Ludwig-Maximilians-Universität München
Department of Statistics

Master-Thesis

Boosting Techniques for Nonlinear Time Series Models

Supervisors:

Prof. Dr. Gerhard Tutz
Dr. Florian Leitenstorfer

May 2008

Nikolay Robinzonov

Acknowledgments

Many people deserve sincere appreciation for their direct or indirect support during the creation of this thesis. I would like to thank:

Gerhard Tutz for his initial suggestion of a starting point for my thesis, followed by his excellent supervision,

Florian Leitenstorfer for his insightful advices and prompt support about technical issues,

Klaus Wohlrabe for providing me the data and for spending with me long hours of discussions at ifo institute, as well as for his comments and encouragements,

Nivien Shafik and Hristina Bojkova for their careful reading and critical notes during the correction of my thesis,

the whole BAYHOST team (Bayerische Hochschulzentrum für Mittel-, Ost- und Südosteuropa) for the financial support during my master studies,

my parents for supporting me even when I decided to go to Munich,

and Milena for her patience and care.

Contents

1 Introduction

Time series could be any sequence of data points measured at successive time intervals. An essential property of this sequence is its unclear evolution over time. Still, some paths remain more probable than others and this motivates researchers to try to understand the generating mechanism of the data points and possibly to forecast its future events, even before they have occurred. Linear models provide a starting point for modelling the nature of time series. Linear time series models, however, encounter various limitations in the real world and that makes them applicable only under certain, very restrictive, conditions. The field of time series has undergone various new developments in the last two decades, which relaxed some of these constraints. In particular, the development of nonparametric regression added more flexibility to the standard linear regression, which was adopted by the time series paradigm as well. A leading aspect to be explored throughout the thesis is the nonparametric modelling and the resulting forecasting techniques.

The second major aspect will concern the issue of high-dimensionality in the models, i.e. models with many covariates. The development of nonparametric analysis even in high dimensions was made possible thanks to the availability of suitable software and technological solutions. One of the most powerful strategies that deal with high-dimensional models comes from the machine learning community. The idea has undergone extensive evolution in the last decade and as a result, now we have available an ensemble procedure for regression under the name boosting. The novel component of the present work is the application of boosting to time series, which is done by letting the covariates be lagged values of a time series.

In the beginning of Section 2 we will explain the underlying idea of nonparametric regression through basis expansions and will emphasize how it links to linear regression. We will also exemplify basis expansions through splines and will explain the real breakthrough in the nonparametric regression, obtained through the application of penalized B-Splines.

Section 3 introduces the gradient-descent view of boosting, which is considered purely as a numerical optimization, rather than as a "traditional" statistical model. We will examine the structure of several boosting algorithms for continuous data and link them to the framework of statistical estimation. We will discuss the general ideas behind componentwise boosting of linear models. We will also study two possible strategies for componentwise boosting of additive models, built on top of penalized B-Splines and will finish with discussion on the theoretical grounds behind

multivariate linear boosting.

Section 4 will start with a close look of the foundation of univariate autoregressive time series analysis. We will discuss some of the aforementioned necessary conditions, which make the linear time series models work. Actually, even for these constrained classes of time series, as the linear autoregressive time series are, there are plenty of aspects that should be considered in order to provide a full research. Our intention is to address just the most relevant modelling principles in order to avoid overwhelming, but still provide a sound theoretical base. We will outline the relevant aspects of vector autoregressive times series models as well. The literature offers a great deal of modelling tools for nonlinear times series. Initially we will outline some of the common parametric nonlinear models, followed by an outlook of the mechanisms of the most popular nonparametric algorithms.

The results of a simulation study will be examined in Section 5. We will analyse the performance of boosting with P-Spline base learners in Monte Carlo simulations with six artificial, nonlinear, autoregressive time series. We will compare the outcomes of boosting to the outcomes, obtained through alternative nonparametric methods. Their performances will be considered in terms of lag-selection and goodness-of-fit.

Boosting of linear and additive model will be applied to real world data in Section 6. The target variable is the German industrial production. We will compare boosting, along with other methods, to the simple univariate autoregressive model in order to answer one very appealing question: do these sophisticated techniques actually manage to outperform the linear autoregressive model in terms of forecasting? Then we will extend the data set with exogenous variables, thus building several bivariate time series. We will include the standard tool from the macroeconomic forecasting field, namely the vector autoregressive model. Finally, we will create a real high-dimensional model by including all nine exogenous variables with their respective lags into one model and will examine the forecasting performance of boosting. In Section 7 we conclude.

2 Splines

In this section we will outline basic statistical methods for nonparametric modelling through basis expansions. Basis expansions are at the heart of many nonparametric methods, presented in this thesis. Therefore, we will reveal their key concepts in order to facilitate the exposition later on. Still, other nonparametric techniques are available in Hastie and Tibshirani (1990, Chapter 2); Hastie, Tibshirani, and Friedman (2001, Chapter 6); Fahrmeir and Tutz (2001, Chapter 5), among others. The exposition in this section will follow mainly: Eilers and Marx (1996), who made the real breakthrough in the development of splines for regression; Wood (2006), who provides a very thorough theoretical treatment and extensive material of practical application of splines, using the statistical software R (R Development Core Team, 2008); and will draw partly on Hastie et al. (2001) and Kneib (2003).

Section 2.1 will introduce the underlying idea of basis expansions and will emphasize on their connection with linear regression. In Sections 2.2 and 2.3 we will examine some simple basis examples such as polynomial splines, truncated splines and B-Splines. Subsequently, we will extend them to the fundamental concept of penalized splines in Section 2.4, which also relates to the paradigm of additive boosting, described later in the thesis.

2.1 Basics

The most famous statistical model is the simple linear model

$$\mathbf{y} = \mathbf{X}\tilde{\boldsymbol{\beta}} + \boldsymbol{\epsilon} \tag{2.1}$$

where vector $\mathbf{y} = (y_1, \ldots, y_n)^T$ contains n realizations of the continuous *response* variable y, $\mathbf{X} = (\mathbf{1}, \mathbf{x}_1, \ldots, \mathbf{x}_p)$, $\mathbf{x}_j = (x_{1j}, \ldots, x_{nj})^T$ is the *design matrix*, which summarizes an intercept and the realizations of say p *predictor* variables $\{x_1, \ldots, x_p\}$, also called *covariates* or *inputs*, $\tilde{\boldsymbol{\beta}}$ gathers the linear impact of each covariate at the response and an error term $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_n)^T$. The notation $\tilde{\boldsymbol{\beta}}$ helps us just to distinguish these parameters from the ones in (2.5) below and implies no unusual interpretation. There are several reasons for the popularity of this model, some of them being the convenient ways of estimation the unknown parameters in $\tilde{\boldsymbol{\beta}}$. Further on, these are easily interpretable and therefore preferred from non-statisticians as well.

However, in practice it is unlikely that the random variable y and the covariates $x_1, \ldots, x_p$ relate in a linear fashion. Therefore the framework of Generalized Additive

Models (GAM) (Hastie and Tibshirani, 1990) proposes an alternative for flexible specification of the dependence through

$$y_i = f_1(x_{i1}) + \ldots + f_p(x_{ip}) + \epsilon_i \tag{2.2}$$

where $x_{ij},\ i = 1, \ldots, n$, is the ith observation of the jth predictor and f_j's represent smooth functions which are to be estimated instead of the parameters in $\tilde{\boldsymbol{\beta}}$. Model structures such as (2.2) represent methods for moving "beyond linearity". The quotation marks serve to emphasize the common knowledge, that in fact, many nonlinear techniques are direct generalizations of the linear methods. The basic idea of basis expansion is to replace the inputs $x_1, \ldots, x_p$ by additional variables, which are their transformations, and then use the linear methods in this new space of derived input features. This can be done by choosing a *basis*, defining the space of say m completely known *basis functions*. Every single f_j in (2.2) is an element of this basis. Denote by $b_j^{[l]}(x_{ij})$ the lth transformation of x_{ij}, then f_j is assumed to have a representation

$$f_j(x_{ij}) = \sum_{l=1}^{m} b_j^{[l]}(x_{ij})\beta_j^{[l]} \tag{2.3}$$

and substituting (2.3) into (2.2) yields a clear linear model

$$\begin{aligned} y_i &= \sum_{l=1}^{m} b_1^{[l]}(x_{i1})\beta_1^{[l]} + \ldots + \sum_{l=1}^{m} b_p^{[l]}(x_{ip})\beta_p^{[l]} + \epsilon_i \\ &= \sum_{l=1}^{m}\sum_{j=1}^{p} b_j^{[l]}(x_{ij})\beta_j^{[l]} + \epsilon_i \end{aligned} \tag{2.4}$$

The latter could be equivalently represented in a matrix notation which leads to

$$\mathbf{y} = \mathbf{Z}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{2.5}$$

where $\mathbf{Z} = (\mathbf{Z}_1, \ldots, \mathbf{Z}_p)$ is the augmented $(n \times mp)$ design matrix, $\mathbf{Z}_1, \ldots, \mathbf{Z}_p$ are $(n \times m)$ matrices, each representing the basis transformation of the initial vectors $\mathbf{x}_1, \ldots, \mathbf{x}_p$ such that $\mathbf{Z}_j = (b_j^{[1]}(\mathbf{x}_j), \ldots, b_j^{[m]}(\mathbf{x}_j))$, $\boldsymbol{\beta} = (\boldsymbol{\beta}_1^T, \ldots, \boldsymbol{\beta}_p^T)^T$ is a $(mp \times 1)$ parameter vector, $\boldsymbol{\beta}_j = (\beta_j^{[1]}, \ldots, \beta_j^{[m]})^T$ and $\boldsymbol{\epsilon}$ is a $(n \times 1)$ error vector. One easily encounters the beauty of this approach, consisting in the similarity between (2.1) and (2.5). Once the basis functions $b_j^{[m]}$ have been determined, the models are linear in these new variables, and the fitting proceeds as in (2.1). Note, that interpreting the name *nonparametric* as absence of any parameters would be not quite precise, since $\boldsymbol{\beta}$ contains, indeed, very large number of parameters, which in turn have to be estimated. In order to precise the description, Eilers and Marx (1996) explain in their introduction basis expansion as an *overparametric* technique or equivalently *anonymous* model, in which the nature of parameters is not knows, i.e. they have

no direct statistical interpretation. Having noted this, however, we continue to follow the common concept, which describes basis expansions as a nonparametric technique.

For the sake of convenience we will restrict the number of the covariates $p = 1$ in the next two sections. Thus, avoiding the redundant notation (the jth index falls away) we will concentrate on the functionality of splines. Besides, the boosting algorithms presented in sections 3.2 and 3.3 imply *componentwise* processing with the covariates. The latter means that we are always working with *one* predictor at a time and therefore escape the need for simultaneous treatment of multiple predictors. This parsimony implies the following interpretation of the notation: $\boldsymbol{\beta} = \boldsymbol{\beta}_1, \mathbf{Z} = \mathbf{Z}_1, b^{[l]} = b_1^{[l]}$ and $f = f_1$.

2.2 Splines

Splines propose a very convenient way for choosing the aforementioned basis functions. Before we introduce splines, we define a *polynomial*. Polynomial is a mathematical expression involving a sum of powers in one or more variables multiplied by coefficients. A *polynomial spline* is a curve, made up of sections of polynomials (*piecewise polynomials*) joined together so that they are continuous in value, as well as derivatives up to the degree of the polynomial minus one. These polynomials are referred to as the *basis* functions. The points at which the sections join are known as the *knots* of the spline. Typically the knots are evenly spread over the domain and are used by the spline function to connect the neighbouring polynomial pieces in a special smooth way. That means, that having knot locations denoted by τ_m, $a = \tau_1, \ldots, < \tau_M = b$, $S : [a, b] \to \mathbb{R}$ consists of polynomial pieces $b_i : [\tau_i, \tau_{i+1}) \to \mathbb{R}$, S is said to be a polynomial spline of degree l if

1. b_i has degree at most l in the subintervals $[\tau_i, \tau_{i+1})$
2. S is $l - 1$ times continuously differentiable.

A general approach to splines can be found in the books by Wahba (1990) and Gu (2002).

Given the knot locations, there are many alternatives of writing down a basis for splines. For instance, a very simple spline function f of lth order could be represented via polynomial basis, such as:

$$b^{[1]}(x) = 1,\ b^{[2]}(x) = x,\ b^{[3]}(x) = x^2,\ b^{[4]}(x) = x^3, \ldots,\ b^{[l+1]}(x) = x^l.$$

This means that (2.3) could be written as

$$\begin{aligned} f(x) &= \beta_1\, b^{[1]}(x) + \beta_2\, b^{[2]}(x) + \beta_3\, b^{[3]}(x) + \beta_4\, b^{[4]}(x) + \ldots + \beta_l\, b^{[m]}(x) \\ &= \beta_1 + \beta_2\, x + \beta_3\, x^2 + \beta_4\, x^3 + \ldots + \beta_m\, x^{m-1}. \end{aligned}$$

Another option proposes the *regression spline*, represented as a linear combination of a truncated power basis,

$$b^{[1]}(x;l) = 1,\ b^{[2]}(x;l) = x \quad \text{and} \quad b^{[k+2]}(x;l) = (x-\tau_k)_+^l,$$

where

$$(x-\tau_k)_+^l = \begin{cases} (x-\tau_k)^l & \text{if } x \geq \tau_k, \\ 0 & \text{otherwise.} \end{cases}$$

One could find even more complicated examples such as the one in Gu (2002, pg. 37):

$$b^{[1]}(x) = 1, \quad b^{[2]}(x) = x \quad \text{and} \quad b^{[k+2]}(x) = R(x,\tau_k)$$

where

$$\begin{aligned} R(x,\tau_k) = &\left[\left(\tau_k - \frac{1}{2}\right)^2 - \frac{1}{12}\right]\left[\left(x - \frac{1}{2}\right)^2 - \frac{1}{12}\right]/4 \\ &- \left[\left(|x-\tau_k| - \frac{1}{2}\right)^4 - \frac{1}{2}\left(|x-\tau_k| - \frac{1}{2}\right)^2 + \frac{7}{240}\right]/24. \end{aligned}$$

However, despite the diversity of available options, all of these basis functions have one purpose, namely to map the scalar x_i into the ith row vector, of the augmented matrix $\mathbf{Z}$, denoted by $\mathbf{z}_{(i)}$, i.e.

$$x_i \mapsto \mathbf{z}_{(i)} = \left[b^{[1]}(x_i), \ldots, b^{[m]}(x_i)\right].$$

It should be noted that in case some subinterval $[\tau_i, \tau_{i+1})$ is empty, that would lead to a singular design matrix $\mathbf{Z}$ and consequently prevent the standard estimation process of $\boldsymbol{\beta}$ in (2.3). Since this it is rather unlikely to happen, we will always assume a full rank design matrix.

The proposed basis functions are relatively simple and straightforward to implement. However, they are not locally defined, which turns out to be a substantial drawback for the penalizing concept later on. A large number of knots often leads to numerical problems due to unbounded basis functions. Therefore, for regression problems we will explore one of the most convenient basis representations, namely the Basic Splines, hence B-Splines.

2.3 B-Splines

A primary reference for B-Splines in numerical analysis is De Boor (1978, 2001). A real breakthrough for statistics was made by Eilers and Marx (1996), who showed that penalized B-Splines are very attractive as basis functions for regression. Before we study their method in the next section, basic overview of B-Splines is proposed. The definition of the ith B-Spline basis of degree l is recursive, and namely:

$$B_i(x;l) = \frac{x-\tau_i}{\tau_{i+l}-\tau_i} B_i(x;l-1) + \frac{\tau_{i+l+1}-x}{\tau_{i+l+1}-\tau_{i+1}} B_{i-1}(x;l-1)$$

where

$$B_i(x;0) = \begin{cases} 1 & \text{if } x \in [\tau_i, \tau_{i+1}) \\ 0 & \text{otherwise.} \end{cases} \tag{2.6}$$

Figure 1 shows a sequence of B-Splines up to order three with evenly spaced knots from 0 to 1. The total number of knots needed for construction of k B-Splines of order l is $k + 2l + 1$. Note, that B-Splines of order l overlap with exactly $2l$ neighbours. In Figure 1, the leftmost and the rightmost splines, which have less overlap, are not depicted (see Appendix C.1, Figure 11 for the full illustration). With the assistance of the illustration in Figure 1 we could easily observe some of the appealing properties of B-Splines. Each function is nonzero over the intervals between $l+2$ adjacent knots, which means that the basis functions are strictly local. Moreover, in contrast to the polynomial basis, the basis functions are now bounded which leads to the numerical advantages and enhanced stability. Consider Eilers and Marx (1996) for further details on the appealing properties of B-Splines.

The application of basis expansions on regression is best seen by example. In Figure 2 is shown an illustration of how B-Splines of order three, also called *cubic splines*, are related to the regression framework. Suppose we have an original basis of six functions, as depicted in Figure 2(a). Each function of 2(a) is multiplied by a coefficient, which results in different curves, shown in 2(b). The coefficients are usually provided by the associated parameter vector $\hat{\boldsymbol{\beta}}$, estimated in (2.5). For an illustration purpose, let us choose: $\hat{\boldsymbol{\beta}} = (-0.2, -1.0, -0.8, 1.2, 0.5, 0.8)^T$. Then, the scaled functions from 2(b) are summed up, thus producing an estimation $\hat{f}$, depicted in 2(c).

Although B-Splines circumvent some of the major drawbacks of polynomial splines, they still require several subjective decisions in order to produce satisfying results. The first one, which is actually of minor importance, is the order of the B-Splines. The practice shows that the estimation procedure is not very sensitive to this option

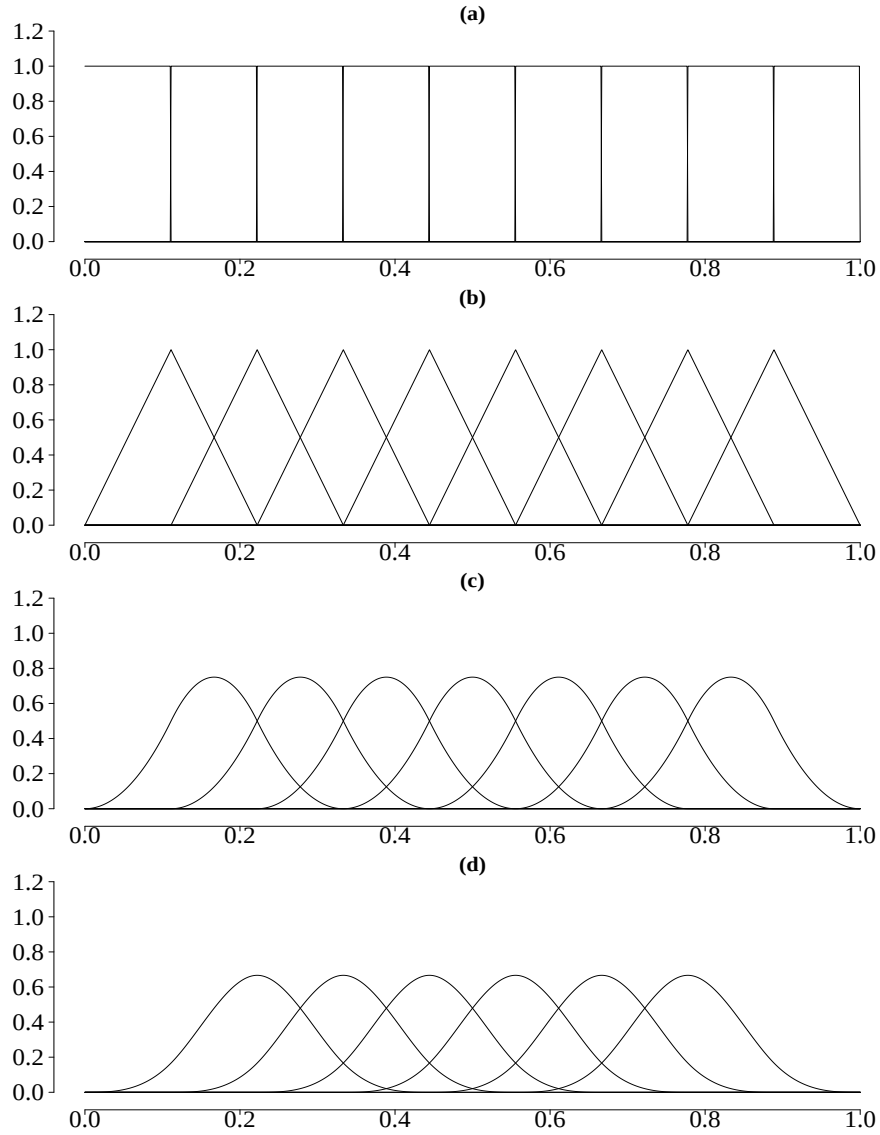


Figure 1: *A sequence of B-Splines of order 0 (a), 1 (b), 2 (c) and 3(d). The leftmost and rightmost splines are discarded.*

and B-Splines of order three are usually a reasonable choice. A quote of Hastie et al. (2001, p. 120) states that,

> *it is claimed that cubic splines are the lowest-order spline for which the knot-discontinuity is not visible to the human eye.*

The next option, however, is of major importance. It addresses the number of knots, which strongly influences the degree of smoothness. Too small number of knots leads to very smooth spline curve, which fits the data poorly. On the other hand, if the number of knots is too large the spline curve is very rough. This is a consequence of fitting the noise, as well as the underlying process. There are two common

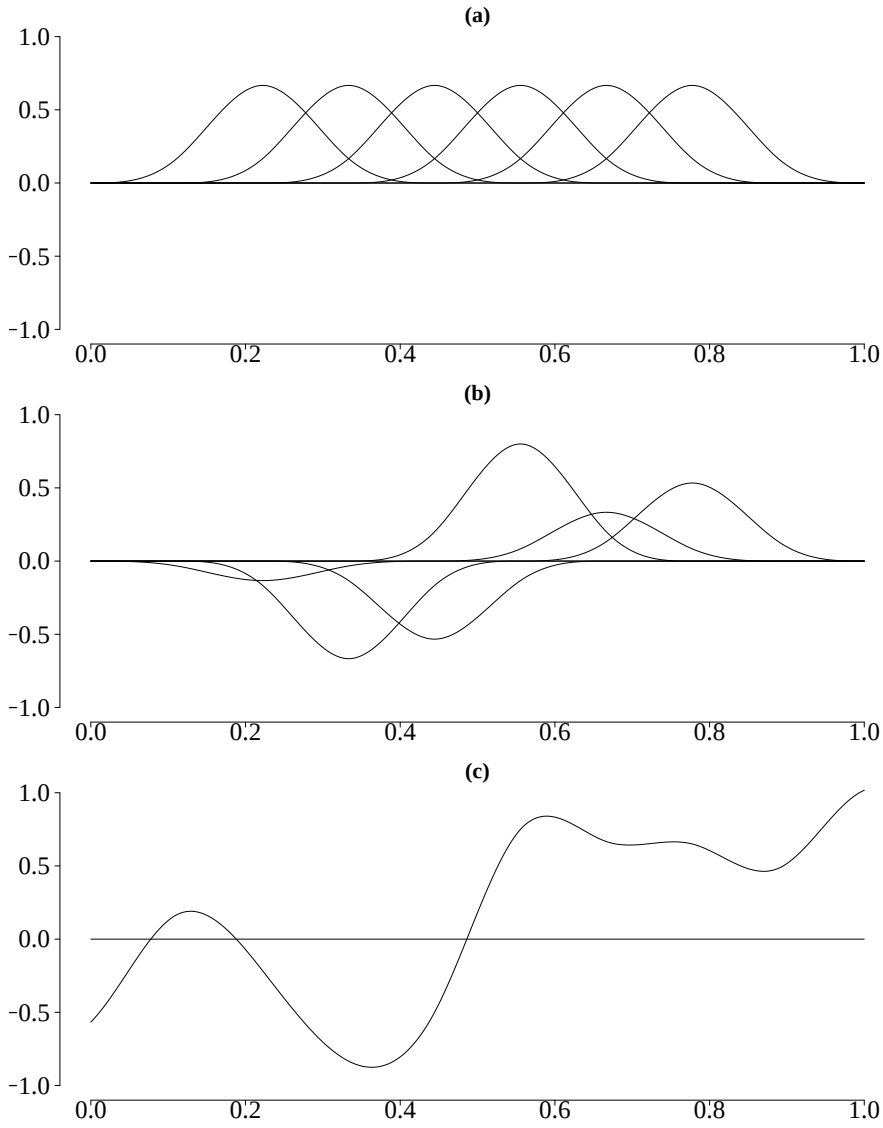


Figure 2: *Nonparametric regression with B-Splines. Panel (a) depicts cubic basis functions, panel (b) shows the basis functions, multiplied by their associated coefficients* $\hat{\boldsymbol{\beta}}$*. The sum of the scaled functions gives the spline itself, shown in panel (c).*

strategies that deal with this issue. One could apply a data-driven knot selection model (Friedman and Silverman (1989); Friedman (1991); Stone et al. (1997)) or use a maximal set of knots and penalize the curvature in the estimated function. Since one of the major topics in this thesis, namely additive boosting (Sections 3.2 and 3.3) develops estimation procedure, which is based on penalized splines, we will consider the penalization concept in the following section.

2.4 P-Splines

Before we get familiar with the so called P-Splines (Eilers and Marx, 1996), we briefly introduce the more general form of penalized regression splines, called *smoothing splines*. The appealing feature of the penalization concept is the fact that a single parameter could control the degree of smoothness. That means that, rather than minimizing

$$\sum_{i=1}^{n}(y_i - f(x_i))^2$$

one could minimize

$$\sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda \int \left(\frac{\partial^2 f}{\partial x^2}\right)^2 dx \tag{2.7}$$

where f has continuous first and second derivatives, and the second derivative is quadratically integrable. The resulting estimation is called a *natural cubic smoothing spline* (Reinsch, 1967). The first term in (2.7) measures the closeness to the data, the second penalizes the curvature in f and the (tuneable) *smoothing parameter* λ establishes the tradeoff between them. If $\lambda = 0$, then we clearly get the unpenalized estimator, which leads to a very rough function estimation. If we let $\lambda = \infty$, that would lead to the smoothest possible curve estimation, namely a straight line. It can be shown that the natural cubic smoothing spline is *unique* minimizer of (2.7) with knots at the values of x_i (Wood, 2006, p. 148). This results in n free parameters which are to be estimated.

Eilers and Marx (1996) propose a special form of penalized regression splines which greatly reduces the number of the parameters to estimate. They consider a discrete approximation of (2.7) with B-Splines and term it Penalized B-Splines, hence P-Splines. The main idea is that one keeps the basis dimension fixed, at a size a little larger than it is believed could reasonably be necessary (more than 30 knots do not indicate significant improvement) and penalizes directly the squared differences of the coefficients of adjacent basis functions. Moreover, the knots are evenly spaced and the recursive definition of B-Splines ensures very convenient computation of their derivatives. Thus, for any function f:

$$f(x) = \beta_1 B_1(x; l) + \ldots + \beta_m B_m(x; l) = \boldsymbol{\beta}^T \boldsymbol{B}(x; l) \tag{2.8}$$

De Boor (1978) showed that the first derivative of f could easily be calculated trough the following formula:

$$\begin{aligned}\frac{\partial f}{\partial x} &= \frac{1}{h}\sum_{i=1}^{m}\beta_i[B_i(x;l-1) - B_{i+1}(x;l-1)] \\ &= \frac{1}{h}\sum_{i=2}^{m}(\beta_i - \beta_{i-1})B_i(x;l-1) \\ &= \frac{1}{h}\sum_{i=2}^{m}\Delta\beta_i B_i(x;l-1) \end{aligned} \tag{2.9}$$

where h denotes the distance between two adjacent knots, Δ is the difference operator, recursively defined by

$$\begin{aligned}\Delta\beta_i &= \beta_i - \beta_{i-1} \\ \Delta^2\beta_i &= \Delta(\Delta\beta_i) = \beta_i - 2\beta_{i-1} + \beta_{i-2}.\end{aligned}$$

Later by induction Eilers and Marx (1996) showed that

$$\begin{aligned}\frac{\partial^2 f}{\partial x^2} &= \frac{1}{h^2}\sum_{i=3}^{m}(\beta_i - 2\beta_{i-1} + \beta_{i-2})B_i(x;l-2) \\ &= \frac{1}{h^2}\sum_{i=3}^{m}\Delta^2\beta_i B_i(x;l-2).\end{aligned} \tag{2.10}$$

It turns out that the second term in (2.7) can be approximated in the following way:

$$\int\left(\frac{\partial^2 f}{\partial x^2}\right)^2 dx = \int(\mathbf{D}_2\boldsymbol{\beta})^T\mathbf{D}_2\boldsymbol{\beta}\,dx = \boldsymbol{\beta}^T\mathbf{D}_2^T\mathbf{D}_2\boldsymbol{\beta} \tag{2.11}$$

where

$$\mathbf{D}_2 = \begin{bmatrix}1 & -2 & 1 & 0 & . \\ 0 & 1 & -2 & 1 & . \\ . & . & . & . & . \\ . & . & 1 & -2 & 1\end{bmatrix} \qquad (m-2\times m)$$

and therefore the approximation of (2.7) can be written as

$$||\mathbf{y} - \mathbf{Z}\boldsymbol{\beta}||^2 + \lambda\boldsymbol{\beta}^T\boldsymbol{\Lambda}\boldsymbol{\beta} \tag{2.12}$$

where $\boldsymbol{\Lambda} = \mathbf{D}_2^T\mathbf{D}_2$. Now minimizing (2.12) with respect to $\boldsymbol{\beta}$, one obtains the penalized least squares estimator

$$\hat{\boldsymbol{\beta}} = (\mathbf{Z}^T\mathbf{Z} + \lambda\boldsymbol{\Lambda})^{-1}\mathbf{Z}^T\mathbf{y} \tag{2.13}$$

(see the derivation in Appendix B.1) and therefore the *hat matrix* (or *influence matrix*) can be written as

$$\mathcal{H}(\lambda) = \mathbf{Z}^T(\mathbf{Z}^T\mathbf{Z} + \lambda\boldsymbol{\Lambda})^{-1}\mathbf{Z}^T. \tag{2.14}$$

The hat matrix is the one which yields the fitted vector, $\boldsymbol{\mu} = \mathbf{Z}\hat{\boldsymbol{\beta}} = \mathcal{H}\mathbf{y}$, when post-multiplied by the data vector $\mathbf{y}$. Moreover, it has the very useful property of determining the flexibility of the model. More precisely, the model's flexibility is determined by the *effective degrees of freedom*, defined by the trace of the hat matrix. Clearly, with λ set to zero the degrees of freedom of the model would be the dimension of $\boldsymbol{\beta}$. At the opposite extreme, if λ is very high then the model will be quite inflexible and will hence have very few degrees of freedom, equal to the order of difference penalty.

