Statistical Society, Series B, Methodological*, 46:149–192, 1984.

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

[43] P.J. Green and B.W. Silverman. *Nonparametric regression and generalized linear models: a roughness penalty approach*, volume 58. Chapman & Hall/CRC, 1994.

[44] P.J.F. Groenen, W.J. Heiser, and J.J. Meulman. Global optimization in least squares multidimensional scaling by distance. *Journal of Classification*, 16:225–254, 1997.

[45] P.J.F. Groenen, B.J. van Os, and J.J. Meulman. Optimal scaling by alternating length-constrained nonnegative least squares, with application to distance-based analysis. *Psychometrika*, 65(4):511–524, 2000.

[46] Aydinli G. Härdle, W. and S. Sperlich. *The Art of Semiparametrics*. Physica-Verlag, Germany, 2006.

[47] W. Härdle. *Applied Nonparametric Regression (Econometric Society Monographs)*. Cambridge University Press, January 1992.

[48] W. Härdle and P. Hall. On the backfitting algorithm for additive regression models. *Statistica neerlandica*, 47(1):43–57, 1993.

[49] W. Härdle, M. Müller, S. Sperlich, and A. Werwatz. *Nonparametric and Semiparametric Models*. Springer Verlag, Heidelberg, 2004.

[50] T. Hastie and R. Tibshirani. Generalized additive models (with discussion). *Statistical Science*, 1:297–318, 1986.

[51] T. Hastie and R. Tibshirani. Generalized additive models, cubic splines and penalized likelihood. Technical report, Stanford University, CA, Department of Statisctics, 1987.

[52] T. Hastie and R. Tibshirani. Generalized additive models: some applications. *Journal of the American Statistical Association*, 82:371–386, 1987.

[53] T. Hastie, R. Tibshirani, and J.H. Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition, 2009.

[54] Tibshirani R. Hastie, T and J.H. Friedman. *The elements of statistical learning*. Springer-Verlag, 2001.

[55] T.J. Hastie and R.J. Tibshirani. *Generalized additive models*. Chapman & Hall/CRC, 1990.

[56] J.B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

[57] J.B. Kruskal. Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, 29(2):115–129, 1964.

[58] J.B. Kruskal. Analysis of factorial experiments by estimating monotone transformations of the data. *Journal of the Royal Statistical Society. Series B (Methodological)*, 27:251–263, 1965.

[59] P. McCullagh and J.A. Nelder. *Generalized linear models*. Chapman & Hall/CRC, 1989.

[60] J.J. Meulman. The integration of multidimensional scaling and multivariate analysis with optimal transformations. *Psychometrika*, 57(4):539–565, 1992.

[61] J.J. Meulman. Principal coordinates analysis with optimal transformation of the variables: Minimizing the sum of squares of the smallest eigenvalues. *British Journal of Mathematical and Statistical Psychology*, 46(2):287–300, 1993.

[62] J.J. Meulman. Optimal scaling methods for graphical display of multivariate data. *COMPSTAT 1998 Proceedings in Computational Statistics*, pages 65–76, 1998.

[63] J.J. Meulman. Discriminant analysis with optimal scaling in r. *Classification and information processing at the turn of the millenium*, pages 32–39, 2000.

[64] J.J. Meulman. Prediction and classification in nonlinear data analysis: Something old, something new, something borrowed, something blue. *Psychometrika*, 68(4):493–517, 2003.

[65] J.J. Meulman, W.J. Heiser, and SPSS Inc. *SPSS Categories 10.0*. SPSS Inc., 2004.

[66] J.J. Meulman and A.J. van der Kooij. Transformations towards independence through optimal scaling. In *International Conference on Measurement and Multivariate Analysis (ICMMA), Banff, Canada*, 2000.

[67] J.J. Meulman, A.J. Van der Kooij, and W.J. Heiser. Principal components analysis with nonlinear optimal scaling transformations for ordinal and nominal data. *Handbook of quantitative methodology for the social sciences*, pages 49–70, 2004.