In Figure 3 are depicted three fits with different values of the smoothing parameter in order to gain a graphical impression of its impact. If λ is too high, then we have oversmoothing (or underfitting) and we miss the underlying dynamics of the real model. With λ too small the data will be undersmoothed (or overfitted), describing this way too much noise. These considerations will be of use once again when we discuss boosting of an additive model in the next section. The choice of λ will be, indeed, of minor importance, since we will always provide sufficiently large value for λ and will compensate the model inflexibility through the number of boosting iterations. Despite that, for the sake of integrity we will now briefly outline some basic techniques for a proper choice of the smoothing parameter. To see this topic in a greater depth, consider for example (Wood, 2006, Chapters 3 and 4).

Generally, we find an estimation of the smoothing parameter through numerical optimization of some information criterion with respect to λ. One possibility for estimating an "ideal" λ proposes the *ordinary cross validation* score, which estimates the expected squared error of the model. The idea consists of consecutive out-of-sample predictions of each value y_i. More precisely, we leave every (x_i, y_i)-pair out of the model, fit the remaining data and provide an estimation of y_i, based on x_i. Finally we calculate the squared difference between the estimated and the real value of y_i. Let $\hat{f}_\lambda^{[-i]}$ denote the model fit at x_i (also known as *jackknifed fit* at x_i). Then the formal definition of ordinary cross validation is

$$CV(\lambda) = \frac{1}{n}\sum_{i=1}^{n}(\hat{f}_\lambda^{[-i]} - y_i)^2. \tag{2.15}$$

Since it is inefficient to calculate n model fits, especially for large sample sizes, we could employ the hat matrix as a useful measure. It can be shown (Hastie and Tibshirani, 1990, p. 47-48) that (2.15) is equivalent to

$$CV(\lambda) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{f}_{\lambda,i}}{1 - \mathcal{H}_{ii}(\lambda)}\right)^2 \tag{2.16}$$

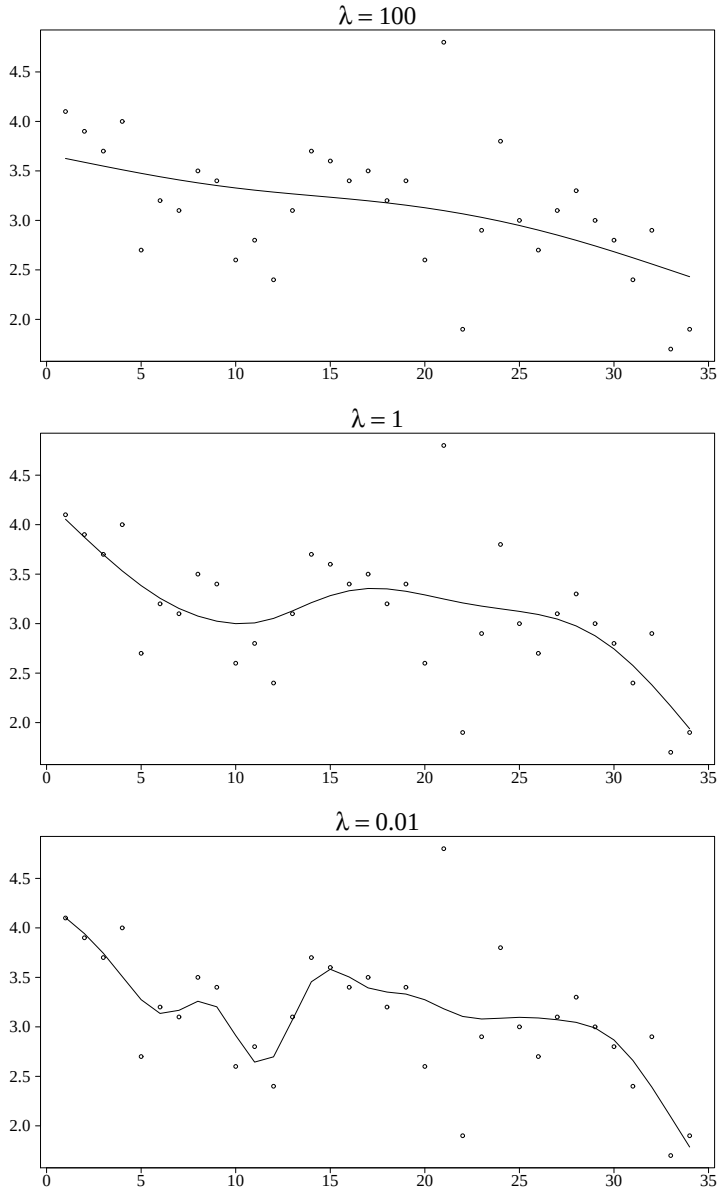


Figure 3: *Penalized regression spline fits using three different values for the smoothing parameter* λ.

where $\hat{f}_{\lambda,i}$ is the ith component of the estimate, produced by fitting the full data, and $\mathcal{H}_{ii}(\lambda)$ are the main diagonal elements of the hat matrix. In practice, $\mathcal{H}_{ii}(\lambda)$ is replaced by $\text{tr}(\mathcal{H}(\lambda))/n$ which defines the *Generalized Cross Validation* (GCV):

$$GCV(\lambda) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{f}_{\lambda,i}}{1 - \text{tr}(\mathcal{H}(\lambda))/n}\right)^2. \tag{2.17}$$

With $\text{tr}(\mathcal{H}(\lambda))$ we indicate the trace of the hat matrix. Thanks to its usefulness, various modelling strategies are based on slight modifications of the GCV criterion.

3 Boosting

Boosting, as one of the most powerful ideas in the machine learning community, has been a field of increased research interest in the last decade. Real breakthrough for a two class problem, i.e. response $y \in \{1, 0\}$, was made by Freund and Schapire (1996) with their AdaBoost algorithm.

Later Breiman (1996) provides experimental and theoretical evidence that his method "**b**ootstrap **agg**regat**ing**", hence bagging, can give substantial improvement in accuracy for both classification and regression prediction. Although unstable in prediction, his method makes a significant step towards optimality, by incorporating the idea of generating multiple versions of a predictor and then producing an aggregated predictor by plurality vote. A "committee" based approach is the basic concept of boosting as well. Breiman (1998) noted that the AdaBoost algorithm can be considered as a gradient descent optimization technique in function space. These findings opened perspective to consider boosting for regression problems which was successfully developed later by Friedman (2001). In the context of regression, the gradient boosting proposed by Friedman is essentially the same as Mallat and Zhang's (1993) matching pursuit algorithm in signal processing.

3.1 Gradient Boosting

3.1.1 Steepest Descent

The statistical framework developed by Friedman (2001) interprets boosting as a method for direct function estimation. He showed that boosting could be interpreted as a basis expansion, in which every single basis term is iteratively defined by the preceding ones. Suppose we have a random output variable y and a set of explanatory or input variables $\mathbf{x} = (x_1, \ldots, x_p)$ the goal is, using a "training" sample $\{y_i, \mathbf{x}_{(i)}\}_1^n$, where $\mathbf{x}_{(i)} \in \mathbb{R}^p$ is a p-dimensional row-vector, to obtain an estimate $\hat{f}(\mathbf{x})$ of the function $f(\mathbf{x})$, which maps $\mathbf{x}$ to y. To achieve this approximation one has to specify a loss function $L(y, f)$, and to minimize the expectation of this loss function with respect to f. A discussion about the specification of several loss functions follows in Section 3.1.2 below. The minimization of f can be viewed as a numerical optimization problem such as

$$\hat{f} = \arg\min_f L(y, f) \tag{3.1}$$

where

$$L(y, f) = \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_{(i)})). \tag{3.2}$$

A common procedure that solves (3.1) is to restrict $f(\mathbf{x})$ to be a member of a parameterized class of functions $f(\mathbf{x};\boldsymbol{\theta})$. For instance, $f(\mathbf{x};\boldsymbol{\theta})$ can be viewed as an additive expansion of the form

$$f(\mathbf{x};\boldsymbol{\beta},\boldsymbol{\gamma}) = \sum_{m=1}^{M} \beta_m \, h(\mathbf{x};\boldsymbol{\gamma}_m)$$

where the basis function $h(\mathbf{x};\boldsymbol{\gamma})$ is characterized by a set of parameters $\boldsymbol{\gamma}$, i.e. the members of this expansion differ in the parameters $\boldsymbol{\gamma}_m$. Thus, the original *function* optimization problem has been changed to a *parameter* optimization problem. That is, optimize

$$\{\hat{\boldsymbol{\beta}},\hat{\boldsymbol{\gamma}}\} = \arg\min_{\beta,\gamma} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_{(i)};\boldsymbol{\beta},\boldsymbol{\gamma})) \tag{3.3}$$

in order to achieve

$$\hat{f} = f(\mathbf{x};\hat{\boldsymbol{\beta}},\hat{\boldsymbol{\gamma}}). \tag{3.4}$$

In many situations, however, it is unfeasible to solve (3.3) directly and therefore an alternative numerical optimization method should be applied. One possibility is to try a *greedy stagewise* approach, which is

$$\{\beta_m,\boldsymbol{\gamma}_m\} = \arg\min_{\beta,\gamma} \sum_{i=1}^{n} L(y_i, f_{m-1}(\mathbf{x}_{(i)}) + \beta \, h(\mathbf{x}_{(i)};\boldsymbol{\gamma})) \tag{3.5}$$

followed by

$$f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \beta_m \, h(\mathbf{x};\boldsymbol{\gamma}_m). \tag{3.6}$$

In machine learning a strategy like the sequence (3.5)-(3.6) is called *boosting* and the function $h(\mathbf{x};\boldsymbol{\gamma})$ is termed a *weak learner* or a *base learner*. There are various modifications of the boosting strategy, differing mostly in the base learner. We will examine some of them in the following sections. However, the solution of (3.5) is not always feasible and requires a numerical optimization method itself. One such method is the *steepest-descent* optimization algorithm. Given any approximation $f_{m-1}(\mathbf{x})$, the increments $\beta_m \, h(\mathbf{x};\boldsymbol{\gamma}_m)$ in (3.5)-(3.6) are determined by computing the current gradient:

$$-g_m(\mathbf{x}_{(i)}) = \left[\frac{\partial L(y_i, f(\mathbf{x}_{(i)}))}{\partial f(\mathbf{x}_{(i)})}\right]_{f(\mathbf{x})=f_{m-1}(\mathbf{x})} \tag{3.7}$$

which gives the steepest-descent direction $-\mathbf{g}_m = -(g_m(\mathbf{x}_{(1)}),\ldots,g_m(\mathbf{x}_{(n)}))^T \in \mathbb{R}^n$. Note, that the gradient is defined only at the training data points and cannot be generalized to other $\mathbf{x}$-values. If the only goal was to minimize the loss on the training data the "steepest descent" would be sufficient. One possibility for generalization to new data, not presented in the training set, is to choose that $h(\mathbf{x};\boldsymbol{\gamma})$ which produces

$\mathbf{h}_m = (h(\mathbf{x}_{(1)}; \boldsymbol{\gamma}_m), \ldots, h(\mathbf{x}_{(n)}; \boldsymbol{\gamma}_m))^T$ most parallel to the gradient $-\mathbf{g}_m$, i.e. this $h(\mathbf{x}; \boldsymbol{\gamma})$, which is most highly correlated with the gradient $-g(\mathbf{x})$. This can be done simply by fitting $h(\mathbf{x}; \boldsymbol{\gamma})$ to the "pseudoresponse" $-g(\mathbf{x})$, i.e.

$$\boldsymbol{\gamma}_m = \arg\min_{\beta, \gamma} \sum_{i=1}^{n} (-g_m(\mathbf{x}_{(i)}) - \beta\, h(\mathbf{x}_{(i)}; \boldsymbol{\gamma}))^2. \tag{3.8}$$

Furthermore, a more powerful strategy would be to multiply the gradient vector with some constant in order to go into a "better" direction, which is called line search.[1] That means that we additionally compute

$$\rho_m = \arg\min_{\rho} \sum_{i=1}^{n} L(y_i, f_{m-1}(\mathbf{x}_{(i)}) + \rho\, h(\mathbf{x}_{(i)}; \boldsymbol{\gamma}_m)) \tag{3.9}$$

and finally the updates are determined via

$$f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \rho_m\, h(\mathbf{x}; \hat{\boldsymbol{\gamma}}_m).$$

Now we summarize steepest descent boosting as the following generic algorithm:

Generic Gradient Descent Boosting

1. Initialize $f_0 = \arg\min_\rho \sum_{i=1}^{n} L(y_i, \rho), \quad m = 0$
2. $m = m + 1$
3. $r_i = -\left[\frac{\partial L(y_i, f(\mathbf{x}_{(i)}))}{\partial f(\mathbf{x}_{(i)})}\right]_{f(\mathbf{x}) = f_{m-1}(\mathbf{x})}, \quad i = 1, \ldots, n$
4. $\boldsymbol{\gamma}_m = \arg\min_{\beta, \gamma} \sum_{i=1}^{n} (r_i - \beta\, h(\mathbf{x}_{(i)}; \boldsymbol{\gamma}))^2$
5. $\rho_m = \arg\min_\rho \sum_{i=1}^{n} L(y_i, f_{m-1}(\mathbf{x}_{(i)}) + \rho\, h(\mathbf{x}_{(i)}; \boldsymbol{\gamma}_m))$
6. $f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \rho_m\, h(\mathbf{x}; \boldsymbol{\gamma}_m)$
7. Iterate 2-6 until $m = M$

where the best value for M can be obtained via cross-validation.

3.1.2 Loss Functions

Expression (3.7) hints at the importance of the prechosen loss function. In addition to the weak learner, this is the second major option which determines the nature of the generic algorithm. Therefore several options for choosing the loss are briefly discussed in the sequel.

One of the frequently employed loss functions $L(y, f(\mathbf{x}))$ is the squared-error loss,

[1] It should be noted that the line search can be automatically provided through β when computing (3.8) and working with certain loss functions. See next section for further details.

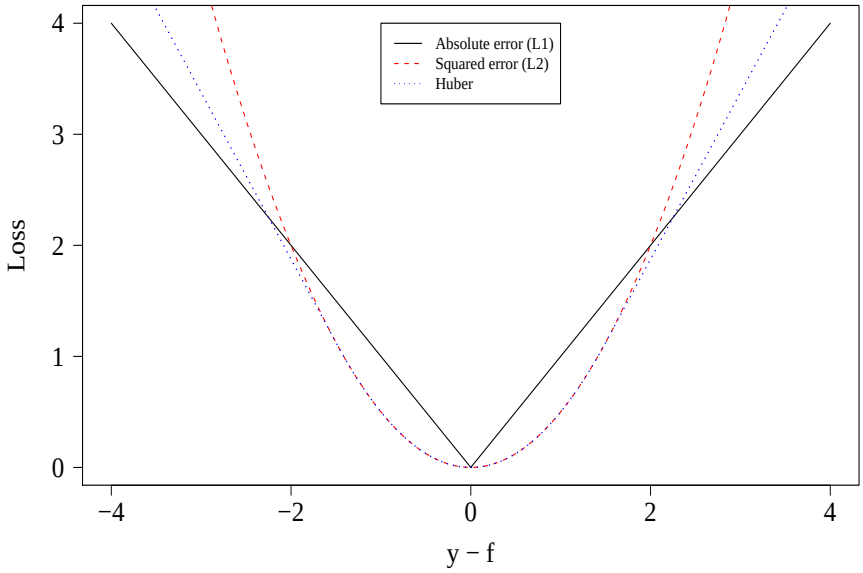


Figure 4: *A comparison of three loss functions for regression, plotted as a function of* $y - f(\mathbf{x})$.

also called L_2-loss, $L(y, f(\mathbf{x})) = \frac{1}{2}(y - f(\mathbf{x}))^2$. It is scaled by the factor $\frac{1}{2}$ thus ensuring a convenient representation of its first derivative (simply the residuals) which becomes very useful at line 3 of the generic algorithm. An absolute-error loss $L(y, f(\mathbf{x})) = |y - f(\mathbf{x})|$ or L_1-loss is another famous example for loss criterion. Although not differentiable at the points $y_i = f(\mathbf{x}_{(i)})$, partial derivatives of L_1-loss could be computed due to the zero probability of realizing single point $y_i = f(\mathbf{x}_{(i)})$ by the data. The last one to examine is the Huber loss criterion used for M-regression (Huber, 1964) and defined as:

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

where a strategy for adaptively changing δ is proposed by Friedman (2001):

$$\delta_m = \text{median}\left(\{|y_i - f_{m-1}(\mathbf{x}_{(i)})|;\ i = 1, \ldots, n\}\right).$$

Figure 4 proposes a graphical interpretation of the loss functions. The squared-error loss penalizes observations with large absolute residuals more heavily than the other two criteria. Thus L_2-loss is far less robust and its performance degrades for distributions with heavy tails and especially by presence of outliers. L_1-loss penalizes the extreme margins only linearly and thus Huber-loss is somewhat a compromise between them.

Furthermore, a very convenient property is the computational simplicity of the gradients of these loss functions. The loss criteria and the corresponding gradients are summarized in Table 1. Recall the generic gradient boosting strategy. Applied

Loss Function	$\partial L(y, f(\mathbf{x}))/\partial f(\mathbf{x})$
L_1	$\text{sign}\{y - f(\mathbf{x})\}$
L_2	$y - f(\mathbf{x})$
Huber	$y - f(\mathbf{x})$ for $\lvert y - f(\mathbf{x})\rvert \leq \delta_m$
	$\delta_m \text{ sign}\{y - f(\mathbf{x})\}$ for $\lvert y - f(\mathbf{x})\rvert > \delta_m$

Table 1: *Gradients for commonly used loss funcitons.*

to the most popular L_2-loss, it turns out that Least Squares Boosting (LS_Boost, Friedman (2001)) is nothing more than repeated least squares fitting of residuals (see line 4 of the generic gradient descent boosting). Furthermore, line 5 (the line search) is not needed anymore because $\rho_m = \beta_m$ and β_m was computed already at line 4. Essentially the same procedure with $M = 2$ has been proposed by Tukey (1977) and termed "twicing". Gradient boosting with squared-error loss produces the following algorithm:

LS_Boost

1. $f_0(\mathbf{x}) = \bar{\mathbf{y}}, \quad m = 0$
2. $m = m + 1$
3. $r_i = y_i - f_{m-1}(\mathbf{x}_{(i)}), \quad i = 1, \ldots, n$
4. $(\beta_m, \boldsymbol{\gamma}_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{n} (r_i - \beta\, h(x_i; \boldsymbol{\gamma}))^2$
5. $f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \beta_m\, h(\mathbf{x}; \boldsymbol{\gamma}_m)$
6. Iterate 2-5 until $m = M$

where the best value for M is usually determined via cross-validation.

3.1.3 Regularization

One virtual problem of prediction is encountered, when the training data is fitted too closely. This hinders the good prediction when working with new data and is called *overfitting*. The impressive performance of boosting is mainly due to its resistance to overfitting. Initially, this appealing property was observed empirically, until Bühlmann and Yu (2003) provided an analytical proof. The key to this resistance comes at the price of one extra parameter introduced by a regularization method

called *shrinkage*. It attempts to prevent overfitting by constraining the fitting procedure with a shrinkage factor. The simple strategy is to replace line 6 of the generic algorithm (respectively line 5 of the LS_Boost) with

$$f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \nu\, \rho_m\, h(\mathbf{x}; \boldsymbol{\gamma}_m) \tag{3.10}$$

where ν is the shrinkage factor. On the other hand, as the boosting iterations evolve, the estimation model has more terms which suggests the "natural" way of overfit prevention - providing small number of covariates, i.e. small M. It turns out that we have two instruments for prevention which work in different manners. Regularizing by controlling the number of influence terms suggests that "sparse" approximations, i.e. models which involve fewer terms, are believed to provide better prediction. However, it has often been found that regularization through shrinkage provides superior results to that obtained by restricting the number of covariates. The shrinkage can be regarded as controlling the learning rate of the boosting procedure. Roughly speaking, it provides the weak learner to be "weak" enough, i.e. the base learner has large bias but low variance. Nevertheless, these parameters do not operate independently and therefore mutually affect their performances. Decreasing the values of ν increases the best value for M, so there is a tradeoff between them. Ideally one should estimate optimal values for both by minimizing a model selection criterion jointly with respect to the values of both parameters. The performance of ν is examined rather empirically and Friedman (2001) was the first to demonstrate that small values ($\nu = 0.3$) are good in sense of low sensitivity of the boosting procedure.

3.1.4 Boosting with Linear Operator

Bühlmann and Yu (2003) made the next significant contribution to the intensively developing boosting framework. Their work contributed in several aspects, most notably: they proposed a learner via linear operator $\mathcal{S}$, which was appropriately adjusted later for the componentwise approach; they managed to prove an exponential dependence between the bias and the variance of the boosted model, which is the reason for the overfit resistane[2] of boosting; they showed how smoothing splines can be adopted by the boosting base procedure. In this section we will consider the first aspect.

Roughly speaking, the key idea is to represent the learner as a linear operator (or *smoother*) $\mathcal{S} : \mathbb{R}^n \to \mathbb{R}^n$, which yields the fitted values when post-multiplied by

[2]It was shown that addition of new terms in the model does not linearly increase its complexity, but rather with exponentially diminishing amounts.

the pseudo response. Let us denote $\mathbf{h}_m = (\beta_m\, h(\mathbf{x}_{(1)}; \boldsymbol{\gamma}_m), \ldots, \beta_m\, h(\mathbf{x}_{(n)}; \boldsymbol{\gamma}_m))^T$, $\mathbf{f}_m = (f_m(\mathbf{x}_{(1)}), \ldots, f_m(\mathbf{x}_{(n)}))^T$, $\mathbf{y} = (y_1, \ldots, y_n)^T$ and $\mathbf{r}_m = \mathbf{y} - \mathbf{f}_{m-1}$. Furthermore we use the basic relation from the generic boosting algorithm $\mathbf{f}_m = \mathbf{f}_{m-1} + \mathbf{h}_m$ with $\mathbf{h}_m = \mathcal{S}\mathbf{r}_m$ to provide the relationship

$$\begin{aligned}\mathbf{r}_m &= \mathbf{y} - \mathbf{f}_{m-2} - \mathcal{S}\mathbf{r}_{m-1} \\ &= \mathbf{r}_{m-1} - \mathcal{S}\mathbf{r}_{m-1}.\end{aligned}$$

Since $\mathbf{f}_0 = \mathcal{S}\mathbf{y}$, then follows $\boldsymbol{r}_1 = (I - \mathcal{S})\mathbf{y}$ and we obtain

$$\mathbf{r}_m = (I - \mathcal{S})^m \mathbf{y}. \tag{3.11}$$

Consequently

$$\mathbf{f}_m = \mathbf{y} - \mathbf{r}_{m+1} = (I - (I - \mathcal{S})^{m+1})\mathbf{y}$$

and finally the operator that maps the initial response vector $\mathbf{y}$ to $\mathbf{f}_m$ is termed *Boosting operator*, that is

$$\mathcal{B}_m = I - (I - \mathcal{S})^{m+1}. \tag{3.12}$$

Expression (3.12) enables us to define the presence of "learning capacity" of the model such that $\|I - \mathcal{S}\| < 1$. In addition, $\mathcal{B}_m$ converges to the identity I as $m \to \infty$, thereby $\mathcal{B}_m\mathbf{y}$ converges to the fully saturated model $\mathbf{y}$, interpolating the response exactly. However, using the same smoother $\mathcal{S}$ still does not explicitly suggest variable selection and is therefore best applicable for univariate problems. In the following Section 3.2 we will shown how this flexibility can be adopted by boosting even for high-dimensional models, where the number of predictor variables is allowed to grow very quickly.

3.2 Boosting High-Dimensional Models

Bühlmann (2006) provided an essential boosting technique, called L_2Boost, for regression problems with rapidly growing number of predictor variables. The key idea in his method is to exercise the weak learner upon *one* variable at a time and to pick out only those components with the "largest contribution to the fit". That is another way of keeping the learner "weak" enough, i.e. having low variance relative to the bias, which is done simply by restraining of a complex structure with many parameters. Thus, he also manages to incorporate an original predictor selection stage into the boosting paradigm. The arbitration of the model's complexity is another distinctive feature of the new boosting technique. This complexity has an essential role for defining the stopping condition of boosting. It is not required to

run the algorithm multiple times for cross-validation, as commonly used by then. A derivation of the hat matrix is provided and the complexity is determined by its trace. Using that trace, one employs a corrected version of an AIC (Hurvich, Simonoff, and Tsai, 1998) to define the stopping criterion for the boosting algorithm. These novelties are illustrated in the sequel.

3.2.1 Componentwise Linear L_2Boost

We consider a linear model with p covariates, $x_1, \ldots, x_p$ and a response variable y, with $\mathbf{x}_j$ denoting a n-dimensional vector with realizations of x_j, and x_{kj} the kth element of this vector. The essential modification of the new strategy concerns the base learner, which is forced to do *componentwise* selection among the predictors at each boosting stage. The base learner $h(x)$ works as follows:

Componentwise linear least squares learner

$$\begin{aligned} h(x_{\hat{s}}) &= \hat{\beta}_{\hat{s}}\, x_{\hat{s}} \\ \hat{\beta}_j &= \frac{(\mathbf{x}_j - \bar{\mathbf{x}}_j)^T \mathbf{r}}{(\mathbf{x}_j - \bar{\mathbf{x}}_j)^T (\mathbf{x}_j - \bar{\mathbf{x}}_j)}, \quad j = 1, \ldots, p \\ \hat{s} &= \arg\min_{1 \le j \le p} \sum_{i=1}^{n} (r_i - \hat{\beta}_j\, x_{ij})^2 \end{aligned} \tag{3.13}$$

where $\mathbf{r} = (r_1, \ldots, r_n)^T$ is the gradient of the loss function, used as a pseudo response. Thus, the base procedure fits a simple linear regression with every single covariate and selects the one that reduces the residual sums of squares most. So we have a "built-in" predictor selection procedure. Finally, the L_2Boost algorithm can be summarised in the following scheme:

Componentwise boosting of linear model

1. Initialize $\mathbf{f}_0 = \bar{\mathbf{y}}\mathbf{1}$, set $m = 0$.
2. $m = m + 1$
3. $\mathbf{r}_m = \mathbf{y} - \mathbf{f}_{m-1}$
4. Find $\hat{s}_m$ as in (3.13).
5. Update $\mathbf{f}_m = \mathbf{f}_{m-1} + \nu\, \mathbf{h}(x_{\hat{s}_m})$.
6. Iterate 2-5 until $m = M$

where ν is the shrinkage parameter and should accordingly be kept small, e.g. $\nu = 0.1$, $\mathbf{1} = (1, \ldots, 1)^T$ and $\mathbf{h}(x_{\hat{s}_m}) = (h(x_{1,\hat{s}_m}), \ldots, h(x_{n,\hat{s}_m}))^T$. Due to the additive

structure in (5), the final estimate can again be interpreted as an additive model, but the componentwise selection suggests that it typically depends on a subset of the original p covariates.

Unlike the common practice in gradient boosting, the stopping condition M for L_2Boost is determined via the computationally more efficient AIC_c information criterion (3.17) defined below. The first requirement for providing this criterion is to determine the model complexity. Therefore, we assign degrees of freedom for boosting. Denote by

$$\mathcal{H}^{(j)} = \frac{\mathbf{x}_j \mathbf{x}_j^T}{\|\mathbf{x}_j\|^2}, \quad j = 1, \ldots, p \tag{3.14}$$

the $(n \times n)$ hat matrix for the linear squares fitting operator using the jth predictor. It acts similarly to the linear operator $\mathcal{S}$ in Section 3.1.4 but employs each time different covariate. The denominator $\|\mathbf{x}_j\|^2$ denotes the Euclidean norm for a n-dimensional vector. Recall the common knowledge that the hat matrix yields the fitted vector when post-multiplied by the (pseudo) response vector, i.e. $\mathbf{h}(x_{\hat{s}_m}) = \mathcal{H}^{(\hat{s}_m)}\mathbf{r}_m$. Moreover $\mathbf{r}_m = \mathbf{y} - \mathbf{f}_{m-1}$ and $\mathbf{f}_m = \mathbf{f}_{m-1} + \nu\,\mathbf{h}(x_{\hat{s}_m})$ which is followed by

$$\begin{aligned} \mathbf{r}_m &= \mathbf{y} - \mathbf{f}_{m-2} - \nu\,\mathcal{H}^{(\hat{s}_{m-1})}\mathbf{r}_{m-1} \\ &= \mathbf{r}_{m-1} - \nu\,\mathcal{H}^{(\hat{s}_{m-1})}\mathbf{r}_{m-1} \end{aligned}$$

and we have

$$\mathbf{r}_m = (I - \nu\,\mathcal{H}^{(\hat{s}_{m-1})}) \ldots (I - \nu\,\mathcal{H}^{(\hat{s}_2)})(I - \nu\,\mathcal{H}^{(\hat{s}_1)})\mathbf{y}. \tag{3.15}$$

Then

$$\mathbf{f}_m = \mathbf{y} - \mathbf{r}_{m+1} = (I - (I - \nu\,\mathcal{H}^{(\hat{s}_m)}) \ldots (I - \nu\,\mathcal{H}^{(\hat{s}_1)}))\mathbf{y}$$

and finally the L_2Boost hat matrix at stage m equals

$$\mathcal{H}_{(m)} = I - (I - \nu\,\mathcal{H}^{(\hat{s}_m)}) \ldots (I - \nu\,\mathcal{H}^{(\hat{s}_2)})(I - \nu\,\mathcal{H}^{(\hat{s}_1)}). \tag{3.16}$$

Note that $\mathcal{H}^{(\hat{s})}$ and $\mathcal{H}_{(m)}$ are hat matrices in different regression problems, i.e. $\mathcal{H}^{(\hat{s})}$ maps the pseudo response while $\mathcal{H}_{(m)}$ maps the initial response. The L_2Boost hat matrix $\mathcal{H}_{(m)}$ ultimately depends upon the selected components $\hat{s}_1, \ldots, \hat{s}_m$. This is a direct consequence of the componentwise approach. Therefore, the term "hat matrix" is in someway liberally transferred to $\mathcal{H}_{(m)}$ and more precisely should be viewed as an approximate hat matrix. Conversely, the Boosting operator (3.12) uses the same operator (or smoother) at every stage, thus excluding the componentwise fashion of modelling.

The degrees of freedom, provided by the trace of $\mathcal{H}_{(m)}$, are employed in a corrected

version of AIC in order to define a stopping condition for boosting:

$$AIC_c(m) = \log(\hat{\sigma}^2) + \frac{1 + \text{tr}(\mathcal{H}_{(m)})/n}{(1 - \text{tr}(\mathcal{H}_{(m)}) + 2)/n} \tag{3.17}$$
$$\hat{\sigma}^2 = n^{-1} \sum_{i=1}^{n} (y_i - (\mathcal{H}_m \mathbf{y})_i)^2.$$

The number of boosting iterations is then estimated

$$\hat{M} = \arg \min_{1 \leq m \leq M} AIC_c(m)$$

where M is large enough to be used as an upper bound for the candidate number of iterations.

3.2.2 Componentwise Additive L_2Boost

The final remarks in Section 3.1.4 suggested the possibility of adding a nonparametric procedure to the boosting framework. Therefore, we assume additive expansion[3] of the predictors. In this case, a smooth function is fitted to the negative gradient of the loss function in each iteration, i.e. the parametric least squares learner from the previous section is simply substituted by its "nonparametric" (or overparametric) counterpart. Bühlmann and Yu (2003) have shown that choosing smoothing splines as a base procedure is very competitive to standard nonparametric models. Later, Schmid and Hothorn (2007) investigated whether boosting with smoothing spline base learners can be successfully approximated by boosting with P-Spline base learners.[4] Similarly to regression, it turned out that P-Splines, which are more advantageous from a computational point of view, propose very good approximation of smoothing splines. Recall that $\mathbf{Z}_1, \ldots, \mathbf{Z}_p$ represent the basis transformations of the initial covariates $\mathbf{x}_1, \ldots, \mathbf{x}_p$ such that $\mathbf{Z}_j = (b_j^{[1]}(\mathbf{x}_j), \ldots, b_j^{[B]}(\mathbf{x}_j))$ is a $(n \times B)$ matrix. Then we have

$$\mathbf{h}(x_s) = \mathbf{Z}_s \boldsymbol{\beta}_s. \tag{3.18}$$

Consequently $\boldsymbol{\beta}_s$ is estimated via penalized least squares estimator as in (2.13). The base procedure is then:

Componentwise P-Splines as base procedure

[3]Note that the term additive expansion can be used in two different contexts. Here we suggest an initial additive expansion of the *covariates*, which should be clearly distinguished from the interpretation of the *boosting iterations* as additive expansions themselves.

[4]We will briefly discuss in the next section the use of P-Spline base learners in an alternative boosting procedure, called *likelihood boosting* (Tutz and Binder, 2006), and show when both strategies coincide.

$$\begin{aligned} \mathbf{h}(x_{\hat{s}}) &= \mathbf{Z}_{\hat{s}}\hat{\boldsymbol{\beta}}_{\hat{s}} \\ \hat{\boldsymbol{\beta}}_j &= (\mathbf{Z}_j^T\mathbf{Z}_j + \lambda\boldsymbol{\Lambda})^{-1}\mathbf{Z}_j^T\boldsymbol{r}, \quad j = 1, \ldots, p \\ \hat{s} &= \arg\min_{1\leq j\leq p} \|\mathbf{r} - \mathbf{h}(x_j)\| \end{aligned} \tag{3.19}$$

where $\boldsymbol{\Lambda}$ is the penalty matrix.

The inevitable price that we pay for increased flexibility consists of additional parameters. Now we should choose not only an appropriate shrinkage factor ν, but also smoothing parameter λ and number of evenly spaced knots. Schmid and Hothorn (2007) carried out a thorough analysis of the effect of the various parameters on the boosting fit and provided very intriguing conclusions. They proved an approximately *linear* dependence between the number of the boosting iterations and ν for regression with one-dimensional covariate and found empirical evidence that the same relationship also holds true for higher dimensions. This implies that the AIC_c criterion (3.17) automatically adapts the stopping value for the iterations to the shrinkage factor.

Furthermore, note that λ determines the degrees of freedom (df) of the base learner. High values of λ lead to low degrees of freedom which is preferable in order to keep the learner "weak". Roughly speaking, λ acts like the shrinkage factor above by reducing the learning rate of the base procedure. It was proposed by Schmid and Hothorn (2007) $df = 3 - 4$ as a suitable amount for the degrees of freedom. Their results, concerning the number of knots confirmed the common knowledge that there is a minimum number of necessary knots which have to be provided and the algorithm is not sensitive to this choice (20-50 knots should be sufficient). Finally, the insertion of the new learner in the boosting paradigm is rather straightforward.

3.3 Likelihood Boosting

Likelihood boosting, proposed by Tutz and Binder (2006), retains the especially useful "built-in" selection feature for high-dimensional models. Besides, it manages to generalize the "boosts" for a response following a simple exponential family like binomial, poisson or Gaussian. This is done by maximizing the likelihood in generalized additive models for all kinds of link functions. The procedure is referred to as GAMBoost. One of the most advantageous innovations in GAMBoost is the relation of the Newton-Raphson method to boosting. It also restrains the explicit usage of the proposed loss functions by incorporating an information AIC criterion instead. AIC is additionally used as stopping condition.

The theory of maximum likelihood estimation is referred to the estimation of Gen-

eralized Linear Models (GLM). A GLM (Nelder and Wedderburn, 1972) allows for the response distributions other than normal and has a basic structure

$$\mu_i = h(\eta_i) = h(\mathbf{x}_{(i)}\boldsymbol{\beta}) \tag{3.20}$$

instead of the linear predictor

$$\mu_i = \eta_i = \mathbf{x}_{(i)}\boldsymbol{\beta}$$

where $\mathbf{x}_{(i)}$ is a p-dimensional predictor vector, $\mu_i = \mathrm{E}\,(y_i|\mathbf{x}_{(i)})$ and h is a specified response function. General assumptions for y_i's are their independence and belonging to some exponential family, i.e.

$$f_\theta(y) = \exp\{(y\theta - b(\theta))/a(\phi) + c(y,\phi)\}$$

where θ is a canonical parameter which completely depends on $\boldsymbol{\beta}$, ϕ is an arbitrary dispersion parameter and a, b and c are arbitrary functions. The log likelihood of θ, given a particular y, is the $\log(f_\theta(y_i))$ considered as a function of θ and *not* of y anymore. Furthermore the log likelihood of θ defines ultimately the log likelihood of $\boldsymbol{\beta}$, that is

$$l(\boldsymbol{\beta}) = \sum_{i=1}^{n} \log(f_{\theta_i}(y_i)) = \sum_{i=1}^{n} (y_i\theta_i - b(\theta_i))/a(\phi) + c(\phi, y_i). \tag{3.21}$$

Unfortunately, if the response is not Gaussian, there is no solution in closed form for (3.21), i.e. it demands numerical optimization methods such as Iteratively Reweighted Least Squares (IRLS) or the Newthon-Raphson method. In combination with an additive structure in the covariates, fitting of model (3.21) is based on maximizing the penalized likelihood

$$l_{(p)} = l(\boldsymbol{\beta}) - \frac{\lambda}{2}\boldsymbol{\Lambda}\boldsymbol{\beta}.$$

At this stage a modified version of the Newton-Raphson method should be applied to obtain an estimation of $\boldsymbol{\beta}$. It is done via the so called penalized score function $s_p(\boldsymbol{\beta}) = s(\boldsymbol{\beta}) - \lambda\boldsymbol{\Lambda}\boldsymbol{\beta}$, where

$$s(\boldsymbol{\beta}) = \mathbf{Z}_j^T\mathbf{D}(\boldsymbol{\beta})\boldsymbol{\Sigma}(\boldsymbol{\beta})^{-1}(\mathbf{y} - \boldsymbol{\mu}) = \mathbf{Z}_j^T\mathbf{W}(\boldsymbol{\beta})\mathbf{D}(\boldsymbol{\beta})^{-1}(\mathbf{y} - \boldsymbol{\mu})$$

with

$$\mathbf{Z}_j^T = (\mathbf{z}_{1j}, \ldots, \mathbf{z}_{nj}) \quad \text{the augmented design matrix,}$$

$$\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)^T, \quad \eta_i = \hat{\eta}_{(m)}(\mathbf{x}_{(i)}) + \mathbf{z}_{ij}^T\boldsymbol{\beta}_j,$$

$$\mathbf{D}(\boldsymbol{\beta}) = \begin{pmatrix} \partial h(\eta_1)/\partial\eta & . & . \\ . & \ddots & . \\ . & . & \partial h(\eta_n)/\partial\eta \end{pmatrix}$$

$$\boldsymbol{\Sigma}(\boldsymbol{\beta}) = \begin{pmatrix} \sigma_1 & . & . \\ . & \ddots & . \\ . & . & \sigma_n \end{pmatrix}$$

$$\sigma_i^2 = \mathrm{Var}_\beta(y_i), \quad \mathbf{W}(\boldsymbol{\beta}) = \mathbf{D}(\boldsymbol{\beta})\boldsymbol{\Sigma}(\boldsymbol{\beta})^{-1}\mathbf{D}(\boldsymbol{\beta}).$$

The penalized Fisher matrix

$$F_p(\boldsymbol{\beta}) = \mathrm{E}\left(\frac{-\partial^2 l_p(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^T}\right)$$

has the form $F_p = F(\boldsymbol{\beta}) + \lambda\boldsymbol{\Lambda}$, where $F(\boldsymbol{\beta}) = \mathrm{E}\left(-\partial^2 l(\boldsymbol{\beta})/\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^T\right) = \mathbf{Z}_j^T\mathbf{W}(\boldsymbol{\beta})\mathbf{Z}_j$. One Fisher scoring step is then

$$\hat{\boldsymbol{\beta}}_{new} = \hat{\boldsymbol{\beta}} + F_p(\hat{\boldsymbol{\beta}})^{-1}s_p(\hat{\boldsymbol{\beta}})$$

and starting with an initial guess $\boldsymbol{\beta}_{(0)}$ the solution is found through successive improvements of $\boldsymbol{\beta}$. Since with boosting one successively corrects the already fitted terms, at this stage we observe the most innovative feature of GAMBoost. Likelihood boosting requires only *one* step of the Fisher scoring algorithm and the estimations $\hat{\boldsymbol{\beta}}_{new}$ are derived simply by refitting the residuals. For further details see Tutz and Binder (2006).

Assuming the special case of a Gaussian response, the notation is consistent with Section 2.1: $\mathbf{f} = \boldsymbol{\mu} = \mathbf{Z}\hat{\boldsymbol{\beta}}$, where $\mathbf{Z} = (\mathbf{Z}_1, \ldots, \mathbf{Z}_p)$ is the augmented $(n \times Bp)$ design matrix, $\mathbf{Z}_1, \ldots, \mathbf{Z}_p$ represent the basis transformations of the initial covariates $\mathbf{x}_1, \ldots, \mathbf{x}_p$, such that $\mathbf{Z}_j = (b_j^{[1]}(\mathbf{x}_j), \ldots, b_j^{[B]}(\mathbf{x}_j))$, $\hat{\boldsymbol{\beta}} = (\boldsymbol{\beta}_1^T, \ldots, \boldsymbol{\beta}_p^T)^T$ is a $(Bp \times 1)$ vector, $\boldsymbol{\beta}_j = (\beta_j^{[1]}, \ldots, \beta_j^{[B]})^T$. The weak learner is essentially the same as in the previous section. Then, the updates of $\hat{\boldsymbol{\beta}}$ are introduced through $\hat{\boldsymbol{\beta}}_{(m)} = \hat{\boldsymbol{\beta}}_{(m-1)} + (\mathbf{0}^T, \ldots, \hat{\boldsymbol{\beta}}_{\hat{s}_m}^T, \ldots, \mathbf{0}^T)^T$, where $1 \leq \hat{s}_m \leq p$ denotes the fitted component at the mth step and $\mathbf{0}$ denotes zero vectors that supplement the second addend to a conformable argument. That leads to the well-known structure

$$\mathbf{f}_m = \mathbf{Z}\hat{\boldsymbol{\beta}}_{(m-1)} + \mathbf{Z}_{\hat{s}_m}\hat{\boldsymbol{\beta}}_{\hat{s}_m} = \mathbf{Z}\hat{\boldsymbol{\beta}}_{(m)} = \mathbf{f}_{m-1} + \mathbf{h}_m.$$

To stop the boosting iterations one could profit again from the sophisticated hat matrix at step k, $\mathcal{H}_{(k)}$ and particularly from its ability to express the model complexity by the effective degrees of freedom stuck in its trace. The hat matrix has the form

$$\mathcal{H}^{(\hat{s}_m)} = \sum_{j=0}^{m} \mathcal{H}^{(\hat{s}_j)} \prod_{i=0}^{j-1} (I - \mathcal{H}^{(\hat{s}_i)}) \tag{3.22}$$

(see the derivation in Appendix B). Thus, in the special case of Gaussian response, additive learner and L_2-loss function, the hat matrices (3.16) and (3.22) coincide.

3.4 Multivariate Boosting

Allowing high dimensionality for the response in a boosting strategy has been considered, up to my knowledge, only in Lutz and Bühlmann (2006) . They present theoretical treatment of the multivariate boosting and also provide empirical evidence that the multivariate approach outperforms individual estimations in several cases. Their technique strongly resembles the $L2$ Boosting scheme from Section 3.2, hence only the most distinctive features are outlined in the sequel. It should be noted, that this more advanced way of boosting was still not implemented in the standard add-on package `mboost` (Hothorn et al., 2008) at the time of writing this thesis.

A multivariate linear regression model with n observations is considered as follows:

$$\boldsymbol{Y} = \mathbf{X}\boldsymbol{B} + \boldsymbol{E} \tag{3.23}$$

with $\boldsymbol{Y} \in \mathbb{R}^{n\times q}$, $\mathbf{X} \in \mathbb{R}^{n\times p}$, $\boldsymbol{B} \in \mathbb{R}^{p\times q}$ and $\boldsymbol{E} \in \mathbb{R}^{n\times q}$. The increase of dimensionality demands a richer nomenclature. Therefore $\mathbf{y}_{(i)}$ denotes the response at the ith sample point, i.e. a q-dimensional row-vector. With $\mathbf{y}_j$ is indicated the jth response variable, i.e. a n-dimensional column-vector. For the error matrix is assumed $\mathrm{E}(\boldsymbol{e}_j) = \mathbf{0}$, $\mathrm{cov}(e_{(k)}, e_{(l)}) = \mathbf{0}$, for $k \neq l$, that is the sample points are independent, $\mathrm{cov}(e_i) = \boldsymbol{\Sigma}$. Additionally it is assumed that all covariates are centered to have zero mean, so no intercepts are worrying. The corresponding loss function is then

$$L(\boldsymbol{B}) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{y}_{(i)}^T - \mathbf{x}_{(i)}^T\boldsymbol{B})\boldsymbol{\Gamma}^{-1}(\mathbf{y}_{(i)}^T - \mathbf{x}_{(i)}^T\boldsymbol{B})^T \tag{3.24}$$

where $\boldsymbol{\Gamma}$ is an estimate of the usually unknown covariance matrix $\boldsymbol{\Sigma}$. The well known componentwise procedure is affected by the new dimensionality in the pseudo response as well. It consistently follows the dimensionality of the initial response and is termeed with $\boldsymbol{R} \in \mathbb{R}^{n\times q}$. The componentwise learner selects again only that component which reduces the loss function most:

Multivariate linear learner

$$\begin{aligned}
\boldsymbol{H}(\mathbf{x}_{\hat{s}}) &= \mathbf{x}_{\hat{s}}\hat{\beta}_{\hat{s},\hat{t}} \\
\hat{\beta}_{jk} &= \frac{\sum_{v=1}^{q}\boldsymbol{R}_v^T\mathbf{x}_j\boldsymbol{\Gamma}_{vk}^{-1}}{\mathbf{x}_j^T\mathbf{x}_j\boldsymbol{\Gamma}_{\boldsymbol{kk}}^{-1}} \\
(\hat{s},\hat{t}) &= \arg\max_{1\leq j\leq p, 1\leq k\leq q}\frac{\left(\sum_{v=1}^{q}\boldsymbol{R}_v^T\mathbf{x}_j\boldsymbol{\Gamma}_{vk}^{-1}\right)^2}{\mathbf{x}_j^T\mathbf{x}_j\boldsymbol{\Gamma}_{kk}^{-1}}
\end{aligned} \tag{3.25}$$

From (3.25) we see the impact of the multivariate structure in $\hat{\beta}_{jk}$, which is not influenced only by the kth response but also by other response-components via $\mathbf{\Gamma}^{-1}$ and their correlation with the jth predictor $\mathbf{x}_j$. The other key definition is the hat matrix, that maps the single components to the response. That is

$$\mathcal{H}^{(jk)} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathcal{H}^j \frac{\Gamma^{-1}_{k1}}{\Gamma^{-1}_{kk}} & \mathcal{H}^j \frac{\Gamma^{-1}_{k2}}{\Gamma^{-1}_{kk}} & \dots & \mathcal{H}^j \frac{\Gamma^{-1}_{kg}}{\Gamma^{-1}_{kk}} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \end{pmatrix} \tag{3.26}$$

where $\mathcal{H}^j = \mathbf{x}_j \mathbf{x}_j^T / \mathbf{x}_j^T \mathbf{x}_j$ is the hat matrix of the univariate componentwise linear learner using the jth predictor. Then the approximate hat matrix of the boosting at step m is

$$\boldsymbol{K}_m = I - (I - \nu \mathcal{H}^{(\hat{s}_m \hat{t}_m)})(I - \nu \mathcal{H}^{(\hat{s}_{m-1} \hat{t}_{m-1})}) \dots (I - \nu \mathcal{H}^{(\hat{s}_1 \hat{t}_1)}). \tag{3.27}$$

Lutz and Bühlmann (2006) also provide the computational complexity of the hat matrix $O(n^2 p + n^3 q^2 m)$ and conclude that such computations are not feasible for large n or q. Finally, the stopping criterion should also be conformable with higher dimensions, which leads to

$$\mathrm{AIC}_c(m) = \log(|\hat{\mathbf{\Sigma}}(m)|) + \frac{q(n + \mathrm{trace}(\boldsymbol{K}_m)/q)}{n - \mathrm{trace}(\boldsymbol{K}_m)/q - q - 1}$$

where $\hat{\mathbf{\Sigma}}(m) = n^{-1} \sum_{i=1}^{n} (\boldsymbol{R}_{(i)} \boldsymbol{R}_{(i)}^T)$ and the number of boosting operations is estimated via

$$\hat{M} = \arg \min_{0 \le m \le M} AIC_c(m)$$

with a pre-chosen, sufficiently large value of M.

Alternatively, one could approach the multivariate structure by *row-boosting*. The concept of row-boosting is to update a whole row of $\boldsymbol{B}$, instead of a single entry. This strategy is supported by the assumption that a single covariate could influence all response-components. The variable, which contributes to the multivariate fit most, is updated at the corresponding step. The multivariate fit is determined via Wilk's Λ,

$$\Lambda = \frac{|(\mathbf{Y} - \mathbf{X}\hat{\boldsymbol{B}})^T (\mathbf{Y} - \mathbf{X}\hat{\boldsymbol{B}})|}{|\mathbf{Y}^T \mathbf{Y}|}$$

where $|.|$ denotes the determinant of a matrix.

Lutz and Bühlmann (2006) showed with simulated data that multivariate boosting performs well, particularly in those situations, where the predictor dimension or the response dimension is large relative to the sample size. Row-boosting does not seem to outperform multivariate boosting, except of the situations with row-complete $\boldsymbol{B}$. In case of correlated errors, multivariate boosting is clearly superior to the individual L_2 Boosting and at least as good with uncorrelated errors. Apparently, on real data none of the boosting techniques proves to be the overall best method. For further details see the cited paper.

4 Time Series Models

In this section we will outline some basic concepts of time series models. In Section 4.1 we will discuss the general ideas behind univariate, autoregressive time series modelling in terms of stationarity conditions, parameter estimation and order selection. For a substantially broader discussion on times series, see Hamilton (1994), which is one of the most frequently quoted textbooks on the topic. Due to the wide application and frequent use of the simple autoregressive model, we will use it as a benchmark in the application part to follow (Section 6). In Section 4.2 we will introduce the relevant modelling techniques for vector autoregressions. Multivariate time series are considered in greater depth by Lütkepohl (1991; 2006). In the last Section 4.3 we will outline some strategies for modelling of nonlinear time series. Summaries of the common nonlinear models are given by Priestley (1980), Tong (1993) and Tsay (2005) among others.

4.1 Univariate Time Series

4.1.1 Stationarity

A *stochastic process* could be treated as a function of one or several deterministic arguments ("inputs") whose values ("outputs") are non-deterministic, i.e. random values to which a probability distribution is assigned. This implies that the future evolution of the process is not clearly determined, even when an equal starting point is ensured. However, some paths are more probable than others.

In the simplest possible case, a stochastic process amounts to a sequence of random variables, which is suitably termed *time series*. Each time series observation is assumed to be generated by a different member of the stochastic process. Suppose we have observed a sample size T of some variable y:

$$\{y_1, y_2, \ldots, y_T\}$$

we denote the stochastic process with $\{y_t\}_{t \in T}$. Note that we will follow the common practice to denote the random variables and the corresponding observations with the same symbol. It would be the context to suggest whether the symbol y_t refers to an observed value or a random variable. Further on, the *covariance* between two variables y_t and y_k is defined as

$$\operatorname{Cov}(y_t, y_k) = \operatorname{E}[(y_t - \mu_t)(y_k - \mu_k)] = \gamma_{tk} \tag{4.1}$$

where $\mu_t = \operatorname{E}(y_t)$ is the expectation, also called *unconditional mean*, of y_t. Note that provided $t - k = h$, (4.1) could be described as the covariance between y_t and

its own lagged value y_{t-h}, hence (4.1) is also referred to as *autocovariance* of y_t.

Time series analysis is based, indeed, on *stationarity*. A stochastic process $\{y_t\}_{t\in T}$ is called *(weakly) stationary* if neither the mean μ_t, nor the covariances γ_{tk} depend on the time index t. That is

$$\begin{aligned} \mathrm{E}\,(y_t) &= \mu \qquad \text{for all } t, \quad \mu < \infty \\ \mathrm{E}\,(y_t-\mu)(y_{t-h}-\mu) &= \gamma_h \qquad \text{for any } t \text{ and any } h \text{ such that } t-h \in T. \end{aligned} \tag{4.2}$$

Due to the practical usefulness and frequent usage of weakly stationary processes we suppress "weakly" and call them just *stationary*. So, by providing that the first and the second moment are time-invariant, we mean that a time series, generated by a stationary process must fluctuate around constant term μ with a constant variance $\sigma^2 = \gamma_0$. Moreover, the covariances between any two random variables depend on the distance between them, $h = t - k$, and not on the realisation dates t and k themselves. The normalized counterpart of γ_h is called *autocorrelation* and is denoted with $\rho_h = \gamma_h/\gamma_0$. Clearly, with $h = 0$, $\rho_h = 1$. Additionally, there is a more restrictive concept of *strict stationarity*, which requires even the joint distributions of $(y_t, y_{t-1}, \ldots, y_{t-h})$ to be unaffected by t. Strict stationarity has barely any practical application and is regarded as a rather theoretical construct. However, if the times series y_t is normally distributed, then weak stationarity is equivalent to strict stationarity.

In practice the expectation and the autocovariance are estimated by their sample counterparts, namely the sample mean

$$\hat{\mu} = \bar{y} = \frac{1}{T}\sum_{t=1}^{T} y_t$$

and the sample autocovariance

$$\hat{\gamma}_h = \frac{1}{T}\sum_{t=h}^{T}(y_t - \bar{y})(y_{t-h} - \bar{y}), \quad \text{for} \quad h = 0, 1, 2, \ldots.$$

Note that even though only $T-h$ observations are used to estimate $\hat{\gamma}_h$, the denominator is T rather than $T-h$. Thus, for large h, the estimates are shrunken towards zero.

4.1.2 Autoregressive Processes

A stochastic process $\{y_t\}_{t\in T}$ is said to be *autoregressive* of order p, if the past p values of y_t jointly determine the conditional expectation of y_t, given the past data.

This is

$$y_t = a_0 + a_1\, y_{t-1} + a_2\, y_{t-2} + \ldots + a_p\, y_{t-p} + \varepsilon_t \tag{4.3}$$

where ε_t is assumed to have mean zero and a constant variance σ^2 and is referred to as *white noise*.