[68] D. Pregibon and Y. Vardi. Estimating optimal transformations for multiple regression and correlation: Comment. *Journal of the American Statistical Association*, 80(391):598–601, 1985.

[69] J.O. Ramsay. Monotone regression splines in action. *Statistical Science*, 4:425–441, 1988.

[70] J. Rice and M. Rosenblatt. Smoothing splines, regression derivatives and convolution. *Annals of Statistics*, 11:141–156, 1983.

[71] R.E. Schapire. A brief introduction to boosting. In *International Joint Conference on Artificial Intelligence*, volume 16, pages 1401–1406. LAWRENCE ERLBAUM ASSOCIATES LTD, 1999.

[72] R.E. Schapire and Y. Singer. Improved boosting algorithms using con dence-rated predictions. *Machine learning*, 37:80–91, 1999.

[73] G.A.F. Seber and C.J. Wild. *Nonlinear regression*, volume 503. John Wiley and Sons, 2003.

[74] P. Spector. *Data Manipulation with R (Use R)*. Springer, 1 edition, March 2008.

[75] C.J. Stone. Consistent nonparametric regression. *The annals of statistics*, 5(4):595–620, 1977.

[76] C.J. Stone. Additive regression and other nonparametric models. *The annals of Statistics*, 13:689–705, 1985.

[77] J.W. Tukey. *Exploratory data analysis*. Addison-Wesley, Reading, Mas., 1977.

[78] E. Van der Burg and J. de Leeuw. Non-linear canonical correlation. *British Journal of Mathematical and Statistical Psychology*, 1983.

[79] A.J. van der Kooij et al. *Prediction accuracy and stability of regression with optimal scaling transformations*. Child & Family Studies and Data Theory (AGP-D), Department of Education and Child Studies, Faculty of Social and Behavioural Sciences, Leiden University, 2007.

[80] A.J. van der Kooij and J.J. Meulman. Murals: Multiple regression and optimal scaling using alternating least squares. *Advances in Statistical Software*, pages 99–106, 1997.

[81] A.J. van der Kooij and J.J. Meulman. Regression with optimal scaling. *Meulman JJ, Heiser WJ, SPSS (eds): SPPS Categories 10.0*, 13:107–157, 1999.

[82] A.J. van der Kooij, J.J. Meulman, and W. Heiser. Local minima in categorical multiple regression. *Computational Statistics & Data Analysis*, 50(2):446–462, January 2006.

[83] A.J. van der Kooij, P. Neufeglise, and J.J. Meulman. Catreg, categorical multiple regression with optimal scaling (revised and updated version). *SPSS, Inc, Chicago*, 2001.

[84] V.N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, 2000.

[85] W.N. Venables and B.D. Ripley. *Modern applied statistics with S*. Springer verlag, 2002.

[86] G. Wahba. A survey of some smoothing problems and the method of generalized cross-validation for solving them. *Applications of Statistics*, 1977.

[87] L. Wasserman. *All of Nonparametric Statistics (Springer Texts in Statistics)*. Springer, May 2007.

[88] E.J. Wegman and I.W. Wright. Splines in statistics. *Journal of the American Statistical Association*, 78:351–365, 1983.

[89] S.N. Wood. *Generalized additive models: an introduction with R*, volume 66. CRC Press, 2006.

[90] H. Yanai, A. Okada, K. Shigemasu, Y. Kano, and J.J. Meulman. *New Developments in Psychometrics*. Springer, 2003.

[91] F.W. Young. Quantitative analysis of qualitative data. *Psychometrika*, 46(4):357–388, 1981.

[92] F.W. Young, J. De Leeuw, and Y. Takane. Regression with qualitative and quantitative variables: An alternating least squares method with optimal scaling features. *Psychometrika*, 41(4):505–529, 1976.