A very convenient alternative for expressing system of equations like (4.3) is provided by the *lag operator* L. It shifts the time index of a times series variable backward by one unit of time, that is $Ly_t = y_{t-1}$. Applying the lag operator twice increases the shift back in time, such as

$$L^2 y_t = L(Ly_t) = L(y_{t-1}) = y_{t-2}.$$

It can be raised to arbitrary integer power p, $L^p = y_{t-p}$, providing this way an equivalent expression of (4.3) through

$$(1 - a_1 L - a_2 L^2 - \ldots - a_p L^p) y_t = a_0 + \varepsilon_t. \tag{4.4}$$

There is a detailed discussion in Hamilton (1994, p. 26-27) about the useful algebraic properties of the lag operator, such as commutativity with the multiplication operator

$$L(c\, y_t) = c\, L y_t$$

or distributivity over the addition operator

$$L(y_t + z_t) = L\, y_t + L\, z_k.$$

Through the application of a lag operator we can easily examine the stationarity. This is done via the so called *characteristic equation* of the AR(p) process:

$$1 - a_1 \lambda - a_2 \lambda^2 - \ldots - a_p \lambda^p = 0, \tag{4.5}$$

whose roots determine the dynamics of the whole stochastic process. The stationarity condition of the AR(p) process is fulfilled, if the absolute values of its *characteristic roots* exceed one, otherwise it is unstable. According to the common terminology, the characteristic roots are said to *lie outside the unit circle*. In should be noted that the nomenclature is not always consistent. Sometimes, characteristic roots indicate the inverses of the solutions of (4.5). Consequently, they are accordingly claimed to lie *inside* the unit circle in order to satisfy the necessary condition for stationarity.[5] Nevertheless, we restrain from this convention.

[5] Characteristic roots should be clearly distinguished from the eigenvalues of the so called *state space* representation of an AR(p) process, see Hamilton (1994, Chapters 1,13).

Suppose we have a simple example of an AR(2) process. The characteristic equation is

$$1 - a_1\lambda - a_2\lambda^2 = 0, \tag{4.6}$$

and the characteristic roots lie outside the unit circle, if

$$\lambda_{1,2} = \frac{a_1 \pm \sqrt{a_1^2 + 4a_2}}{2a_2}, \quad |\lambda_{1,2}| > 1.$$

It is worth mentioning that in situations, in which the characteristic roots are complex numbers,[6] the dynamic behaviour of the process is characterized by decaying sinusoids, instead of exponential decays, as with real valued solutions. For the special case, in which the characteristic equation (4.5) has a unit root, i.e. $\lambda_i = 1$, the process is said to be *unit root* process, or an *integrated* process, denoted by $I(1)$. Having unit roots, the original AR(p) could be factored as

$$(1 - a_1L - a_2L^2 - \ldots - a_pL^{p-1})\Delta y_t = a_0 + \varepsilon,$$

where Δ denotes the difference operator: $\Delta y_t = y_t - y_{t-1} = (1 - L)y_t$. Such transformation is commonly used in practice because we obtain an AR($p-1$) model of the $\{\Delta y_t\}_{t\in T-1}$ process.

4.1.3 Parameter Estimation

In practice we usually have to estimate the parameters a_i. Throughout this subsection we will assume that the order of the process is known as p. Later on we will show a strategy to estimate the order of the process as well. For now, assuming the stationarity is preliminarily satisfied, one could use very useful relations between the autocovariances of the process. Multiplying both sides of (4.3) by y_{t-j} and taking the expectations leads to

$$\mathrm{E}\,(\tilde{y}_t\tilde{y}_{t-j}) = a_1\,\mathrm{E}\,(\tilde{y}_{t-1}\tilde{y}_{t-j}) + \ldots + a_p\,\mathrm{E}\,(\tilde{y}_{t-p}\tilde{y}_{t-j}) + \mathrm{E}\,(\varepsilon_t\tilde{y}_{t-j}) \tag{4.7}$$

where $\tilde{y}_t = y_t - \mu$. Then we have

$$\gamma_j = a_1\,\gamma_{j-1} + a_2\,\gamma_{j-2} + \ldots + a_p\,\gamma_{j-p}, \quad \text{for } j = 1, 2, \ldots\, p. \tag{4.8}$$

Dividing both sides of (4.8) by γ_0 produces the famous *Yule-Walker equations*:

$$\rho_j = a_1\,\rho_{j-1} + a_2\,\rho_{j-2} + \ldots + a_p\,\rho_{j-p}. \tag{4.9}$$

[6]Note that the modulus of a complex number $\lambda = a + i\,b$ is defined by $|\lambda| = \sqrt{a^2 + b^2}$. See Hamilton (1994), p. 14-18 for a broader discussion on the effect of complex numbers to the system dynamics.

One can employ the fact that $\rho_1 = \rho_{-1}$ and rewrite (4.9) in a system of equations

$$\begin{aligned}
\rho_1 &= a_1\,\rho_0 + a_2\,\rho_1 + a_3\,\rho_3 + \ldots + a_{p-1}\rho_{p-2} + a_p\,\rho_{p-1} \\
\rho_2 &= a_1\,\rho_1 + a_2\,\rho_0 + a_3\,\rho_1 + \ldots + a_{p-1}\rho_{p-3} + a_p\,\rho_{p-2} \\
&\vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
\rho_{p-1} &= a_1\,\rho_{p-2} + a_2\,\rho_{p-3} + a_3\,\rho_{p-4} + \ldots + a_{p-1}\rho_0 + a_p\,\rho_1 \\
\rho_p &= a_1\,\rho_{p-1} + a_2\,\rho_{p-2} + a_3\,\rho_{p-3} + \ldots + a_{p-1}\rho_1 + a_p\,\rho_0
\end{aligned}$$

which is equivalent to

$$\underbrace{\begin{pmatrix} \rho_1 \\ \rho_2 \\ \vdots \\ \rho_{p-1} \\ \rho_p \end{pmatrix}}_{\boldsymbol{\rho}} = \underbrace{\begin{pmatrix} 1 & \rho_1 & \rho_2 & \ldots & \rho_{p-1} \\ \rho_1 & 1 & \rho_1 & \ldots & \rho_{p-2} \\ \vdots & & & & \vdots \\ \rho_{p-2} & \rho_{p-3} & \rho_{p-4} & \ldots & \rho_1 \\ \rho_{p-1} & \rho_{p-2} & \rho_{p-3} & \ldots & 1 \end{pmatrix}}_{\boldsymbol{\Phi}} \underbrace{\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_{p-1} \\ a_p \end{pmatrix}}_{\boldsymbol{a}}.$$

Finally, the estimates are found through

$$\hat{\boldsymbol{a}} = \boldsymbol{\Phi}^{-1}\boldsymbol{\rho}. \tag{4.10}$$

Another particularly easy estimation of an AR process of order p can be done via ordinary least squares (OLS). This model is in the same form as the well-known simple linear regression model in which y_t is the response and the $y_{t-1}, \ldots, y_{t-p}$ are the explanatory variables. Hence, the resulting over-determined system is

$$\underbrace{\begin{pmatrix} y_{p+1} \\ y_{p+2} \\ \vdots \\ y_T \end{pmatrix}}_{\mathbf{y}} = \underbrace{\begin{pmatrix} y_p & y_{p-1} & \ldots & y_1 \\ y_{p+1} & y_p & \ldots & y_2 \\ \vdots & & & \vdots \\ y_{T-1} & y_{T-2} & \ldots & y_{T-p} \end{pmatrix}}_{\boldsymbol{X}} \underbrace{\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{pmatrix}}_{\boldsymbol{a}}. \tag{4.11}$$

As usual, the estimator is $\hat{\boldsymbol{a}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$.

There are even more estimation techniques such as the Burg estimation method or Maximum Likelihood Estimation which, indeed, all provide very similar results with increasing sample size (see Brockwell and Davis (1991) for further details).

4.1.4 Order Selection

In order to complete the exposition of the estimation process we still have to find the initially unknown order of the AR-model. We will explore this problem from

the perspective of the most common estimator, the OLS. The strategy consists of successive computations of an information criterion for different orders, $m = 1, \ldots, l$, where l is a preliminary specified positive integer. Consequently, the selected m is the one, which provides the best value according to this criterion. As shown in Lütkepohl and Krätzig (2004, p. 33), the information criterion is of the general form

$$\mathrm{Cr}(m) = \log \hat{\sigma}_\varepsilon^2(m) + c_T \, \varphi(m) \tag{4.12}$$

where $\hat{\sigma}_\varepsilon^2(m) = T^{-1} \sum_{t=1}^{T} \hat{\varepsilon}_t(m)$ is the error variance estimator based on the OLS residuals $\hat{\varepsilon}(m)$ from an m-ordered AR model, c_T is a sequence indexed by the sample size, and $\varphi(m)$ is a function that penalizes large AR orders. The use of an information criterion like (4.12) leads to a parsimonious time series model, as it not only rewards goodness-of-fit but includes a penalty term, that is an increasing function of the number of the estimated parameters. This penalty term thus discourages overfitting. In the general case $\varphi(m)$ denotes the order of the fitted process and c_T is a weighting factor that may depend on the sample size. Several modifications of the information criterion (4.12) exist, which differ mainly in the choice of the weighting factor.

The application of OLS actually rearranges the lagged values as explanatory variables, which leads to reduction of the response length by a factor of l. It is important to note that the sample size should be kept constant for all orders. Let us have an initial sample size of length $T + l$. In order to simplify the notation, one starts to count the response values from $-l+1$ to T, instead from 1 to $T+l$. In other words, we relabel the observations by defining "presample" realizations $y_{-l+1}, \ldots, y_0$. Such partitioning implies that the sample size of all regressions is T. Then, the order that minimizes the criterion is chosen as estimator $\hat{p}$ of the true order p.

Specifying $c_T = 2/T$ leads to the well-known criterion of Hirotugu Akaike (Akaike, 1973, 1974)

$$\mathrm{AIC}(m) = \log \hat{\sigma}_\varepsilon^2(m) + \frac{2}{T} m$$

which is frequently used in practice. Another specifications of c_T produce the criterion

$$\mathrm{HQ}(m) = \log \hat{\sigma}_\varepsilon^2(m) + \frac{2 \log \log T}{T} m \qquad \text{(Hannan and Quinn, 1979)}$$

or

$$\mathrm{SC}(m) = \log \hat{\sigma}_\varepsilon^2(m) + \frac{\log T}{T} m \quad \text{(Schwarz, 1978).}$$

It is straightforward to see that for moderate to large sample sizes the Schwarz Criterion (SC) is more likely to produce parsimonious model, compared to the AIC.

However, after denoting the selected orders by $\hat{p}(\text{AIC})$, $\hat{p}(\text{HQ})$ and $\hat{p}(\text{SC})$ the following relations hold even in small samples of fixed size $T \geq 16$:

$$\hat{p}(\text{SC}) \leq \hat{p}(\text{HQ}) \leq \hat{p}(\text{AIC})$$

(Lütkepohl 1991, Chapters 4 and 11; Lütkepohl and Krätzig 2004, p. 33).

Each of the aforementioned criteria suggests the inclusion of all $\hat{p}$ lags in the model, regardless of how significant they are. However, it may happen that some lags are not significant. To overcome this problem one can specify a threshold value for the t-ratios of the estimated coefficients. If one t-value is smaller that the threshold, say 2, then the corresponding coefficient is restricted to zero and the model is re-estimated.

4.2 Vector Autoregressive Model

Now we will introduce vector autoregressions, which are particularly convenient for estimation and forecasting. They became very appealing for economic times series analysis since the seminal work of Sims (1980) has been introduced. We will consider the basic, stationary finite order vector autoregressive (VAR) model in particular. The VAR model suggests that every variable is a linear combination of its past observations and the past observations of supplemental variables. Additionally, the forecasting errors are uncorrelated for the different time periods. In practice such assumptions enjoy great popularity. We will examine the practical usefulness of the VAR model with real data in Section 6.

4.2.1 Stationary Vector Processes

In contrast to the univariate autoregressive time series, we assume $\mathbf{y}_t$ to be a q-dimensional random variable. Accordingly, the pth-order *vector autoregression*, denoted with VAR(p), is a vector generalization of (4.3):

$$\mathbf{y}_t = A_0 + A_1 \, \mathbf{y}_{t-1} + \ldots + A_p \, \mathbf{y}_{t-p} + \boldsymbol{\varepsilon}_t \tag{4.13}$$

with A_0 denoting a $(q \times 1)$ vector of constants and A_j a $(q \times q)$ matrix of autoregressive coefficients with $j = 1, 2, \ldots, p$. The $(q \times 1)$ error vector $\boldsymbol{\varepsilon}_t$ is affected by the new dimensionality in the following way:

$$\mathrm{E}\left(\boldsymbol{\varepsilon}_t\right) = \mathbf{0}$$

and

$$\mathrm{E}\left(\boldsymbol{\varepsilon}_t \boldsymbol{\varepsilon}_k'\right) = \begin{cases} \boldsymbol{\Sigma} & \text{for } t = k, \\ \mathbf{0} & \text{otherwise} \end{cases} \tag{4.14}$$

where $\mathbf{\Sigma}$ denotes a $(q \times q)$ symmetric positive definite matrix and $\mathbf{0}$ denotes a $(q \times 1)$ zero vector. Note that every row in the vector autoregression system (4.13) represents a scalar variable which is regressed on a constant, its own p past-values and the additional p past-values of the others $q-1$ variables, e.g. the ith row is

$$y_{it} = a_i^{(0)} + \underbrace{a_{i1}^{(1)} y_{1,t-1} + \ldots + a_{iq}^{(1)} y_{q,t-1}}_{\text{First lag}} + \ldots + \underbrace{a_{i1}^{(p)} y_{1,t-p} + \ldots + a_{iq}^{(p)} y_{q,t-p}}_{p\text{th lag}} + \varepsilon_{it}.$$

where $a_i^{(0)}$ denotes the ith element of matrix A_0, $a_{ij}^{(k)}$ the row i, column j element of matrix A_k. The regressors in all equations are the same which turns out to be very helpful for the application of OLS estimators later on, i.e. we will see that the generalized least squares estimator coincide with the ordinary least squares estimator.

Again, stationarity guarantees that the "essential" properties of the times series remain constant over time. A vector process $\{\mathbf{y}_t\}_{t \in T}$ is said to be weakly stationary[7] if its first and second moments ($\mathrm{E}(\mathbf{y}_t)$ and $\mathrm{E}(\mathbf{y}_t \mathbf{y}_{t-h})$) are time invariant. This means that

$$\begin{aligned} \mathrm{E}(\mathbf{y}_t) &= \boldsymbol{\mu}, \qquad \|\boldsymbol{\mu}\| < \infty \\ \mathrm{E}[(\mathbf{y}_t - \boldsymbol{\mu})(\mathbf{y}_{t-h} - \boldsymbol{\mu})] &= \mathbf{\Gamma}_h, \qquad \|\mathbf{\Gamma}\| < \infty. \end{aligned}$$

where $\boldsymbol{\mu}$ and $\mathbf{\Gamma}_h$ do not depend on t. After application of the lag operator L, (4.13) can be rewritten in

$$(I_n - A_1 L - A_2 L^2 - \ldots - A_p L^p)\mathbf{y}_t = A_0 + \boldsymbol{\varepsilon}_t.$$

and the stationarity condition holds if all values of λ satisfying

$$|I_n - A_1 \lambda - A_2 \lambda^2 - \ldots - A_p \lambda^p| = 0$$

lie outside the unit circle.

4.2.2 Estimation of Vector Autoregressive Processes

The estimation principles of univariate and multivariate autoregressive processes are closely related. Again, we have to go through order selection, parameter estimation and possibly parameter restrictions, in order to estimate a reasonable model. Therefore, in the following we will summarize the most distinctive features of the VAR model, drawing partly up on Lütkepohl (2006, Chapters 3 and 4).

[7] Also called covariance-stationary, wide-sense stationary or stationary

Before we start the estimation process, it should be noted that there are different possibilities for estimating a VAR model. Many of them are multivariate generalizations of the methods, met in the univariate case such as Least Squares Estimation, Yule-Walker Estimation or Maximum Likelihood Estimation. Since discussion on all methods would go beyond the scope of the present work, we will discuss the first method, which is most commonly used in practice: the Least Squares Estimation.

In order to estimate the elements in $A_0, A_1, \ldots, A_p$ and $\mathbf{\Sigma}$, generated by the q-dimensional VAR(p) process (4.13), it is assumed that $T + p$ time series observations are available. As previously discussed, such assumption facilitates the notation by partitioning the data into p "presample" observations, that is $\mathbf{y}_{p-1}, \ldots, \mathbf{y}_0$, and remaining observations $\mathbf{y}_1, \ldots, \mathbf{y}_T$. Then we introduce the following notation:

$$
\begin{aligned}
\mathbf{Y} &= [\mathbf{y}_1, \ldots, \mathbf{y}_{\boldsymbol{T}}] \qquad (q \times T) \\
\boldsymbol{B} &= [A_0, A_1, \ldots, A_p] \quad (q \times (qp+1)) \\
\boldsymbol{Z}_t &= \begin{bmatrix} 1 \\ \mathbf{y}_t \\ \vdots \\ \mathbf{y}_{t-p+1} \end{bmatrix} \qquad ((qp+1) \times 1) \\
\boldsymbol{Z} &= [\boldsymbol{Z}_0, \ldots, \boldsymbol{Z}_{T-1}] \qquad ((qp+1) \times T) \\
\boldsymbol{\mathcal{E}} &= [\boldsymbol{\varepsilon}_1, \boldsymbol{\varepsilon}_2, \ldots, \boldsymbol{\varepsilon}_T] \quad (q \times T) \\
\mathbf{y} &= \mathrm{vec}(\mathbf{Y}) \qquad (qT \times 1) \\
\boldsymbol{\varepsilon} &= \mathrm{vec}(\boldsymbol{\mathcal{E}}) \qquad (qT \times 1) \\
\boldsymbol{\beta} &= \mathrm{vec}(\boldsymbol{B}) \qquad (q^2 p \times 1) \\
\boldsymbol{b} &= \mathrm{vec}(\boldsymbol{B}') \qquad (q^2 p \times 1)
\end{aligned}
$$

where vec is the column stacking operator as defined in Appendix B.5. The initial T equations (4.13) can now be written compactly in a matrix form as

$$\mathbf{Y} = \boldsymbol{B}\boldsymbol{Z} + \boldsymbol{\mathcal{E}}. \tag{4.15}$$

Then, appying the vectorization rules (1) and (3) from Appendix B.5 to (4.15), we come up with

$$\mathrm{vec}(\mathbf{Y}) = (\boldsymbol{Z}' \otimes I_q)\mathrm{vec}(\boldsymbol{B}) + \mathrm{vec}(\boldsymbol{\mathcal{E}}) \tag{4.16}$$

where $\otimes$ denotes the Kronecker Product (Eves, 1980). The latter could be equivalently represented as:

$$\mathbf{y} = (\boldsymbol{Z}' \otimes I_q)\boldsymbol{\beta} + \boldsymbol{\varepsilon}. \tag{4.17}$$

Note that the covariance matrix of $\boldsymbol{\varepsilon}$ is

$$\mathbf{\Sigma}_\varepsilon = I_T \otimes \mathbf{\Sigma}. \tag{4.18}$$

Since $\boldsymbol{\Sigma}_\varepsilon$ is not a diagonal matrix we employ the *generalized least squares estimator* (GLS). That means that an estimation for $\boldsymbol{\beta}$ is obtained via minimizing

$$S(\boldsymbol{\beta}) = \boldsymbol{\varepsilon}'\boldsymbol{\Sigma}_\varepsilon^{-1}\boldsymbol{\varepsilon}$$

which in turn produces

$$\hat{\boldsymbol{\beta}} = ((\boldsymbol{Z}\boldsymbol{Z}')^{-1}\boldsymbol{Z} \otimes I_q)\mathbf{y}. \tag{4.19}$$

(see Appendix B.3 for derivation of (4.19)). It is worth noting that the GLS-Estimator $\hat{\boldsymbol{\beta}}$ is independent from $\boldsymbol{\Sigma}_\varepsilon$. So, if we examine the trivial OLS-Estimator, obtained simply by minimizing

$$\tilde{S}(\boldsymbol{\beta}) = \boldsymbol{\varepsilon}'\boldsymbol{\varepsilon}$$

we have exactly the same estimation of $\boldsymbol{\beta}$, as in (4.19) (see derivation in Appendix B.4). This result is attributable to Zellner (1962), who showed that GLS and OLS estimation in a multiple equation model coincide if the regressors in all equations are the same. Further on, in optimization problems, the definiteness of the Hessian matrix determines the quality of an extremal value. In this case, the Hessian matrix of $S(\boldsymbol{\beta})$

$$\frac{\partial^2 S(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}'} = 2(\boldsymbol{Z}\boldsymbol{Z}' \otimes \boldsymbol{\Sigma}^{-1}) \tag{4.20}$$

is positive definite which guarantees that $\hat{\boldsymbol{\beta}}$ does indeed minimize $S(\boldsymbol{\beta})$. More precisely, $\hat{\boldsymbol{\beta}}$ is a strict local minimum of S.

Alternatively, one can use the OLS estimator of (4.15) in order to derive estimates via

$$\hat{\boldsymbol{B}} = \mathbf{Y}\boldsymbol{Z}'(\boldsymbol{Z}\boldsymbol{Z}')^{-1}$$

or, in vector form,

$$\hat{b} = \mathrm{vec}(\hat{\boldsymbol{B}}) = (I_q \otimes (\boldsymbol{Z}\boldsymbol{Z}')^{-1}\boldsymbol{Z})\mathrm{vec}(\mathbf{Y}').$$

Now we provide four conditions:

(1) $\mathrm{E}(\boldsymbol{\varepsilon}_t) = \mathbf{0}$

(2) $\mathrm{E}(\boldsymbol{\varepsilon}_t\boldsymbol{\varepsilon}_t') = \boldsymbol{\Sigma}$, $\boldsymbol{\Sigma}$ nonsingular

(3) $\mathrm{E}(\boldsymbol{\varepsilon}_t\boldsymbol{\varepsilon}_k') = \mathbf{0}$ for $k \neq t$

(4) $\mathrm{E}\,|\varepsilon_{it}\varepsilon_{jt}\varepsilon_{kt}\varepsilon_{mt}| \leq c$, with c being positive constant, i.e. fourth moment exists

which jointly define $\boldsymbol{\varepsilon}_t$ as being a *standard white noise process*. Provided standard white noise process, it is assured that the estimator converges in probability to the true value, i.e. it is *consistent*, and is *asymptotically normal* (see Lütkepohl 2006, p. 74, Proposition 3.1):

- $\operatorname{plim} \hat{\boldsymbol{B}} = \boldsymbol{B}$
- $\sqrt{T}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}) = \sqrt{T}\operatorname{vec}(\hat{\boldsymbol{B}} - \boldsymbol{B}) \xrightarrow{d} \mathrm{N}(\mathbf{0}, \boldsymbol{\Gamma}^{-1} \otimes \boldsymbol{\Sigma})$

where $\boldsymbol{\Gamma} = \operatorname{plim} \boldsymbol{Z}\boldsymbol{Z}'/T$ and $\xrightarrow{d}$ denotes convergence in distribution. The second asymptotical property will be considered in the discussion about the proper parameter constraints that follows.

Up to this moment we have assumed that the real order of the VAR process is known. Apparently, we need procedures for choosing an adequate VAR order, p, in practice. In order to make significant estimations for larger response dimensions, the number of the sample size should (exponentially) increase. This phenomenon is commonly referred to as *the curse of the dimensionality* (Bellman, 1961). Due to insufficient degrees of freedom, the model parameters are then imprecisely estimated, thus yielding large standard errors and high estimation uncertainty. In the past, a common practice to handle this problem was simply to specify a model with a shorter lag length. Sims (1980) motivated a criticism against this methodology, stating that the economic models often suffer from "incredible zero restrictions". Therefore, it may be more appropriate to come up with schemes, reducing the number of free parameters without shortening lag lengths.

A straightforward strategy, which includes t-, χ^2- or F-tests for inference regarding the parameters, may be inappropriate too (see Toda and Phillips, 1993). In the presence of unit roots, i.e. presence of integrated or cointegrated variables, this strategy raises another problem concerning the convergence rate of the parameters. As the second asymptotical property shows, the parameters converge with a rate of $T^{1/2}$. But, if there are integrated or cointegrated variables, some estimated coefficients or linear combinations of coefficients converge with a faster rate than $T^{1/2}$, which makes the proper interpretation of the t-ratios unclear. Nevertheless, the harmful influence of unit roots could be relaxed to some extent. As shown by Toda and Yamamoto (1995) and Dolado and Lütkepohl (1996), if all variables are I(1) or I(0), the usual tests have their standard asymptotic properties. In other words, t-ratios have their usual standard normal distributions and are suitable statistics for testing that a single coefficient is zero.

Finally, we determine the order of the autoregressive process via model selection

procedure. We employ a generalized version of (4.12) via

$$\mathrm{Cr}(m) = \log(\det(\tilde{\Sigma}_u(m))) + c_T\,\varphi(m), \tag{4.21}$$

where det(.) denotes the determinant and $\tilde{\Sigma}_u(m) = T^{-1}\sum_{t=1}^{T}\hat{\boldsymbol{\varepsilon}}\hat{\boldsymbol{\varepsilon}}'$ is the residual covariance matrix estimator for a model of order m, c_T is a sequence indexed by the sample size T and $\varphi(m)$ is a function which penalizes large VAR orders. The general strategy again is to fit VAR models of different orders $m = 0, \ldots, l$ and to choose an order estimator $\hat{p}$ which minimizes the preferred criterion. The three criteria from Section 4.1.4 are now generalized to

$$\mathrm{AIC}(m) = \log(\det(\tilde{\Sigma}_u(m))) + \frac{2}{T}\,m\,q^2,$$

$$\mathrm{HQ}(m) = \log(\det(\tilde{\Sigma}_u(m))) + \frac{2\log\log T}{T}\,m\,q^2$$

and

$$\mathrm{SC}(m) = \log(\det(\tilde{\Sigma}_u(m))) + \frac{\log T}{T}\,m\,q^2.$$

4.3 Nonlinear Autoregressive Models

Linear time series models are generally the starting point for modeling both stationary univariate and multivariate times series data. However, the practice shows that real world data quite often exhibits nonstationary behaviour, e.g. structural breaks, variance increases, changing lag order. When nonlinear dynamics are an objective, it is no longer sufficient to consider linear models. The literature offers a great amount of nonlinear modeling tools. For the sake of integrity, we will summarize the most common of them.

The first four models of our overview are developed in the spirit of nonlinear *parametric* models. Nonlinear parametric models have one substantial drawback, which is the reason for their varying performance. They require an a priori choice of parametric functions, which are *believed* to be appropriate in certain situations. This approach is used mainly in financial applications, when we have sufficient knowledge to prespecify the nonlinear structure between the covariates and the response. However, the appropriateness is usually hard to be justified. Consequently, these methods are not always capable to capture the relevant features. In this case, one has to choose an alternative nonlinear parametric model. Here are the options.

- The *Bilinear Model* is maybe the simplest nonlinear model. It is a natural extension of the simple autoregressive model (4.3). Bilinear models incorporate the class of linear models considered by Box and Jenkins (1976), namely

the integrated auto-regressive moving average (ARIMA) models. The ARIMA model is considered as the first-order Taylor expansion of the true, underlying function. Bilinear model employs the second-order Taylor expansion in order to improve the estimation. This model was introduced by Granger and Andersen (1978).

- *Self-Exciting Threshold AutoRegressive* (SETAR) model is another extension of the autoregressive model. It allows higher degree of flexibility in the model parameters through a regime switching behaviour. The model consists of k regimes, each considering different autoregressive parts. The major criticism of the SETAR model is based on the discontinuity of its mean function. Furthermore, the transition is determined by a particular lagged variable. Consequently, this suggests deterministic scheme of switching. Therefore, the model is justified only in cases in which nonlinearity is caused by declining or rising patterns in the stochastic process. The model was fully developed by Tong and Lim (1980).

- *Smooth Tansition AutoRegressive* (STAR) model has been proposed in response of the criticism of the SETAR model. It actually contains the two-regime SETAR model as a special case. It can be understood as two-regime SETAR model with smooth transition between the regimes, or as *continuum* of the regimes. Thus, the presence of transition function is the defining feature of this model. It was developed by Chan and Tong (1986). The method is implemented in the **J**ava-based **Mul**tiple **Ti**me series software `JMulti`, based on Lütkepohl and Krätzig (2004).

- The last and the most popular model in this first part of our overview is the *Markov Switching Autoregressive* (MSA) model. It uses probability switching with aperiodic transition between various states. More precisely, MSA controls the transition from one conditional mean function to another via hidden Markov chain. Thus, the MSA has the very appealing property of stochastic scheme of switching, i.e. it does not require the presence of a distinct pattern, explicitly followed by the process. MSA is considered in Hamilton (1989). There is a specialized implementation of MSA, namely **R**egression **A**nalysis of **T**ime **S**eries (RATS) software package.

In contrast to the *parametric* nonlinear models, when using *nonparametric* techniques, we are not restricted to a particular choice of parametric function classes. One principal strategy is to study the times series counterpart of the additive model (2.2), which is

$$y_t = f_1(y_{t-i_1}) + f_2(y_{t-i_2}) + \ldots + f_p(y_{t-i_p}) + \varepsilon_t \tag{4.22}$$

where i_j's are positive integers, f_j represent the essence of nonparametric models, namely smooth functions and a white noise term ε_t. Models like (4.22) are termed by Chen and Tsay (1993) *Nonlinear Additive AutoRegressive* (NAAR) models. When further (*exogenous*) variables are available, we suitably extend the model from (4.22) with more functions and call it NAARX. Thus, NAARX encompasses linear regressive models and many nonlinear models as special cases. Then we could employ different nonparametric strategies to select the significant covariates (or lags) and fit a reasonable model. Many of these strategies have one fundamental ingredient in common, namely basis expansions, which we have discussed in detail in Section 2. Therefore, in the remainder of this section we give a partial sketch of the key ideas underpinning common nonparametric algorithms, without being very exhaustive. The aim is to get familiar with the general principles, while a series of references point to the original sources for an in-depth exposition.

4.3.1 Spline Fitting with BIC

Huang and Yang (2004) recently introduced a study that gained much of an attention. Their method manages to demonstrate very appealing lag-selection properties for univariate nonlinear time series. It is fairly simple because, in fact, it represents an additive version of the *linear stepwise procedure*. The procedure proposes truncated splines or B-Splines as base expansions of the predictors. Note that the proposed base functions are not penalized. Instead, the study suggests a formula, which determines a quite small number of evenly spaced knots. The maximal lag length of the process is defined via the integer d, which is called *total* number of candidate variabes. Another index S_{max} indicates the maximal number of the variables (among the candidate ones) which are allowed in the model. Clearly, S_{max} should not be larger than d. The actual method is divided into three stages: *forward stage, backward stage* and *final selection*. The forward stage starts with a current model, which is the null model, i.e. $y_t = c + \varepsilon_t$, where c is a constant. Then, it adds to the current model one variable at a time, by choosing the one, which minimizes a modified version of the mean-squared error (MSE). The fitting is actually the least squares method, as described by the augmented linear model (2.5). The "best" variable is included into the current model. Then, we continue to add from the remaining variables one at a time, until we reach the maximal number S_{max} of candidate variables in the current model.

The backward stage starts with the model, resulted from the forward stage. It deletes from the current model one variable at a time in accordance to the MSE, excluding the "worst" ones. The backward stage continues in this fashion until no

variables remain in the current model. Both forward and backward stages identify a collection of "good" models. The final selection chooses from this collection the model, which provides the best performance according to a corrected version of the Schwarz Criterion (or BIC), discussed in 4.1.4. In terms of lag selection, the proposed method performs quite well with simulated time series. However, no results are provided, that concern the goodness-of-fit of the model. We will use some of the artificial times series, provided by Huang and Yang (2004) in the Section 5 and will shed light upon the goodness-of-fit as well.

4.3.2 Multivariate Adaptive Regression Splines

The base reference about *Multivariate Adaptive Regression Splines* (MARS) is Friedman (1991). A neat overview of the method is proposed by Hastie, Tibshirani, and Friedman (2001, Chapter 9) and an application of MARS in a time series context is provided by Lewis and Stevens (1991). Probably the first thing to be noted about MARS is that the term *multivariate* actually suggests a procedure which includes multivariate tensor-splines bases and does not assume a multidimensional response. It is considered as a generalization of the pioneering adaptive procedure for regression splines by Friedman and Silverman (1989), called TURBO. However, high dimensionality of the response is also considered by Stone, Hansen, Kooperberg, and Truong (1997) and is suitably termed POLYMARS. The second main thing about MARS is the term *adaptive*, which implies an optimizing procedure over the number and the location of the knots in an adaptive way. We have already mentioned this in Section 2 as an alternative of the penalizing concept.

MARS uses linear splines of the form $(x-\tau)_+$ and $(\tau-x)_+$, where

$$(x-\tau)_+ = \begin{cases} x-\tau & \text{if } x > \tau, \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad (\tau-x)_+ = \begin{cases} \tau-x & \text{if } x \leq \tau, \\ 0 & \text{otherwise} \end{cases}$$

which are nonzero linear functions as shown in the example of Figure 5. The starting point is the forming of such pairs (or *reflected pairs*) for every single observation, which means that we have $2np$ basis functions, where p is the number of the predictors and n is the sample size, and describe them as candidate functions, collected in a set $\mathcal{C}$. Note that these basis functions share the appealing property of B-Splines to operate locally: they are nonzero over a small part of the domain. The modelling strategy is based on the well known least squares estimation (2.5), applied in a stepwise manner to a hierarchically enlarging model. The enlargement is done via the reflected pairs or their products from $\mathcal{C}$. That means, that every observed predictor value is a candidate knot site. So, the knot selection implies an automatic variable

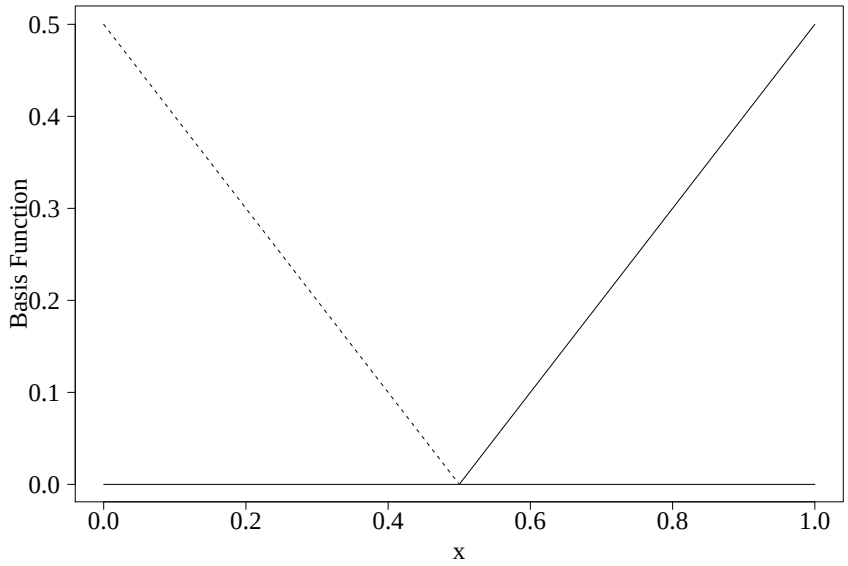


Figure 5: *An example of the basis functions* $(0.5 - x)_+$ *(broken line) and* $(x - 0.5)_+$ *(solid line) used by MARS.*

selection. This original addition procedure stops when the number of the terms in the current model reaches some preliminary specified limit.

At the end of the addition stage, one usually has an overfitted model, to which a backward deletion procedure should be applied. The deletions are performed according to a modified version of the GCV criterion. The essence of this modification consists in a tuneable *penalty* term, which charges a cost per basis function.

4.3.3 BRUTO

The last nonparametric model in our review is the BRUTO procedure (Hastie and Tibshirani, 1990, Chapter 9). Inspired by TURBO, BRUTO combines inputs selection with backfitting by using smoothing splines: once the selection process stops, the model is backfit. It was applied to time series by Chen and Tsay (1993). The BRUTO method can be thought of as optimizing of an approximation to the GCV criterion over all p smoothing parameters λ_j in a p-term additive model. It addresses another central concept for estimating additive models, namely *backfitting*. The underlying idea of backfitting is to estimate the individual functions iteratively, conditioned on the results of the other functions. Heuristically explained, this is an estimation of the smooth components of an additive model by iteratively smoothing the partial residuals from that model. The partial residuals relating to the jth

smooth term are the residuals, which result from subtracting all the current estimates from the response variable, except for the estimates of the jth component. Let us assume we have an additive model of the form

$$y_i = \alpha + \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i.$$

The partial residuals of the kth terms are: $\mathbf{y} - \hat{\alpha} - \hat{\mathbf{f}}_1 - \ldots - \hat{\mathbf{f}}_{k-1} - \hat{\mathbf{f}}_{k+1} \ldots - \hat{\mathbf{f}}_p$, where $\hat{\mathbf{f}}_j = (\hat{f}_j(x_{1j}), \ldots, \hat{f}_j(x_{nj}))^T$. Consequently, the basic backfitting algorithm is an iterative procedure as follows:

1. Initialize: $\hat{\alpha} = \bar{y}$ and $\hat{\mathbf{f}}_j = \mathbf{0}$ for $j = 1, \ldots, p$
2. Cycle: $j = 1, \ldots, p, 1, \ldots, p, \ldots$

   Calculate: $\boldsymbol{e}_j = \mathbf{y} - \hat{\alpha} - \sum_{k \neq j} \hat{\mathbf{f}}_k$

   Set: $\hat{\mathbf{f}}_j$ equal to the result of smoothing $\boldsymbol{e}_j$ with respect to x_j.
3. Repeat 2 until $\hat{\mathbf{f}}_j$ stop changing.

It is provided that the backfitting algorithm always converge. See Hastie and Tibshirani (1990), p. 90-91 for details concerning backfitting and p. 262 for the BRUTO algorithm.

5 Simulation Study

In this section we will investigate the performance of boosting an additive model in Monte Carlo simulations with six artificial, nonlinear, autoregressive time series. We will compare the outcomes of boosting to the outcomes, obtained through alternative nonparametric methods. Their performance will be considered in two categories: in terms of lag-selection (Section 5.2) and goodness-of-fit (Section 5.3). The dynamics of the simulated processes are shown in Table 2. All of them fulfil the stationary condition (4.2). Models NLAR1U1-NLAR1U2 have one lag and were used by Huang and Yang (2004). Besides, there are three models with two lags: NLAR1-NLAR3 which were originally used by Tschernig and Yang (2000). The last model NLAR4 has four lags and was used by Shafik and Tutz (2007). The present work makes a difference mostly thanks to the alternative assessment of the models' dynamics.

5.1 Implementation

All data analyses presented in this thesis have been carried out using the `R` system for statistical computation (R Development Core Team, 2008), version 2.6.2. In the following we will discuss some `R` extensions that have been useful for the application of alternative methods.

There are several implementations of boosting techniques, available as add-on packages for `R`. Package `mboost` (Hothorn et al., 2008) provides an implementation for fitting generalized linear models, as well as additive gradient based boosting, while package `GAMBoost` (Tutz and Binder, 2006; Binder, 2006) provides an implementation of the likelihood boosting approach, as described in Section 3.3. In our special case of Gaussian response with L_2-loss, both techniques for fitting an additive model coincide and are referred to as GAMBoost. Our simulations were carried out with `mboost`. For base procedure were used P-Splines, provided by the function

Model	Function
NLAR1U1	$y_t = -0.4(3 - y_{t-1}^2)/(1 + y_{t-1}^2) + 0.1\epsilon_t$
NLAR1U2	$y_t = 0.6(3 - (y_{t-2} - 0.5)^3)/(1 + (y_{t-2} - 0.5)^4) + 0.1\epsilon_t$
NLAR1	$y_t = -0.4(3 - y_{t-1}^2)/(1 + y_{t-1}^2) + 0.63(3 - (y_{t-2} - 0.5)^3)/(1 + (y_{t-2} - 0.5)^4) + 0.1\epsilon_t$
NLAR2	$y_t = (0.4 - 2\exp(-50y_{t-6}^2))y_{t-6} + (0.5 - 0.5\exp(-50y_{t-10}^2))y_{t-10} + 0.1\epsilon_t$
NLAR3	$y_t = (0.4 - 2\cos(40y_{t-6})\exp(-30y_{t-6}^2))y_{t-6} \quad + \quad (0.55 - 0.55\sin(40y_{t-10})\sin(40y_{t-10}))\exp(-10y_{t-10}^2) + 0.1\epsilon_t$
NLAR4	$y_t = 0.9((\pi/8)y_{t-4}) - 0.75\sin((\pi/8)y_{t-5}) + 0.52\sin((\pi/8)y_{t-6}) + 0.38\sin((\pi/8)y_{t-7}) + 0.1\epsilon_t$

Table 2: *Dynamics of six artificial time series.*

`gamboost()` with option for the base learner `bbs`. Additionally, the knots were set to 20, i.e. `knots = 20` and the degrees of freedom were set to 3.5, i.e. `degree = 3.5`. For all other options the default values were used.

Further on, we consider the method proposed by Huang and Yang (2004), described in Section 4.3.1, which uses spline fitting with BIC. Their novel approach was "manually" implemented (see Appendix D, function `stepwise()`), since it is currently not available as an extension package for `R` or in any other statistical software. It is labeled with the acronym HaY. The implementation was carried out via the package `mgcv` (Wood, 2006, 2007) with unpenalized cubic splines. The maximal number of candidate variables has been equalled to the maximal number of lags.

A classical candidate for additive fitting with component selection is the BRUTO algorithm. As mentioned before, it fits a model by adaptive backfitting using smoothing splines. An implementation of BRUTO could be found in package `mda` (Hornik et al., 2006), originally provided by Trevor Hastie and Robert Tibshirani and maintained by Kurt Hornik. The corresponding function `bruto()` has a tuning parameter `cost`, which specifies the cost per degree-of-freedom change. It was empirically investigated by Huang and Yang (2004), that a value of $\log(n)$ provides much better results than the default value of two, where n indicates the sample size. Therefore, in our application `cost` was set to $\log(n)$ too.

Another nonparametric alternative in data mining is MARS. We include MARS in the comparison as a powerful strategy to detect non-monotone relationships between the predictors and the covariates, which is particularly suitable for problems with many variables and possible interaction effects. Like BRUTO, it includes an automatic variable selection by identifying all "promising" variables, which makes it a "natural rival" of boosting. An implementation of MARS is available in package `mda` and the corresponding function is `mars()`. It has a tuning parameter, which charges a cost per basis function, denoted by `penalty`. This tuning parameter was also set to $\log(n)$.

In order to make results reproducible, the random number generator `set.seed()` has been fixed and the simulated time series stored locally. All models from Table 2 have been simulated 100 times with sizes $400 + N$, the first 400 values discarded and $N = p + T$, with $p = 10$ pre-sample values and $T = 50, 100, 200$ in-sample observations. In Section 4 we have argued that such partitioning of the time series values is convenient in order to ensure same sample size of T for each variable at a given period and to simplify the notation. As p suggests, the number of maximal

lags has been limited to ten. In the next section we will compare the performance of the different procedures in terms of lag selection.

5.2 Lag Selection

For each process, we have an index set s, consisting of the numbers of the true variables, e.g. for NLAR3, $s = \{6, 10\}$. Let $\hat{s}$ be a particular model estimation of s. The correctness of the estimation is quantified by the following rule: $\hat{s}$ is said to be *correct* if $\hat{s} = s$; $\hat{s}$ is an *overfit* if $\hat{s} \supset s$; and $\hat{s}$ is an *underfit* if $(\hat{s} \cap s) \subset s$. Note that $\hat{s}$ can be larger than s and still underfitting. In other words, underfit indicates that some significant variables have been erroneously omitted by the model, while overfit stays for inclusion of redundant variables in addition to the significant ones.

Table 3 contains a summary of the Monte Carlo simulations with all four fitting procedures. Each stochastic process and its corresponding in-sample length are presented horizontally. For each setup in the table the first, second and third columns present the numbers of underfit, correct and overfit outcomes over 100 simulation runs. For example, MARS at NLAR3 with $T = 50$ has identified the index set 34 times correctly, has neglected at least one of the significant lags 46 times and has added more lags in 20 cases.

As Table 3 promptly suggests, boosting of an additive model is likely to overfit most of the times. This tendency is especially noticeable in the cases with one significant lag only (NLAR1U1, NLAR1U2) and in NLAR1. Such performance of GAMBoost is in some sense expected. If in a single boosting step, some non-significant variable has been considered, that would be sufficient to add it to the estimated set $\hat{s}$. Although redundant variables have been erroneously selected, the corresponding function estimates can still be close to zero and therefore be interpreted as random errors. In the the next section we will explore whether such an influence is really considered as minimal or it has a substantial counterproductive impact.

Furthermore, boosting almost never missed significant components in the cases with more than one lag. On the contrary, the other methods are more likely to underfit larger models. This is evident for NLAR2 and NLAR3, and becomes especially noticeable for NLAR4. The last process is repeatedly underfitted by BRUTO, MARS and HaY, while GAMBoost encourages inclusion of more lags. Nevertheless, we should keep in mind that the mathematical properties of boosting for variable selection are still open questions.

Model	Length	GAMBoost			BRUTO			MARS			HaY		
NLAR1U1	50	0	0	100	0	29	71	0	73	27	0	98	2
	100	0	0	100	0	22	78	0	78	22	0	100	0
	200	0	0	100	0	27	73	0	61	39	0	100	0
NLAR1U2	50	0	0	100	0	0	100	0	73	27	0	32	68
	100	0	0	100	0	0	100	0	81	19	0	96	4
	200	0	0	100	0	0	100	0	71	29	0	100	0
NLAR1	50	0	0	100	44	16	40	0	46	54	4	93	3
	100	0	0	100	5	37	58	0	73	27	0	98	2
	200	0	0	100	0	57	43	0	65	35	0	97	3
NLAR2	50	1	0	99	98	2	0	64	27	9	99	1	0
	100	5	7	88	84	16	0	7	73	20	92	8	0
	200	0	7	93	0	92	0	0	83	17	88	12	0
NLAR3	50	1	0	99	44	47	9	46	34	20	68	32	0
	100	0	8	92	4	81	15	6	67	27	26	74	0
	200	0	12	88	0	93	7	0	81	19	1	99	0
NLAR4	50	15	4	81	100	0	0	100	0	0	100	0	0
	100	0	12	88	98	2	0	91	6	3	100	0	0
	200	0	17	83	89	11	0	75	22	3	100	0	0

Table 3: *Simulation results for lag selection. For each setup in the table, the first, second and third columns represent the underfit, correct and overfit outcomes over 100 simulations.*

The performances of BRUTO, MARS and HaY for the first five processes are consistent with the results provided by Huang and Yang (2004). It should be noted that, in contrast to the cited paper, we have examined small to moderate sample sizes. Under these conditions, the promising algorithm HaY still demonstrates very good detection of true variables and steadily increases the frequency of correct fitting with increasing sample size. As reported in the cited paper, the cubic spline fitting faces some difficulties with NLAR2, where it performed relatively poorly in our simulations too. However, HaY is the single model which underfitted 100% of the NLAR4 realizations. Moreover, the stepwise approach used in this algorithm turns out to be an inevitable drawback in high-dimensional models. Combining both forward and backward stages with maximum number of d lags and number of candidate variables S_{max}, the forward stage requires $\sum_{i=1}^{S_{max}}(d-i+1)$ computations and the backward stage $\sum_{j=1}^{S_{max}} j$. Particularly, when $S_{max} = p$, where p denotes the number of covariates, the number of the required computations is $p * (p+1)$, which means that every covariate contributes quadratically to the computational burden. For high dimensions that would be an essential issue.

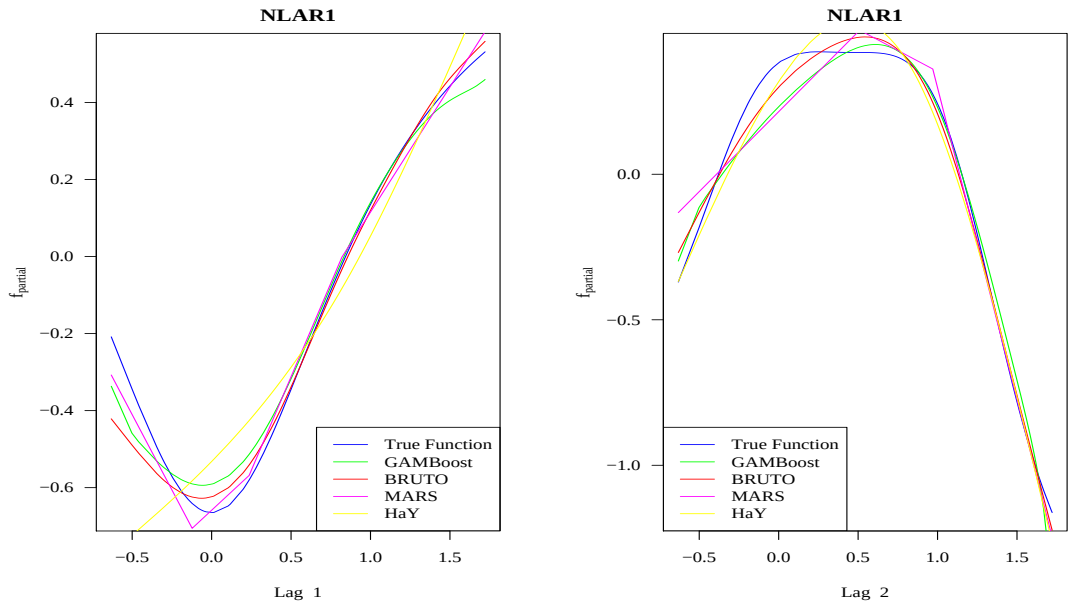


Figure 6: *A comparison between true lag functions (blue) and estimated functions for NLAR1. Left panel shows true partial function and four estimations of the first lag. Right panel shows respectively true partial function and four estimations of the second lag (with centering of all functions to mean zero).*

MARS shows an overall good performance. It is the single non-boosting method that manages to "catch" about a fourth of NLAR4 realizations with $T = 200$ correctly. On the other hand, BRUTO shows a rather erratic behaviour by favouring processes like NLAR2, NLAR3 and performing very poorly with others (NLAR1U1, NLAR1U2, NLAR4).

5.3 Dynamics Estimation

In simulations we can measure how precisely a fitting procedure reflects the true dynamics of a simulated process. In case of linear time series, a convenient measure would be the Euclidian distance between the true parameter vector and the estimated one. However, when dealing with nonparametric models we need some more sophisticated accuracy measure for the discrepancy between *functions*. Note that simply averaging the sum of squared residuals could be a misleading measure, due to its property to favour overfitting.[8] When we know the true underlying process in regression problems, a common measure for the goodness-of-fit is obtained in terms of MSE. However, in this case we fit stationary time series, which implies that true

[8]Overfitting is now used in sense of fitting the training data too closely and not in terms of selected variables as in the previous section.

Model	Length T	GAMBoost	BRUTO	MARS	HaY
NLAR1U1	50	.551	.984	.362	**.070**
	100	.205	.937	.183	**.038**
	200	.130	.338	.102	**.029**
NLAR1U2	50	.195	.442	.122	**.121**
	100	.179	.256	**.134**	.183
	200	.052	.156	**.050**	.074
NLAR1	50	.193	.241	.152	**.125**
	100	.063	.030	**.009**	.027
	200	.058	**.004**	.005	.022
NLAR2	50	.212	.206	.228	**.206**
	100	.208	.201	**.197**	.199
	200	.199	.190	**.188**	.195
NLAR3	50	.140	**.124**	.191	.154
	100	.103	**.076**	.099	.081
	200	.079	.069	**.067**	.069
NLAR4	50	**.332**	.380	.460	.411
	100	**.184**	.218	.291	.337
	200	**.099**	.103	.150	.193

Table 4: *Simulation results of average $MSPE$. The results of NLAR1U2, NLAR1 are multiplied by 10, NLAR2, NLAR3 are multiplied by 100 and NLAR1U1, NLAR4 are multiplied by* 10^3*. Boldface numbers indicate the best model performance for each setup.*

process is time invariant and therefore the expectation of its observations is always constant, i.e. $\mu_t = \mu = 0$. Therefore, the average sum of squared residuals between the realizations of true *partial functions* or *lag functions* (centered to mean zero) and the estimated ones gives a convenient goodness-of-fit measure. A single illustration of the differences between true lag functions and the estimated functions for NLAR1 with length $T = 200$ is depicted in Figure 6. In Appendix C.3 all true lag functions for the simulated processes are represented graphically. Let us denote with $\tilde{f}_k$ the kth lag function after centering it to mean zero, i.e. $\tilde{f}_k(\cdot) = f_k(\cdot) - \bar{f}_k(\cdot)$. Then the mean squared prediction error is given by

$$MSPE_k = T^{-1} \sum_{i=1}^{T} [\tilde{f}_k(y_{ik}) - \hat{\tilde{f}}_k(y_{ik})]^2 \tag{5.1}$$

where $\hat{\tilde{f}}_k$ is the estimated counterpart of $\tilde{f}_k$ and y_{ik} denotes the ith observations of the kth lag. The accuracy measure is the mean of the individual $MSPE$'s

$$MSPE = p^{-1} \sum_{k=1}^{p} MSPE_k. \tag{5.2}$$

The results of the average MSPE across all 100 simulation runs are summarized in Table 4, where the rows give the simulated series and the columns represent the different modelling techniques.

NLAR1U and NLAR1U2 are the most parsimonious models. Their dynamics seems to be explained very well by MARS and HaY, while BRUTO performed very poorly. The performance across fitting methods differed most within these two processes. For NLAR1U2, we notice that despite overfitting in sense of selected lags, boosting estimated the relevant function quite precisely, e.g. with $T = 200$. This suggests that the redundant functions were considered close to zero. It is reassuring to see that the functionals are really close to zero for overfitted lagged variables of NLAR1U2 in Figure 7. However, this was not the case for all processes which can be explained with the presence of serially correlated covariates. Strong serial dependence might mislead the fitting procedures to produce erroneous transformations. For instance, this is evident for boosting of NLAR1, where the third variable was strongly overfitted, see Figure 8 (function estimations, obtained by boosting for all time series are available graphically in Appendix C.4). The literature on nonparametric regression for dependent data is relatively sparse, especially when related to boosting. Of course, further study on the use of boosting algorithms in time series context is needed to justify the general use of this procedure.

With increasing number of significant covariates both BRUTO and GAMBoost strongly improved their performance. Moreover, excluding significant covariates by the non-boosting methods turned out to be very counterproductive at the largest model NLAR4, where GAMBoost provides the best description of the model's dynamics in all sample sizes. Boosting also performs comparably good with NLAR2 and NLAR3. However, my limited experience indicates that neither algorithm is universally superior to the others.

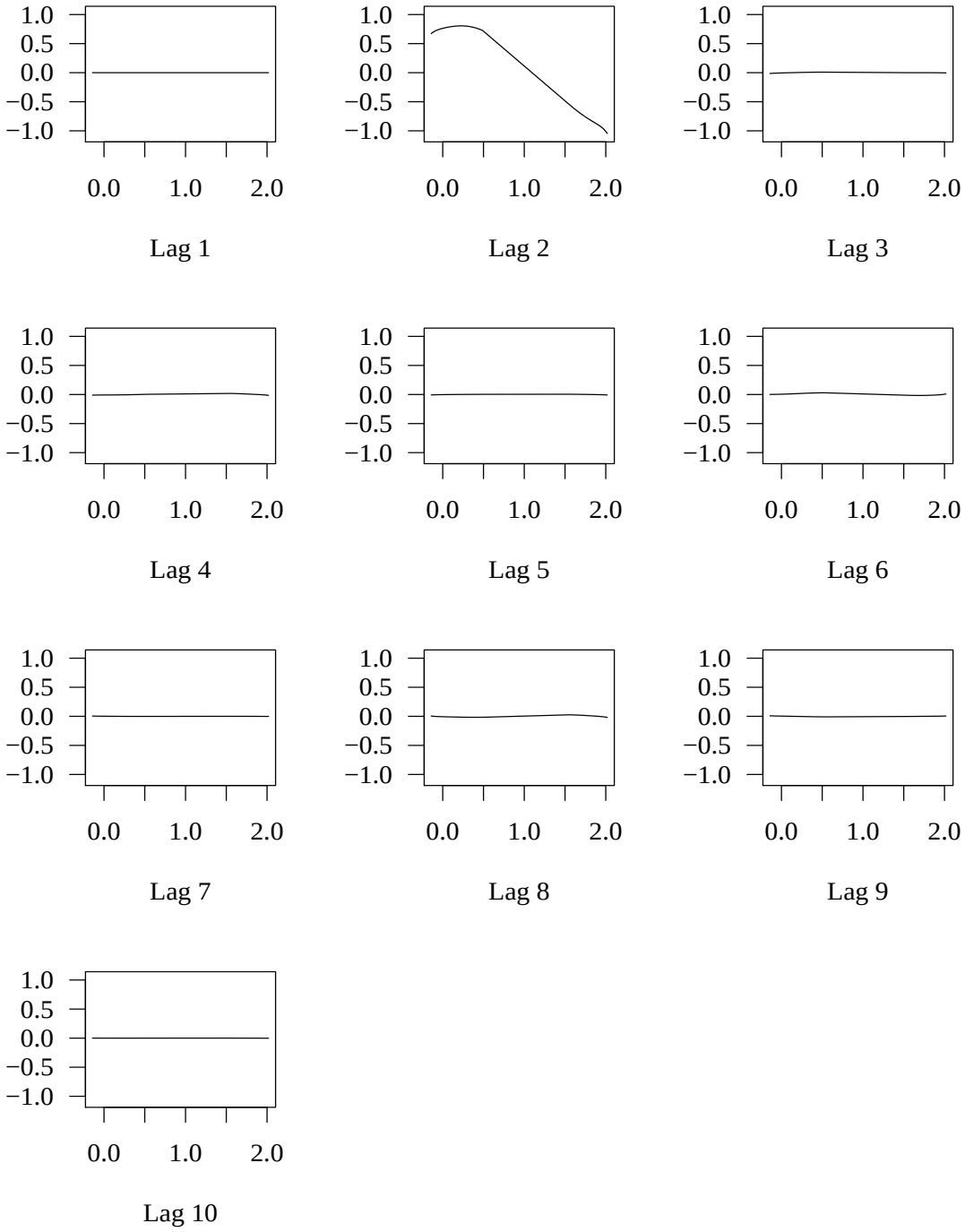


Figure 7: *Boosting estimations of the lag functions of NLAR1U2. True lag is 2. The functions are mean zero centered.*

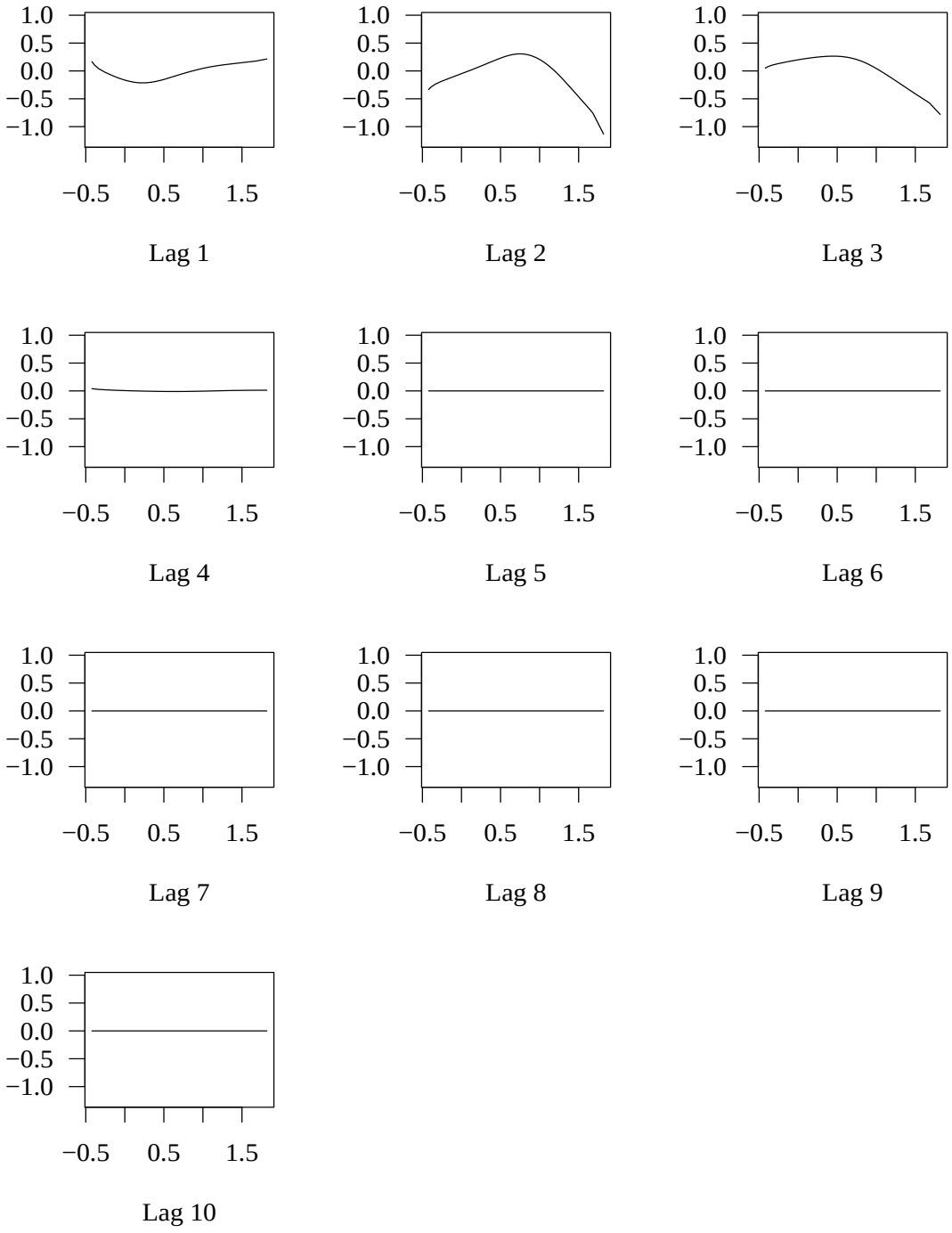


Figure 8: *Boosting estimations of the lag functions of NLAR1. True lags are 1 and 2. The functions are mean zero centered.*

6 Application

Boosting, along with other parametric and nonparametric models, will be applied to real data in this section. The target variable is the German industrial production (IP) from 1992:01 to 2006:08.[9] Forecasting of IP is frequently performed in practice. Contributions to the forecasting of German industrial production include Hüfner and Schröder (2002), Benner and Meier (2004), Dreger and Schumacher (2005) among many others. The series was obtained from Deutsche Bundesbank[10] and is seasonally and workday adjusted. Along with the leading indicators in Section 6.3, the data was also used by Robinzonov and Wohlrabe (2008). The exact monthly growth rates are taken to eliminate non-stationarity, that is

$$\Delta(IP) = \frac{IP_t - IP_{t-1}}{IP_{t-1}}.$$

Forecasting, being the major concern of macroeconomic time series, will be thoroughly explored in the current section. Some basic forecasting principles and a motivation about the choice of the consecutive forecasting strategy will be delivered in Section 6.1. An application, based solely on the industrial production time series will be delivered in Section 6.2. Most of the models used so far will be examined in the spirit of forecasting. The promising technique by Huang and Yang (2004) is omitted because Section 6.3 extends the available data set with exogenous variables, the so called leading indicators, and demonstrates how the additional information affects the performance of the models. The inclusion of exogenous variables and their lags rapidly increases the number of covariates, forming this way a classical high-dimensional modelling problem. In this context, the method of Huang and Yang (2004) would be no longer applicable.

6.1 Forecasting

Historically, the focus in forecasting has been on low-dimensional univariate or multivariate models, all sharing the common linearity in the parameters. Recently additional studies exist that investigate the forecasting performance of nonlinear time series models. Clements, Franses, and Swanson (2004) provide a thorough literature overview, Teräsvirta, van Dijk, and Medeiros (2005) examine the Smooth Transition Autoregressive (STAR) and neural networks models, Claveria, Pons, and Ramos (2007) study Markov-switching and Self-Exciting Autoregressive (SETAR)

[9] In order to circumvent any structural breaks due to the reunification, the data before 1991 is usually omitted. Data from 1991 is not included either, because some of the exogenous variables, such as ZEW Economic Sentiment, FAZ Indicator, have only been available after 1992.

[10] Series USNA01.

models. Elliot and Timmermann (2008) review almost all issues concerning economic forecasting.

When a specific model for a time series is assumed and a set of observations is given, we want to make statements about the future, thus *forecasting*. We will use the terms forecasting and *prediction*[11] interchangeably, although not quite precisely. The given set of observations is called a *training set* or an *information set*. The intention is using an information set $\mathcal{I}_t$,

$$\mathcal{I}_t = \{\mathbf{x}_\tau : \tau \leq t\}$$

our artificial outputs $\hat{y}_{t+h}, h = 1, 2, \ldots$ to be close enough to the real outputs y_{t+h}. If no exogenous variables are taken into account, then $\mathbf{x}_\tau$ contains information about IP only. Thus, forecasts for the 6 months from January 2003 to June 2003 are computed from models estimated using only data available through December 2002. Further on, we distinguish between one-period ahead forecast $\hat{y}_{t+1}$ and multi-period ahead forecast $\hat{y}_{t+h}$, where t is referred to as the forecasting origin and h is the forecasting horizon. To evaluate how concerned we are if our forecast is off by a particular amount, we need to specify a *cost function*. Therefore, minimizing the quadratic *expected cost* or *loss*

$$\mathrm{E}\,(y_{t+h} - \hat{y}_{t+h}|\mathcal{I}_t)^2 \tag{6.1}$$

is often set as an objective. Expression (6.1) is known as the *mean squared error*, associated with the forecast $\hat{y}_{t+h}$, denoted by

$$\mathrm{MSE}(\hat{y}_{t+h}) = \mathrm{E}\,(y_{t+h} - \hat{y}_{t+h}|\mathcal{I}_t)^2.$$

The choice of an accuracy measure is a major topic by itself. Hyndman and Koehler (2006) widely discuss and compare different measures of accuracy of times series forecasts such as Mean Absolute Percentage Error(MAPE), Median Absolute Percentage Error(MdAPE), Root Mean Squared Error (RMSE), Mean Absolute Scaled Error(MASE) and many more. They warn about the misleading properties of some of the measures. The references therein point the reader to different studies with often controversial conclusions about the "best" forecasting measure. Still, the literature being inconsistent, the MSE withstands the time proof and remains one of the most popular out-of-sample measures. Therefore, it is used in this elaboration as well.

[11] Prediction is a similar to forecasting, but is a more general term which is concerned with statements about the likely values of unobserved events, not necessarily those in the future.

In case that $\mathcal{I}_t$ contains endogenous variables only, the prediction $\hat{y}_{t+h}$ can be delivered successively through h one-period ahead forecasts. This means that multiperiod-ahead forecasts are made using a one-period ahead model, iterated forward to the desired number of periods. This strategy of forecasting is called *iterative forecasting* (Marcellino et al., 2006). It is applicable to both univariate and multivariate autoregressive modelling strategies.

The presence of exogenous variables, however, raises another question, namely how multi-steps-ahead forecasts are to be delivered. The endogenous variable is modelled on its own past values, as well as on the past values of the exogenous variables. Thus, the prediction concerns only the first out-of-sample endogenous variable, whereas no exogenous estimations are delivered. That obviously hinders the iterative approach of forecasting and demands an alternative technique for long term forecasts. A common strategy to overcome this problem is to use the so called *direct forecasting*[12] approach used in Marcellino, Stock, and Watson (2006), Chevillon and Hendry (2005). The idea is to make horizon-specific estimation model, where the response is the the multiperiod ahead value being forecasted, i.e. the dependent variable is directly modelled on the corresponding horizon (instead of starting with y_{t+1}) on its past values, as well as on past values of the exogenous variables. By doing so, every estimation refers to the future random variable in a straight manner. An obvious and, sadly, inevitable drawback of this technique is the fact that it requires h separate modellings in order to deliver all forecasts for a desired horizon. We will apply this type of forecasting to the univariate case in next section. Later on, the set of the explanatory variables will be extended and the forecasts will be delivered in the same fashion. That ensures a juxtaposition of the forecasts with and without exogenous variable, all other things being equal.

In the next section, the direct forecasting strategy is adopted by the GAMBoost, BRUTO and MARS for the univariate IP. We will also apply boosting with componentwise linear least squares learner and squared error loss, which is implemented in the add-on package `mboost` (Hothorn et al., 2008) via the function `glmboost()`. This type of boosting will be referred to as GLMBoost or for simplicity as linear boosting.

[12]It should be noted that the comparison iterated vs. direct forecasting of univariate time series is a theoretically ambiguous concept and the question which method is preferable to choose is rather an empirical one.

6.2 Univariate Forecasting of Industrial Production

Now we apply GLMBoost, GAMBoost, as well as BRUTO and MARS on the German industrial production. The univariate autoregressive model (AR) offers one of the simplest and most commonly used techniques for forecasting. It is easily applicable and therefore is often used as a benchmark model. The underlying assumption is that every alternative method should be at least as good as the autoregressive model in order to justify an increase in the model's complexity. The estimation of AR is carried out via the `ar()` function in package `stats` with AIC criterion (see Section 4.1).

So, the univariate case of IP is modelled. We have a total length of 176 observations. The initial information set is defined from the beginning until 2003:12, thus consisting of 144 observations. The maximal number of lags is limited for every fitting procedure to 12. Then, at the first stage twelve forecasts are calculated, i.e. clarify prognoses for 2004:1-2004:12. At the consecutive stage, the information set is enlarged with one observation and the corresponding horizon is re-estimated. We continue in this fashion until 2005:8, where the information set reaches its maximum, and then we compute the final twelve forecasts. Thus, we compute twenty stages in total. This method of enlarging the information set for every new forecasting horizon is called *recursive* scheme for forecasting. However, one could hold the information set at a fixed size, that is to leave out the oldest observation, when a new arrives. This is referred to as a *rolling* scheme for forecasting. The latter scheme is not considered in any greater detail than just mentioning it. See Elliot and Timmermann (2008) for further details concerning both schemes.

Table 5 gives a summary of the average squared forecast errors for IP, delivered by the different methods. Apparently, in short term forecasting, the standard autore-

Horizon	AR	GLMBoost	GAMBoost	BRUTO	MARS
1	.0668	.0626	.0638	.0845	.0892
6	.1052	.0784	.0845	.1029	.0898
12	.1214	.1211	.1145	.1168	.1169

Table 5: *Average squared forecast errors, multiplied by* 10^3*, of IP for 1, 6 and 12-periods ahead forecasts of the monthly industrial production growth rates in Germany. The results are based on 20 forecasts.*

gressive model is quite a hard one to overcome. This simple, yet very powerful, model is superior to BRUTO and MARS for short-term forecasting. On the other hand,

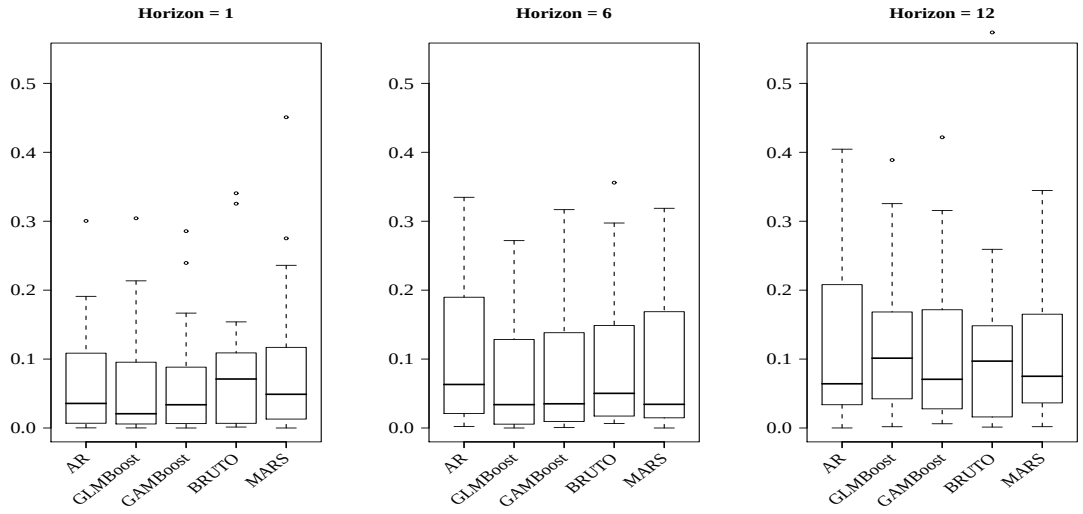


Figure 9: *Boxplots of the average squared forecast errors (multiplied by* 10^3*) for 1, 6 and 12-periods ahead forecasts of the univariate IP, based on 20 forecasts.*

GLMBoost and GAMBoost seem to be very precise in short term forecasting. With increasing forecasting horizon, all alternative models provide better forecasts for the monthly German industrial production growth rates, compared to AR. Both boosting methods prove to be very efficient in forecasting, especially the linear boosting in short and middle-term forecasting, where it offers the smallest prediction error in average. For the longest horizon GLMBoost remains at least as good as its parametric counterpart AR, but performs relatively poorly in comparison to GAMBoost, BRUTO and MARS. Figure 9 depicts the differences between the models of the prediction squared errors. In addition, Table 6 is considered to give an impression of the selected lags, chosen by the models. Selected lags may differ at the different stages, therefore we review the outcome at the stage where the information set reached its maximum (2005:08), being in this way the most representative. Both boosting techniques estimated quite large models, which is consistent with the results of the simulation study (the selected smooth components by GAMBoost are available in greater detail in Appendix C.2, Figure 12). This confirms the statement that regularization through shrinkage (as done by boosting) can provide superior results in terms of prediction to that obtained by restricting the number of covariates.

	AR	GLMBoost	GAMBoost	BRUTO	MARS
Selected lags	1,2	1,2,3,6,7,8,11	1,2,3,5,6,7,9,10,11,12	1	1

Table 6: *Selected variables when information set reached its maximum.*

6.3 Forecasting Industrial Production with Exogenous Variables

Forecasting of industrial production is based on the assumption that different leading indicators should relate significantly and stably with the response, and therefore positively influence its prediction. However, there are many leading indicators that "claim" such an appealing property. Usually, one indicator is taken and its forecasting potential is judged by a bivariate autoregressive model, e.g. Dreger and Schumacher (2005) compare four indicators. The additional dimension does not necessarily improve the forecasting quality, on the contrary, in case of an "inappropriate" extra variable, it deteriorates it. In consequence, different studies provide surprisingly a large variety of controversial conclusions about the forecasting power of the indicators. Instead of focusing on the indicators' prediction quality, we collect the nine most commonly used indicators and investigate how they affect the fitting. In other words, we will examine in this section, how redundant variables are considered from the fitting procedures. The aim is to gain knowledge, whether it is still possible to obtain good forecasts, despite the presence of probably inappropriate additional variables. Table 7 contains a list of the nine frequently used leading indicators on forecasting German IP (see Appendix A.1 for a detailed description of the indicators).

Indicator	Provider	Label
Ifo Business Climate	Ifo Institute	ifo
ZEW Economic Sentiment	ZEW Institute	zew
OECD Composite leading indicator for Germany	OECD	oecd
Early Bird Indicator	Commerzbank	com
FAZ Indicator	FAZ Institute	faz
Interest Rate: overnight	IMF	rovnght
Interest Rate: spread	IMF	rspread
Employment Growth	Bundesbank	emp
Factor	Bundesbank	factor

Table 7: *Leading Indicators.*

Since vector autoregressive analysis has evolved as a standard instrument in econometrics for analysing multivariate times series, we will consider nine bivariate mod-

els, each consisting of the IP and one leading indicator from Table 8 in its restricted[13] (VARr) and unrestricted (VAR) form. There are various add-on packages in `R` which deal with time series such as `tseries`, `dse`, `fMultivar`, `MSBVAR` and different functions in the base distribution of `R`. The package `vars`, provided by Bernhard Pfaff, offers "standard" tools in the context of purely vector autoregressive models and will be therefore used for the following computations of VAR and VARr. The corresponding information criterion is AIC.

The inclusion of one exogenous variable in the model means that boosting, BRUTO and MARS should deal with 24 covariates, i.e. twelve for the IP and twelve for the exogenous variable. The forecasting is conducted as described in Section 6.1, and the respective outcome is documented in Table 8. Every triplet shows the average performance of the corresponding models, respectively for 1, 6 and 12-periods ahead forecasts. In addition, it is indicated whether the forecast quality increased or decreased with respect to the univariate forecasts in Table 5. Both VAR and VARr are compared to AR.

Figure 10 depicts the results from Table 8 together with the AR model in a more compact form in order to put an emphasize on the comparison. Now follows a summary of the empirical results:

(a) The out-of-sample forecasting results from Table 8 suggest that both boosting techniques remain robust to the impact of the exogenous variables. GLMBoost remains almost immune to redundant variables. Apparently, in five cases of middle to long-term forecasting (ifo, zew, oecd, faz and rovnght) GLMBoost did not consider the exogenous variable at all. This explains why these forecasts are identical to the univariate case in Table 5. Transferred to the indicators, this interpretation suggests that they have only a short term effect on IP. In short-term forecasting GLMBoost remained very stable as well. Note that in one-period ahead forecasting the exogenous variable exerted negative impact in two cases (zew, com) only and outperformed AR in all cases except for com. In general, substantial changes of GLMBoost, compared to the univariate forecasting, were not found. That implies that linear boosting considered IP with its own lags to a larger extent than the remaining covariates. As a result, it showed a very strong overall performance and outperformed most of the models for one and six-periods ahead forecasting.

(b) The addition of exogenous variables changed the prediction power of GAM-

[13] The restrictions are obtained via standard statistical t-tests. See the documentation of function `restrict()` in package `vars`.

Leading Indicator	Horizon	VAR	VARr	GLMBoost	GAMBoost	BRUTO	MARS
ifo	1	.0990▼	.0771▼	.0623▲	.0661▼	.0845▲	.0892▲
	6	.1070▼	.1050▲	.0784▲	.0813▲	.1029▲	.0898▲
	12	.1215▼	.1207▲	.1211▲	.1127▲	.1168▲	.1169▲
zew	1	.0641▲	.0621▲	.0639▼	.0727▼	.0765▲	.0826▲
	6	.1048▲	.1053▼	.0784▲	.0839▲	.1157▼	.0892▲
	12	.1211▲	.1205▲	.1211▲	.1105▲	.1155▲	.1163▲
oecd	1	.0690▼	.0690▼	.0625▲	.0721▼	.0556▲	.1040▼
	6	.1108▼	.1108▼	.0784▲	.0825▲	.1244▼	.0829▲
	12	.1232▼	.1232▼	.1211▲	.1119▲	.1116▲	.1187▼
com	1	.0884▼	.0846▼	.0697▼	.0757▼	.0789▲	.0764▲
	6	.1082▼	.1092▼	.0773▲	.0836▲	.1093▼	.0908▼
	12	.1352▼	.1080▲	.1216▼	.1143▲	.1064▲	.1069▲
faz	1	.0684▼	.0522▲	.0626▲	.0711▼	.0830▼	.0916▼
	6	.1147▼	.1035▲	.0784▲	.0858▼	.1642▼	.0895▼
	12	.1180▲	.1200▲	.1211▲	.1144▲	.1388▼	.1047▲
rovnght	1	.0702▼	.0762▼	.0626▲	.0728▼	.0716▲	.0909▼
	6	.1082▼	.1078▼	.0784▲	.0877▼	.1111▼	.0895▼
	12	.1277▼	.1267▼	.1211▲	.1153▼	.1163▼	.1017▲
rspread	1	.0614▲	.0607▲	.0620▲	.0673▼	.0742▲	.0927▼
	6	.1060▼	.1040▼	.0781▲	.0840▲	.1004▲	.0889▲
	12	.1202▲	.1149▲	.1208▲	.1135▲	.1003▲	.1052▲
emp	1	.0704▼	.0762▼	.0624▲	.0637▲	.0704▲	.0916▼
	6	.1076▼	.1078▼	.0991▼	.0988▼	.1395▼	.0918▼
	12	.1267▼	.1267▼	.1262▼	.1190▼	.1361▼	.1081▲
factor	1	.0607▲	.0516▲	.0585▲	.0637▲	.0558▲	.0948▼
	6	.1021▲	.1006▲	.0811▼	.0871▼	.0988▲	.0913▼
	12	.1179▲	.1143▲	.1214▼	.1187▼	.1034▲	.1147▲

Table 8: *Average squared forecast errors of the monthly industrial production growth rates in Germany, with one leading indicator as an exogenous variable. The results are based on 20 forecasts, multiplied by* 10^3*. The symbol* ▲*indicates forecast improve with respect to Table 5 and* ▼*indicates decreased forecasting quality.*

Boost, BRUTO and MARS with varying success. Most notably GAMBoost and MARS performed comparably good and stable for six and twelve-periods ahead forecasting. This is best seen by the illustration in Figure 10. BRUTO improved its short term forecasting performance with almost every variable (except for faz), but in general remained worse than AR. For longer horizons it showed a rather erratic behaviour.

(c) There are four leading indicators, which proved to have good forecasting qual-

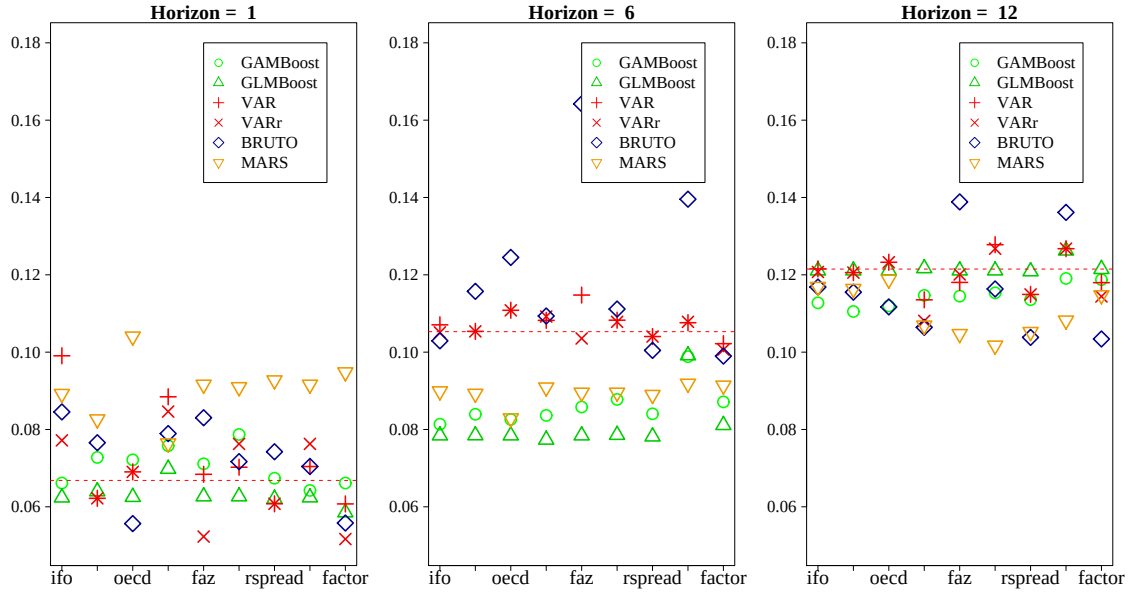


Figure 10: *Average squared forecast errors of the monthly industrial production growth rates in Germany, with one leading indicator as an exogenous variable. Dashed red-line shows the value of the univariate autoregressive model. The results are based on 20 forecasts, multiplied by* 10^3.

ity in terms of bivariate linear autoregression. These are zew, faz, rspread and factor, which increased the forecasting precision of IP, compared to AR. Moreover, the restricted bivariate autoregressive model with factor and faz provided the best short-term forecasts, but was easily outperformed for longer horizons. It is evident also that the restricted model is superior to the unrestricted one in most of the cases.

(d) From a computational point of view, MARS and GLMBoost were the fastest procedures. Closely followed by BRUTO, VAR and VARr, they all perform comparably fast. Boosting with P-Spline base learners was more computationally demanding.[14]

In Table 9 are collected lags, selected by boosting, BRUTO and MARS. The bivariate autoregressive models selected in most cases lag length of one (the results are not shown) which explains to some extent their relatively bad performance for

[14]Probably it is worth mentioning that I made a comparison between boosting with smoothing splines and with P-Splines as base procedure. P-Spline based boosting was considerably faster, while the estimations differences were negligibly small. This is consistent with the results, provided by Schmid and Hothorn (2007).

Variable	GLMBoost	GAMBoost	BRUTO	MARS
IP	1,2,3,6,7,8	1,2,3,5,6,7,9,11,12	1,2	1
ifo	1	1,7,11	1,2,3,4,5,8,11,12	-
IP	1,2,6,7,8	1,2,3,5,6,7,9,11,12	1,2	1
zew	1	1	1,2,3,5,6	1
IP	1,2,6,11,12	1,2,3,5,6,7,9,11,12	1,2	1
oecd	1	1,2,12	1,3,4,5,6,7,8,9,10	-
IP	1,2,6,7,8,11,12	1,2,3,5,6,7,9,10,12	1,2	1
com	1,2,3	1,7,10	3,4,6,8	1
IP	1,2,3,6,7,8,11	1,2,3,6,7,9,10,11,12	1	1
faz	-	7	1,2,3,4,5,6,7,8,9,10,11,12	-
IP	1,2,3,6,7,8,11	1,2,3,6,7,9,10,11,12	1	1
rovnght	-	7,10	2,3,4,5,6,7,8,9,10,11,12	-
IP	1,2,3,6,7,8,11,12	1,2,3,6,7,9,10,11,12	1	1,3
rspread	1	1,4	1,2,3,4,7,10,11,12	1,4,7
IP	1,2,3,6,7,8,11,12	1,2,3,6,7,9,10,11,12	1	1
emp	1,4,6,9,12	1,8,9,11,12	-	-
IP	1,2,6,7,8,12	1,3,6,7,9,10,12	1,2,3,7	1,2
factor	3,4,11	1,2,3,5,7,8,11,12	3,7	1,7

Table 9: *Selected lags of IP and of the exogenenous variable when the information set reached its maximum (horizon $h = 1$).*

longer forecasting horizons. It should be clearly stated that the selected lags by each method in Table 9 have resulted from a single, one-period ahead model with maximal information set. Therefore, they do *not* reflect the whole forecasting process and thus are not strictly related to the results, presented in Table 8. The intention is to gain a rather general impression of the selecting process.

It is reassuring to find a support to the statement that GLMBoost considered IP with its own lags more heavily than the exogenous variables. In accordance to the intuition this is probably the most plausible forecasting strategy, since we forecast IP, and in accordance to the forecasting results this was definitely the most successful one. Boosting with P-Spline base learners seems to be very consistent in the selection of endogenous lags - the same subset of IP lags is almost always present. At the same time, it estimates the largest models. BRUTO is the single modelling strategy, which repeatedly considered more exogenous than endogenous lags. This partially explains its erratic forecasting behaviour, each time conducted by the new indicator.

The biggest advantage of boosting is its capability to deal with high-dimensional models. This allows us to include all indicators in the model, together with their twelve lags in addition to the IP lags, forming this way 120 covariates. Not surprisingly, VAR and VARr were overwhelmed by the number of the estimation parameters and performed very poorly (the results are not shown). The remaining four modelling strategies are compared in Table 10. They deteriorate slightly, with respect to the univariate case in Table 5, where the only exception is MARS for 12-periods ahead forecasting. The results confirm that boosting is still capable to provide robust forecasts even when the number of the covariates increased dramatically. Note that boosting with least squares base procedure of regression with 120 covariates still outperforms the autoregressive model in one and six-periods ahead forecasting.

Horizon	GLMBoost	GAMBoost	BRUTO	MARS
1	.0650	.0731	.0904	.0974
6	.0994	.0938	.1137	.0952
12	.1259	.1290	.1261	.1037

Table 10: *Average squared forecast errors for German industrial production with nine leading indicators as exogenous variables. The results are based on 20 forecasts, multiplied by* 10^3.

In conclusion, for the monthly growth rates of the industrial production in Germany, I found evidence that boosting can be very competitive to the standard techniques. Particularly, least squares boosting predicts better than linear autoregressive models. The increased flexibility of the nonparametric models does not seem to pay-off in short term foreacasting, but manages to improve the prediction quality when the information content of the data decreases, i.e. low signal-to-noise ratio, which is observed in long-period ahead forecasting.

7 Concluding Remarks

In this thesis several parametric and nonparametric modelling techniques for autoregressive time series were shown, with boosting being the particular focus. By letting the covariates be lagged values of a time series, we have applied different strategies in order to identify its the relevant lags, estimate a model and possibly forecast the future realizations. In Section 5 we proposed componentwise boosting of additive autoregressive model with P-Spline base learners. Alternative modelling strategies were also applied on several nonlinear autoregressive time series. It was evidenced that boosting of high-order autoregressive time series can be very competitive in terms of dynamics estimation. Unlike regression analysis, however, the serial dependence in time series data might mislead the fitting procedure to produce erroneous transformations. Care must be taken in using boosting algorithms in time series with strong serial correlation of the data. Further study on the use of boosting in time series context is needed to justify the general use of this procedure.

Another boosting strategy with parametric base learners (GLMBoost) was included in order to perform a forecasting comparison, based on real world data in Section 6. The forecasting comparison was conducted over the monthly growth rates of the German industrial production (IP). Both boosting strategies managed to outperform the benchmark in macroeconomic forecasting, namely the linear autoregressive model. Moreover, it became clear that GLMBoost was the most successful strategy in terms of short and middle-term forecasting.

Additionally, the model was extended with different exogenous variables (leading indicators). We had nine indicators available and we included each of them separately, in addition to the target variable - the industrial production. Our intention was to test whether these variables do indeed improve the forecasting quality of the industrial production and how boosting handles these high-dimensional models. Thus, having formed nine high-dimensional models, we forecasted again the monthly growth rates of IP. Linear bivariate autoregressive models were also considered as standard tools for forecasting.

The variables' impact on the forecasting quality had debatable success, since in many of the cases their inclusion worsened the forecasting performance, compared to the univariate case. GLMBoost, on the other hand, was almost immune to redundant variables by performing at least as good as in the univariate case. In one-period ahead forecasting, GAMBoost was affected by the additional variables rather strongly, which was counterproductive for its overall performance, when compared

to the univariate case. The increased flexibility of GAMBoost was useful, however, in middle and long term forecasting, where the information content of the data is very low, i.e. it has low signal-to-noise ratio.

Finally, IP and all nine indicators were included together in a single regression model, each with its twelve lags, thus forming a high-dimensional model with 120 covariates. Both boosting strategies only slightly worsened their forecasting performances, compared to the forecasting of the univariate IP. Under these conditions, GLMBoost was still superior to the simple autoregressive model, which is frequently used in practice. An issue to be addressed further are the possible combinations of the leading indicators, to be included in the model. Besides, there are numerous tuneable parameters in boosting, that can open new perspectives when altered. Careful research on the effect of the base procedure, the loss function or even the shrinkage factor could possibly improve the boosting fit and respectively its forecasting power. Boosting is definitely very fruitful research field for further extensions, since GLMBoost managed to perform strongly with the default parameters.

Another crucial topic for further development addresses the multivariate generalization of boosting. The first steps toward high dimensionality in the response were made by Lutz and Bühlmann (2006), who provided theoretical grounds and empirical evidence for its usability. Applying this approach would open new perspective for forecasting with boosting, based on iterative forecasts of multivariate models.

Appendices

A APPENDIX

A.1 The Choice of Leading Indicators

In Section 6.3 nine leading indicators are chosen. These are summarized as follows:

The Ifo Business Climate Index is based on about 7,000 monthly survey responses of firms in manufacturing, construction, wholesaling and retailing. The firms are asked to give their assessments of the current business situation and their expectations for the next six months. The balance value of the current business situation is the difference of the percentages of the responses "good" and "poor", the balance value of the expectations is the difference of the percentages of the responses "more favourable" and "more unfavourable". The business climate is a transformed mean of the balances of the business situation and the expectations. For further information see Goldrian (2007).

The ZEW Indicator of Economic Sentiment is ascertained monthly. Up to 350 financial experts take part in the survey. The indicator reflects the difference between the share of analysts that are optimistic and the share of analysts that are pessimistic for the expected economic development in Germany in six months (see Hüfner and Schröder, 2002).

The FAZ indicator (Frankfurter Allgemeine Zeitung) pools survey data and macroeconomic time series. It consists of the Ifo index (0.13), new orders in manufacturing industries (0.56), the real effective exchange rate of the Euro (0.06), the interest rate spread (0.08), the stock market index DAX (0.01), the number of job vacancies (0.05) and lagged industrial production (0.11). The Ifo index, orders in manufacturing and the number of job vacancies enter the indicator equation in levels, while the other variables are measured in first differences.

The Early Bird indicator, compiled by Commerzbank, also pools different time series and stresses the importance of international business cycles for the German economy. Its components are the real effective exchange rate of the Euro (0.35), the short-term real interest rate (0.4), defined as the difference between the short-term nominal rate and core inflation, and the purchasing manager index of U.S. manufactures (0.25).

The OECD composite leading indicator is delivered by using a modified version of the Phase-Average Trend method (PAT) developed by the US National Bureau of

Economic Research (NBER). The indicator is compiled by combining de-trended component series in either their seasonally adjusted or raw form. The component series are selected based on various criteria such as economic significance, cyclical behaviour, data quality, timeliness and availability. For Germany the following time series are compiled: Orders inflow or demand: tendency (manufacturing) (% balance), Ifo Business climate indicator (manufacturing) (% balance), Spread of interest rates (% annual rate), Total new orders (manufacturing), Finished goods stocks: level (manufacturing) (% balance) and Export order books: level (manufacturing) (% balance).

Financial indicators, such as overnight interbank interest rate an interest spread, are taken as possible predictors as well. Since the seminal paper by Estrella and Hardouvelis (1991) financial indicators are more and more in focus for forecasting. Stock and Watson (2003) review this literature and conduct a huge case study for different OECD countries by forecasting Gross Domestic Product (GDP), Inflation and Industrial production. The growth of the employment in Germany has been taken from their paper.

Finally, a factor indicator obtained from a large data set from Germany, is included. The data set contains German quarterly GDP and 111 monthly indicators from 1992 to 2006.[15]

[15]The estimated factor was provided by Christian Schumacher and is based on the paper Marcellino and Schumacher (2007).

B APPENDIX

B.1 Derivation of 2.13

We have to minimize

$$\begin{aligned}\mathcal{S}_p(\boldsymbol{\beta}) &= ||\mathbf{y}-\mathbf{Z}\boldsymbol{\beta}||^2+\lambda\boldsymbol{\beta}^T\boldsymbol{\Lambda}\boldsymbol{\beta}\\ &= (\mathbf{y}-\mathbf{Z}\boldsymbol{\beta})^T(\mathbf{y}-\mathbf{Z}\boldsymbol{\beta})+\lambda\boldsymbol{\beta}^T\boldsymbol{\Lambda}\boldsymbol{\beta}\\ &= \mathbf{y}^T\mathbf{y}-\mathbf{y}^T\mathbf{Z}\boldsymbol{\beta}-(\mathbf{Z}\boldsymbol{\beta})^T\mathbf{y}+\boldsymbol{\beta}^T\mathbf{Z}^T\mathbf{Z}\boldsymbol{\beta}+\lambda\boldsymbol{\beta}^T\boldsymbol{\Lambda}\boldsymbol{\beta}\\ &= \mathbf{y}^T\mathbf{y}-2\boldsymbol{\beta}^T\mathbf{Z}^T\mathbf{y}+\boldsymbol{\beta}^T(\mathbf{Z}^T\mathbf{Z}+\lambda\boldsymbol{\Lambda})\boldsymbol{\beta}.\end{aligned}$$

In order to minimize $\mathcal{S}_p(\boldsymbol{\beta})$ w.r.t $\boldsymbol{\beta}$ we compute the first-order partial derivatives. For that purpose we use the rules for matrix differentiation: $\frac{d\boldsymbol{a}^T\mathbf{x}}{d\mathbf{x}}=\frac{d\mathbf{x}^T\boldsymbol{a}}{d\mathbf{x}}=\boldsymbol{a}^T$ and $\frac{d\mathbf{x}^TA\mathbf{x}}{d\mathbf{x}}=\mathbf{x}^T(A^T+A)$, where $\boldsymbol{a}$ is a vector and A is a matrix. Then

$$\begin{aligned}\frac{d\mathcal{S}_p(\boldsymbol{\beta})}{d\boldsymbol{\beta}}=-2\mathbf{Z}^T\mathbf{y}+\boldsymbol{\beta}^T(2\mathbf{Z}^T\mathbf{Z}+2\lambda\boldsymbol{\Lambda}) &= \mathbf{0}\\ 2\boldsymbol{\beta}^T(\mathbf{Z}^T\mathbf{Z}+\lambda\boldsymbol{\Lambda}) &= 2\mathbf{Z}^T\mathbf{y}\\ \Rightarrow\qquad \hat{\boldsymbol{\beta}}=(\mathbf{Z}^T\mathbf{Z}+\lambda\boldsymbol{\Lambda})^{-1}\mathbf{Z}^T\mathbf{y}.&\end{aligned}$$

□

B.2 Derivation of 3.22

$$\begin{aligned}\hat{\boldsymbol{\mu}}_{(l+1)} &= \hat{\boldsymbol{\mu}}_{(l)}+\mathbf{Z}_j\,\hat{\boldsymbol{\gamma}}_{j(l+1)}\\ &= \hat{\boldsymbol{\mu}}_{(l)}+\mathbf{Z}_j(\mathbf{Z}_j^T\mathbf{Z}_j+\lambda\boldsymbol{\Lambda})^{-1}\mathbf{Z}_j^T(\mathbf{y}-\hat{\boldsymbol{\mu}}_{(l)})\\ &= \hat{\boldsymbol{\mu}}_{(l)}+\mathcal{H}^{(\hat{s}_l)}(\mathbf{y}-\hat{\boldsymbol{\mu}}_{(l)})\\ &= \hat{\boldsymbol{\mu}}_{(l)}+\mathcal{H}^{(\hat{s}_l)}(\mathbf{y}-\hat{\boldsymbol{\mu}}_{(l)}+\hat{\boldsymbol{\mu}}_{(l-1)}-\hat{\boldsymbol{\mu}}_{(l-1)})\\ &= \hat{\boldsymbol{\mu}}_{(l)}+\mathcal{H}^{(\hat{s}_l)}(\mathbf{y}-\hat{\boldsymbol{\mu}}_{(l-1)}-\mathcal{H}^{(\hat{s}_{l-1})}(\mathbf{y}-\hat{\boldsymbol{\mu}}_{(l-1)}))\\ &= \hat{\boldsymbol{\mu}}_{(l)}+\mathcal{H}^{(\hat{s}_l)}(I-\mathcal{H}^{(\hat{s}_{l-1})})(\mathbf{y}-\hat{\boldsymbol{\mu}}_{(l-1)})\\ &= \hat{\boldsymbol{\mu}}_{(l)}+\mathcal{H}^{(\hat{s}_l)}(I-\mathcal{H}^{(\hat{s}_{l-1})})\ldots(I-\mathcal{H}^{(\hat{s}_1)})(I-\mathcal{H}^{(\hat{s}_0)})\mathbf{y}.\end{aligned}$$

Then follows

$$\begin{aligned}\hat{\boldsymbol{\mu}}_{(m)} &= \sum_{j=0}^{m}\mathcal{H}^{(\hat{s}_j)}\prod_{i=1}^{j-1}(I-\mathcal{H}^{(\hat{s}_i)})\mathbf{y}\\ \Rightarrow\qquad \mathcal{H}^{(\hat{s}_m)} &= \sum_{j=0}^{m}\mathcal{H}^{(\hat{s}_j)}\prod_{i=1}^{j-1}(I-\mathcal{H}^{(\hat{s}_i)})\\ \text{where}\qquad \mathcal{H}^{(\hat{s}_0)} &= \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T.\end{aligned}$$

□

B.3 Derivation of the GLS Estimator in VAR model

The GLS estimator of $\boldsymbol{\beta}$ is obtained via minimizing the following expression:

$$\begin{aligned} S(\boldsymbol{\beta}) &= \boldsymbol{\varepsilon}'\boldsymbol{\Sigma}_{\varepsilon}^{-1}\boldsymbol{\varepsilon} \\ &= \boldsymbol{\varepsilon}'(I_T \otimes \boldsymbol{\Sigma})^{-1}\boldsymbol{\varepsilon} \\ &= (\mathbf{y} - (\boldsymbol{Z}' \otimes I_q)\boldsymbol{\beta})'(I_T \otimes \boldsymbol{\Sigma})^{-1}(\mathbf{y} - (\boldsymbol{Z}' \otimes I_q)\boldsymbol{\beta}) \\ &= \mathbf{y}'(I_T \otimes \boldsymbol{\Sigma})^{-1}\mathbf{y} + \boldsymbol{\beta}'(\boldsymbol{Z} \otimes I_q)(I_q \otimes \boldsymbol{\Sigma}^{-1})(\boldsymbol{Z}' \otimes I_q)\boldsymbol{\beta} \\ &\quad - 2\boldsymbol{\beta}'(\boldsymbol{Z} \otimes I_q)(I_T \otimes \boldsymbol{\Sigma}^{-1})\mathbf{y} \\ &= \mathbf{y}'(I_T \otimes \boldsymbol{\Sigma}^{-1})\mathbf{y} + \boldsymbol{\beta}'(\boldsymbol{Z}\boldsymbol{Z}' \otimes \boldsymbol{\Sigma}^{-1})\boldsymbol{\beta} - 2\boldsymbol{\beta}'(\boldsymbol{Z} \otimes \boldsymbol{\Sigma}^{-1})\mathbf{y}. \end{aligned} \tag{B.1}$$

In order to minimize $S(\boldsymbol{\beta})$ w.r.t $\boldsymbol{\beta}$ we compute the first-order partial derivatives:

$$\frac{\partial S(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = 2(\boldsymbol{Z}\boldsymbol{Z}' \otimes \boldsymbol{\Sigma}^{-1})\boldsymbol{\beta} - 2(\boldsymbol{Z} \otimes \boldsymbol{\Sigma}^{-1})\mathbf{y}. \tag{B.2}$$

Equating (B.2) to zero gives the *normal equations*

$$(\boldsymbol{Z}\boldsymbol{Z}' \otimes \boldsymbol{\Sigma}^{-1})\hat{\boldsymbol{\beta}} = (\boldsymbol{Z} \otimes \boldsymbol{\Sigma}^{-1})\mathbf{y} \tag{B.3}$$

which are used to obtain the GLS-Estimator

$$\begin{aligned} \hat{\boldsymbol{\beta}} &= ((\boldsymbol{Z}\boldsymbol{Z}')^{-1} \otimes \boldsymbol{\Sigma})(\boldsymbol{Z} \otimes \boldsymbol{\Sigma}^{-1})\mathbf{y} \\ &= ((\boldsymbol{Z}\boldsymbol{Z}')^{-1}\boldsymbol{Z} \otimes I_q)\mathbf{y}. \end{aligned} \tag{B.4}$$

□

B.4 Derivation of the OLS Estimator in VAR model

The OLS estimator of $\boldsymbol{\beta}$ is obtained via minimizing the following expression:

$$\begin{aligned} \tilde{S}(\boldsymbol{\beta}) &= \boldsymbol{\varepsilon}'\boldsymbol{\varepsilon} \\ &= (\mathbf{y} - (\boldsymbol{Z}' \otimes I_q)\boldsymbol{\beta})'(\mathbf{y} - (\boldsymbol{Z}' \otimes I_q)\boldsymbol{\beta}) \end{aligned}$$

Then we use the Gaussian estimator for $\boldsymbol{\beta}$, which produces

$$\begin{aligned} \hat{\boldsymbol{\beta}} &= ((\mathbf{Z}' \otimes I_q)'(\mathbf{Z}' \otimes I_q))^{-1}(\mathbf{Z}' \otimes I_q)'\mathbf{y} \\ &= ((\mathbf{Z} \otimes I_q)(\mathbf{Z}' \otimes I_q))^{-1}(\mathbf{Z} \otimes I_q)\mathbf{y} \\ &= (\mathbf{Z}\mathbf{Z}' \otimes I_q)^{-1}(\mathbf{Z} \otimes I_q)\mathbf{y} \\ &= ((\mathbf{Z}\mathbf{Z}')^{-1} \otimes I_q)(\mathbf{Z} \otimes I_q)\mathbf{y} \\ &= ((\mathbf{Z}\mathbf{Z}')^{-1}\mathbf{Z} \otimes I_q)\mathbf{y}. \end{aligned}$$

□

B.5 Definition of the column stacking operator vec

Let A be a $(n \times m)$ matrix of the form:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ . & . & . & . \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{pmatrix}$$

Then operator vec transforms A into a $(nm \times 1)$ vector by stacking the columns, that is,

$$\text{vec}(A) = \begin{pmatrix} a_{11} \\ \vdots \\ a_{n1} \\ a_{12} \\ \vdots \\ a_{n2} \\ \vdots \\ a_{1m} \\ \vdots \\ a_{nm} \end{pmatrix}$$

Then the following vectorization rules apply:

$$\text{vec}(A + B) = \text{vec}(A) + \text{vec}(B) \tag{1}$$

$$\text{vec}(AB) = (I_q \otimes A)\text{vec}(B) \tag{2}$$

$$= (B' \otimes I_m)\text{vec}(A) \tag{3}$$

$$\text{vec}(ABC) = (C' \otimes A)\text{vec}(B) \tag{4}$$

where A, B and C denote $(m \times n), (n \times p)$ and $(p \times q)$ matrices and $\otimes$ denotes the Kronecker Product (Eves, 1980).

C APPENDIX

C.1 B-Splines

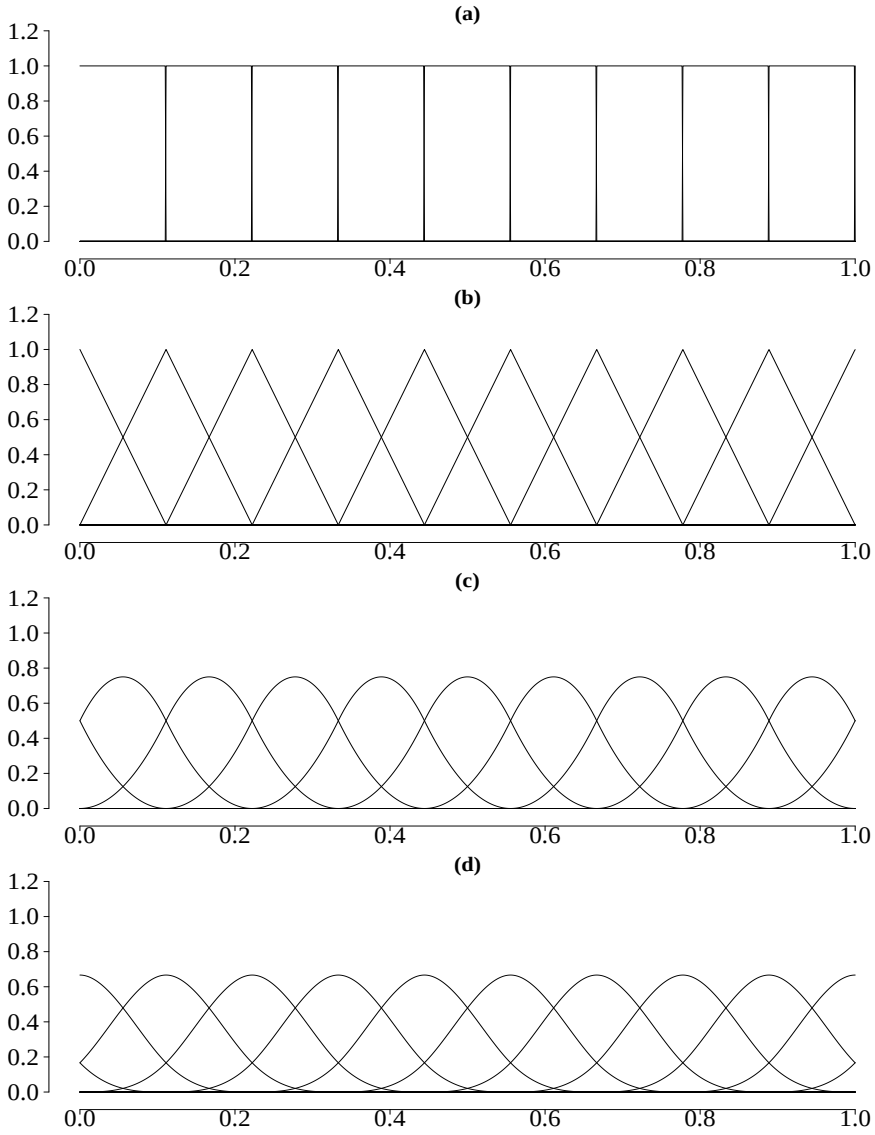


Figure 11: *The sequence of B-Splines of order 0 (a), 1 (b), 2 (c) and 3(d).*

C.2 Selected Lags by GAMBoost

In Table 12 are shown the selected smooth components by GAMBoost, based on the monthly growth rates of the German industrial production for the period 1992:01-2005:08 (maximal information set for forecasting).

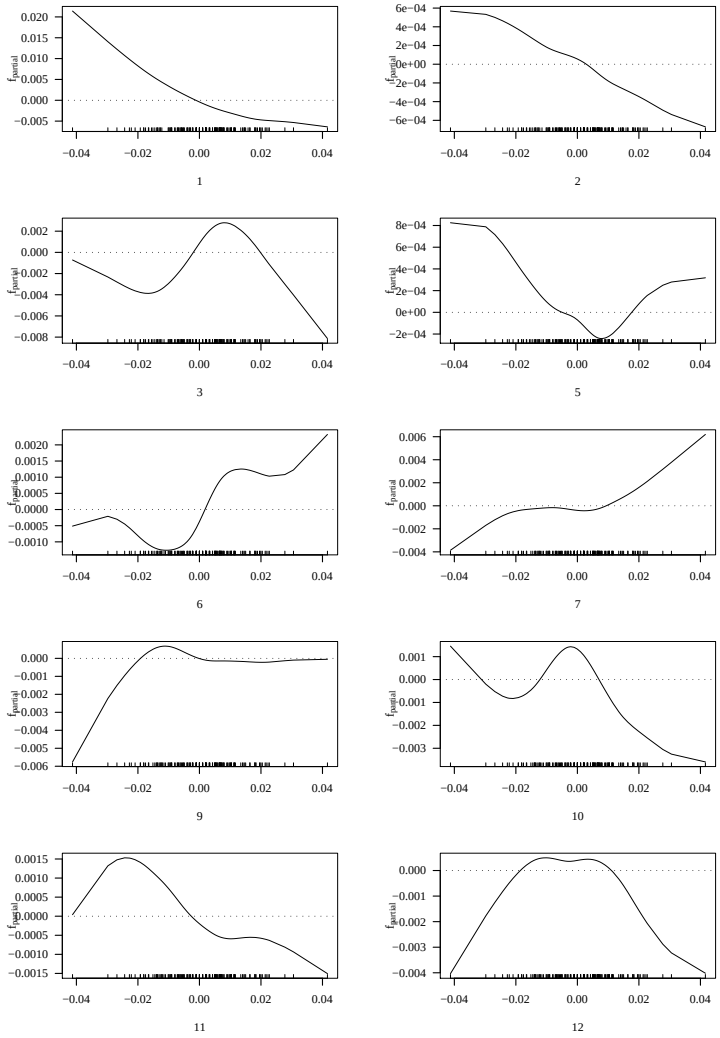
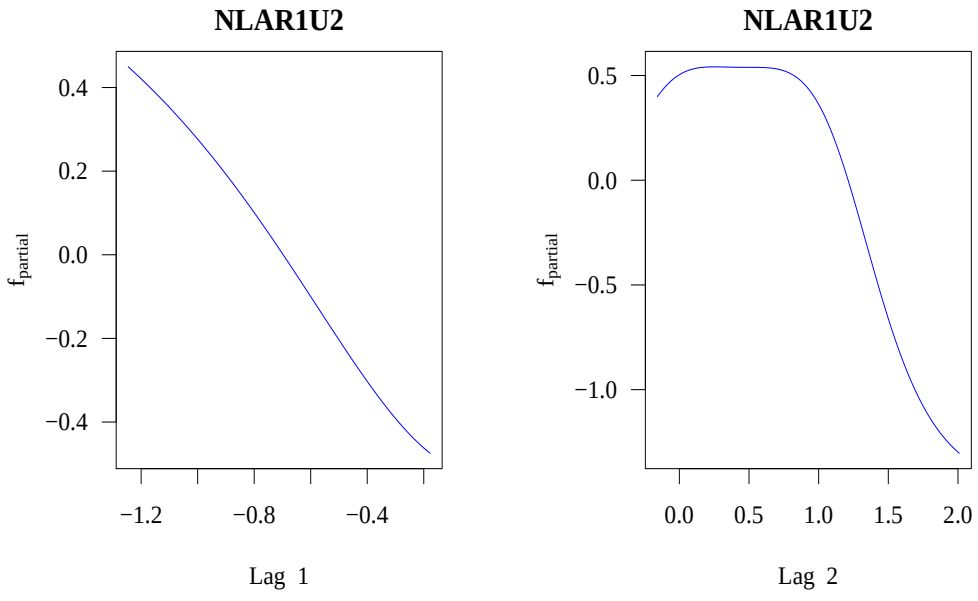
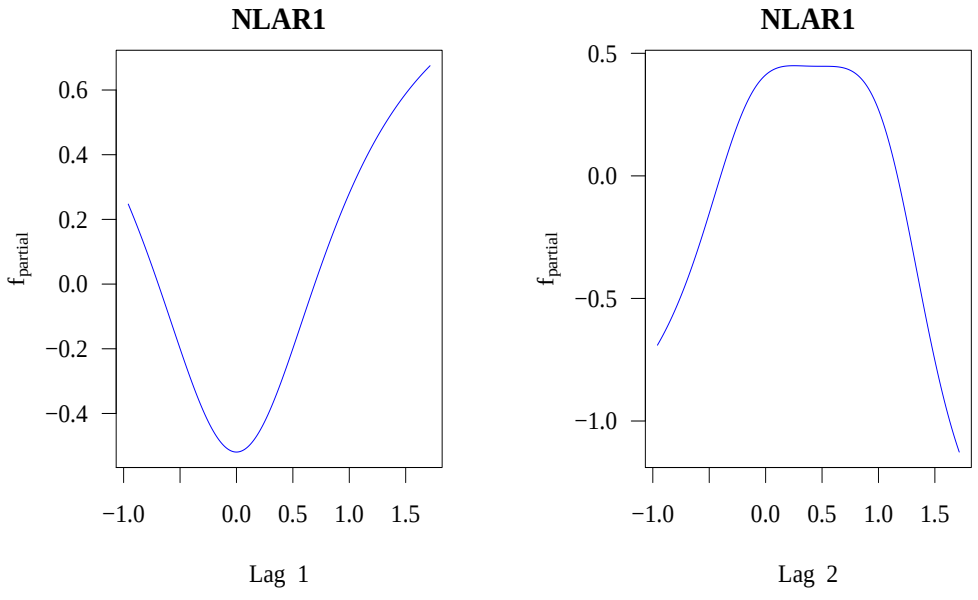


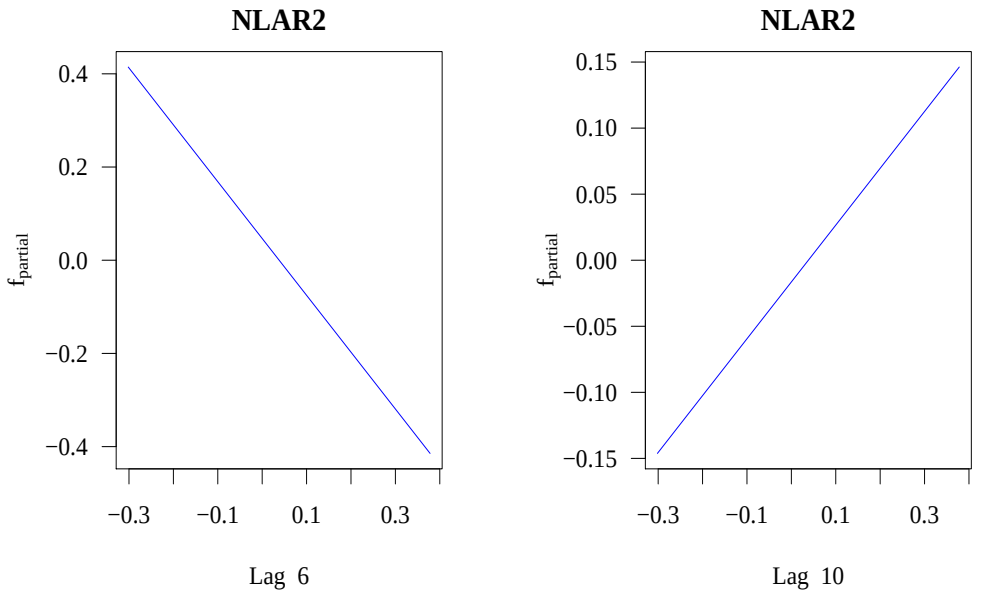
Figure 12: *Selected smooth components by GAMBoost of the univariate IP when the information set reached its maximum.*

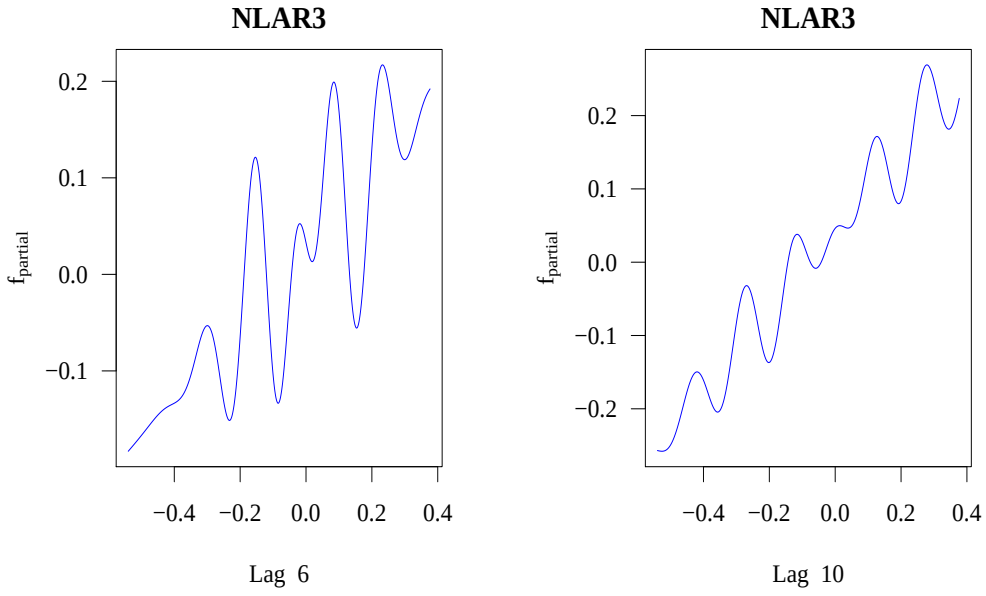
C.3 Lag Functions

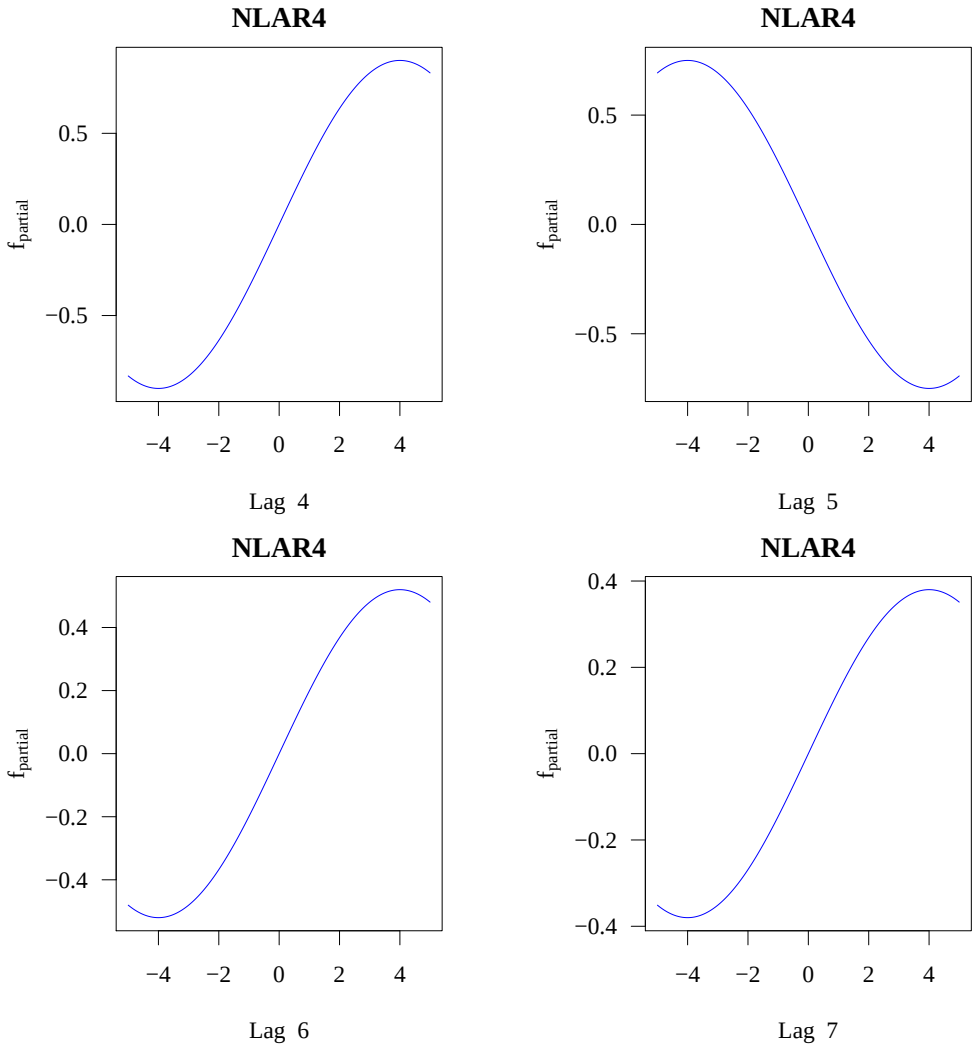
Here are depicted the lag functions of all models from Section 5, Table 2 (with centering to mean zero).






NLAR2
0.4
0.2
0.0
−0.2
−0.4
f_partial
−0.3
−0.1
0.1
0.3
Lag 6
NLAR2
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
f_partial
−0.3
−0.1
0.1
0.3
Lag 10


NLAR3
0.2
0.1
0.0
−0.1
f_partial
−0.4
−0.2
0.0
0.2
0.4
Lag 6
NLAR3
0.2
0.1
0.0
−0.1
−0.2
f_partial
−0.4
−0.2
0.0
0.2
0.4
Lag 10


NLAR4
f_partial
0.5
0.0
−0.5
−4
−2
0
2
4
Lag 4
NLAR4
f_partial
0.5
0.0
−0.5
−4
−2
0
2
4
Lag 5
NLAR4
f_partial
0.4
0.2
0.0
−0.2
−0.4
−4
−2
0
2
4
Lag 6
NLAR4
f_partial
0.4
0.2
0.0
−0.2
−0.4
−4
−2
0
2
4
Lag 7

C.4 Boosting Estimates of Lag Functions

Here are depicted boosting estimates of the lag functions from Section 5, Table 2 with maximal lag length 10 (with centering to mean zero).

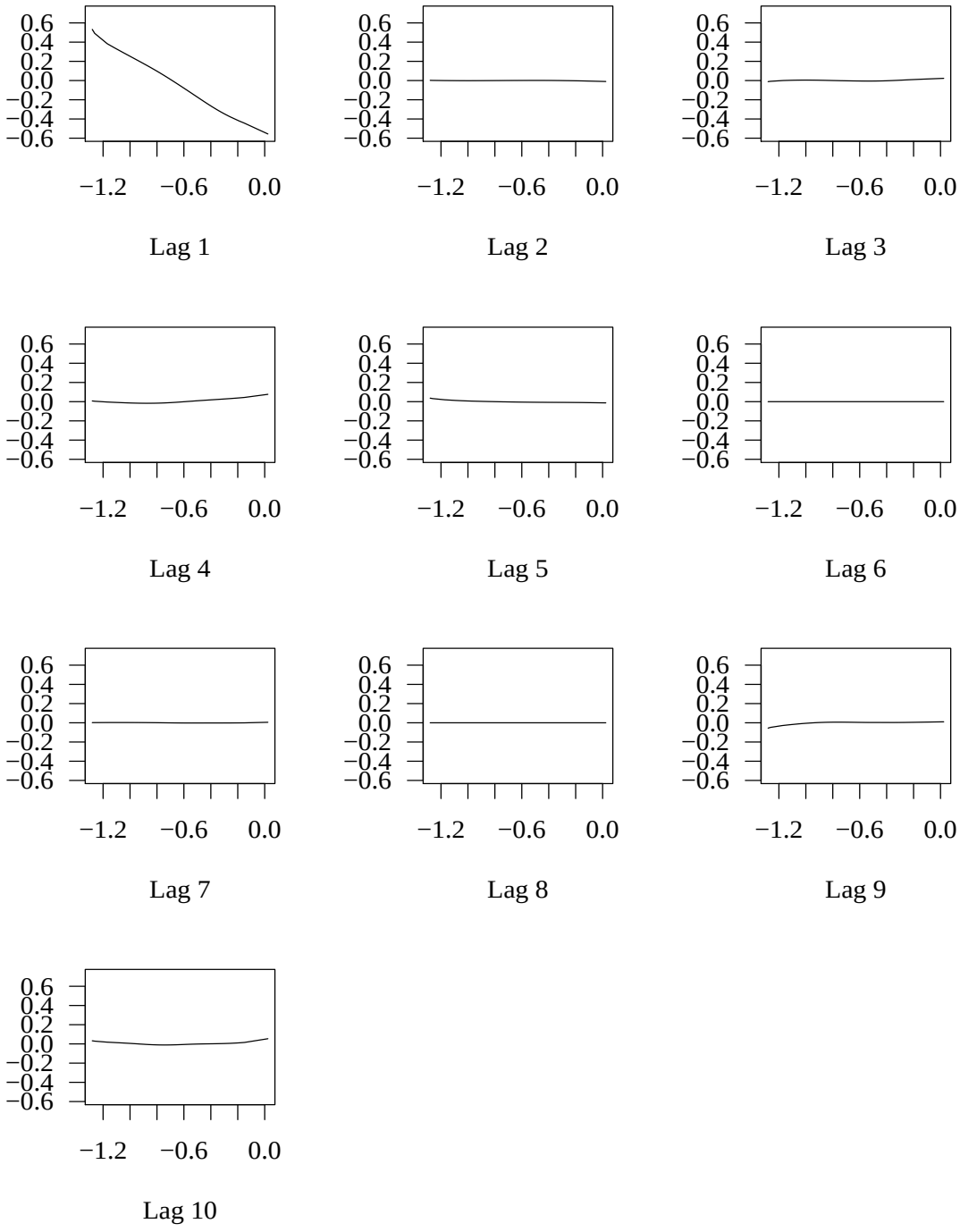


Figure 13: *NLAR1U1 with true lag: 1*

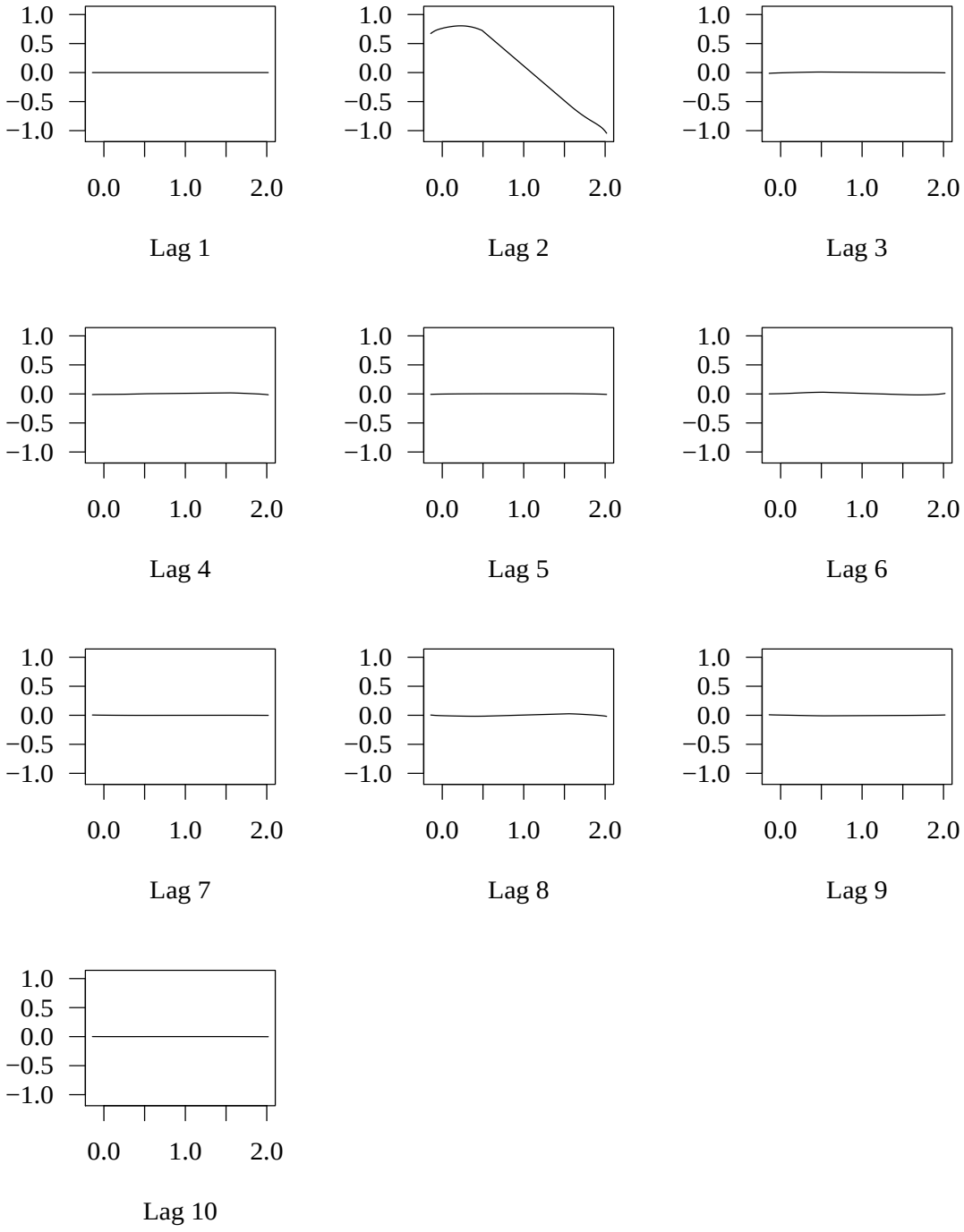


Figure 14: *NLAR1U2 with true lag: 2*

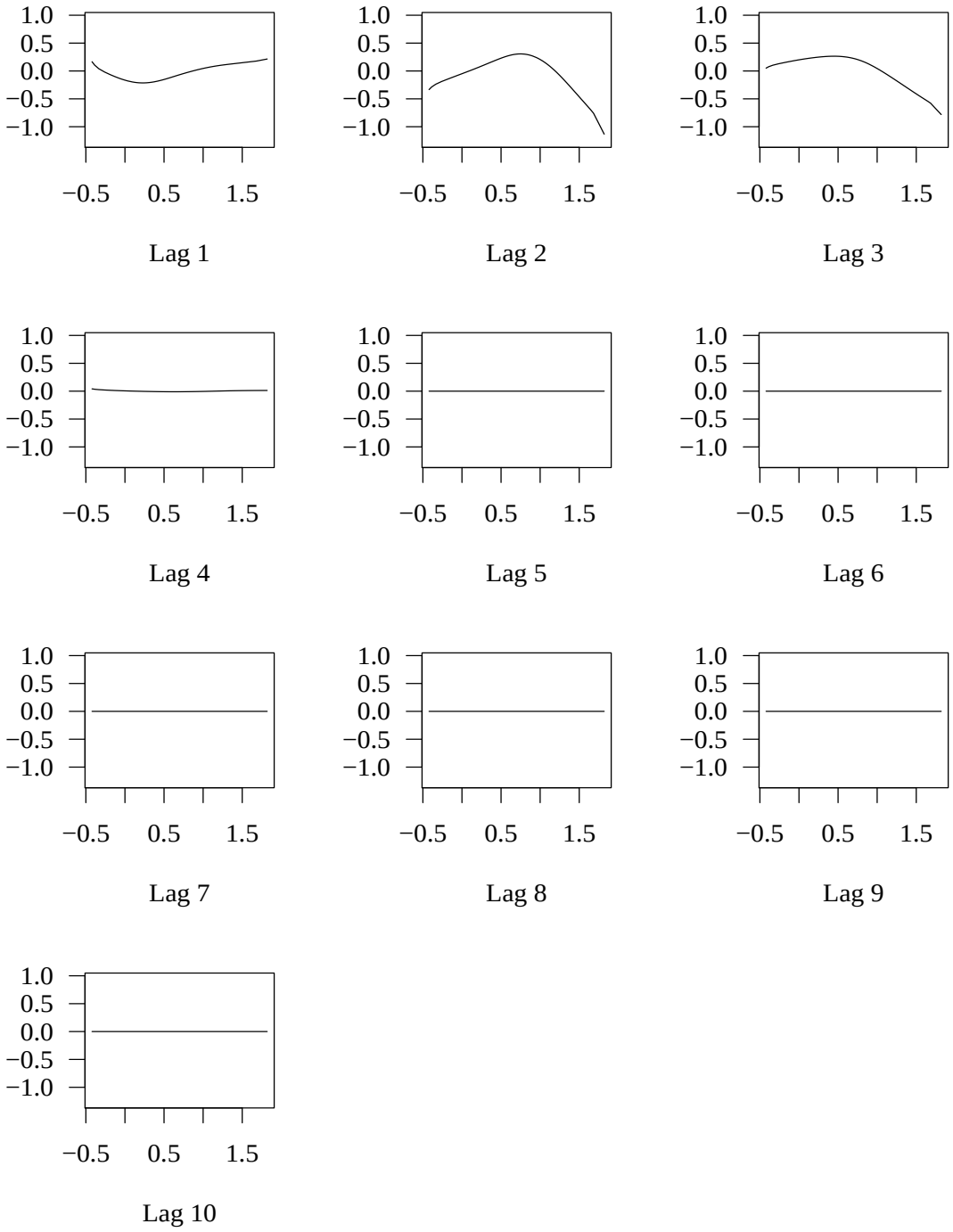


Figure 15: *NLAR1 with true lags: 1,2*

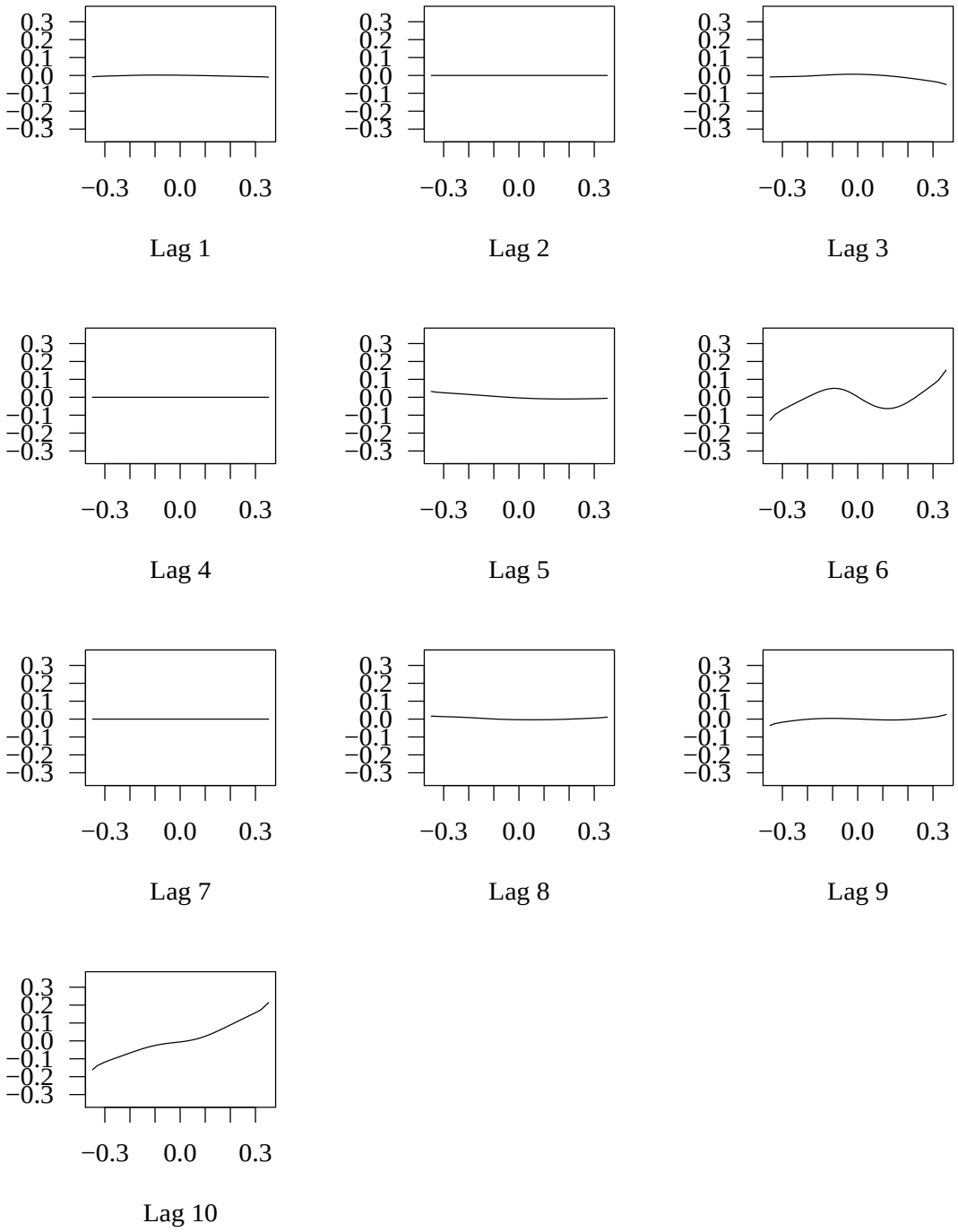


Figure 16: *NLAR2 with true lags: 6,10*

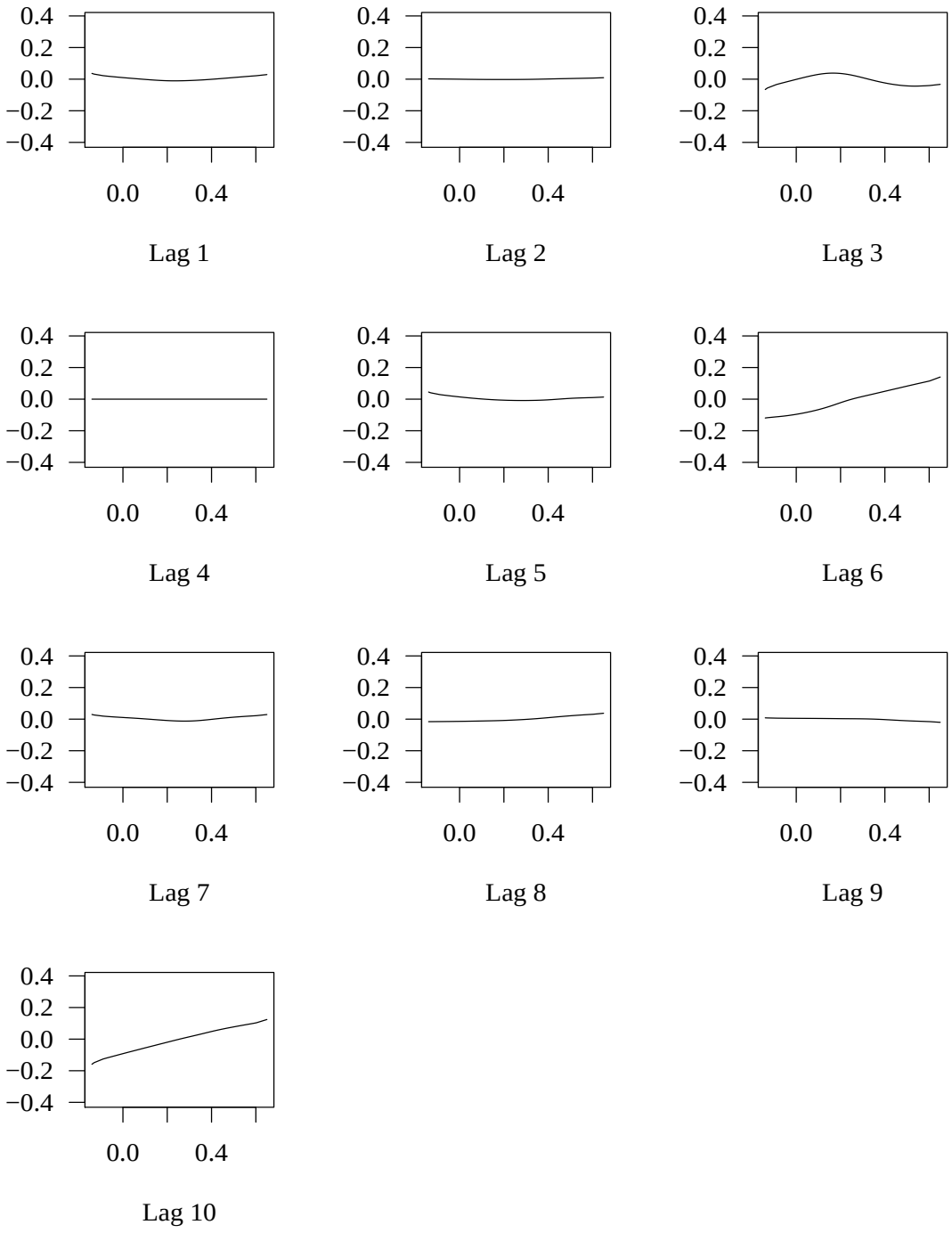


Figure 17: *NLAR3 with true lags: 6,10*

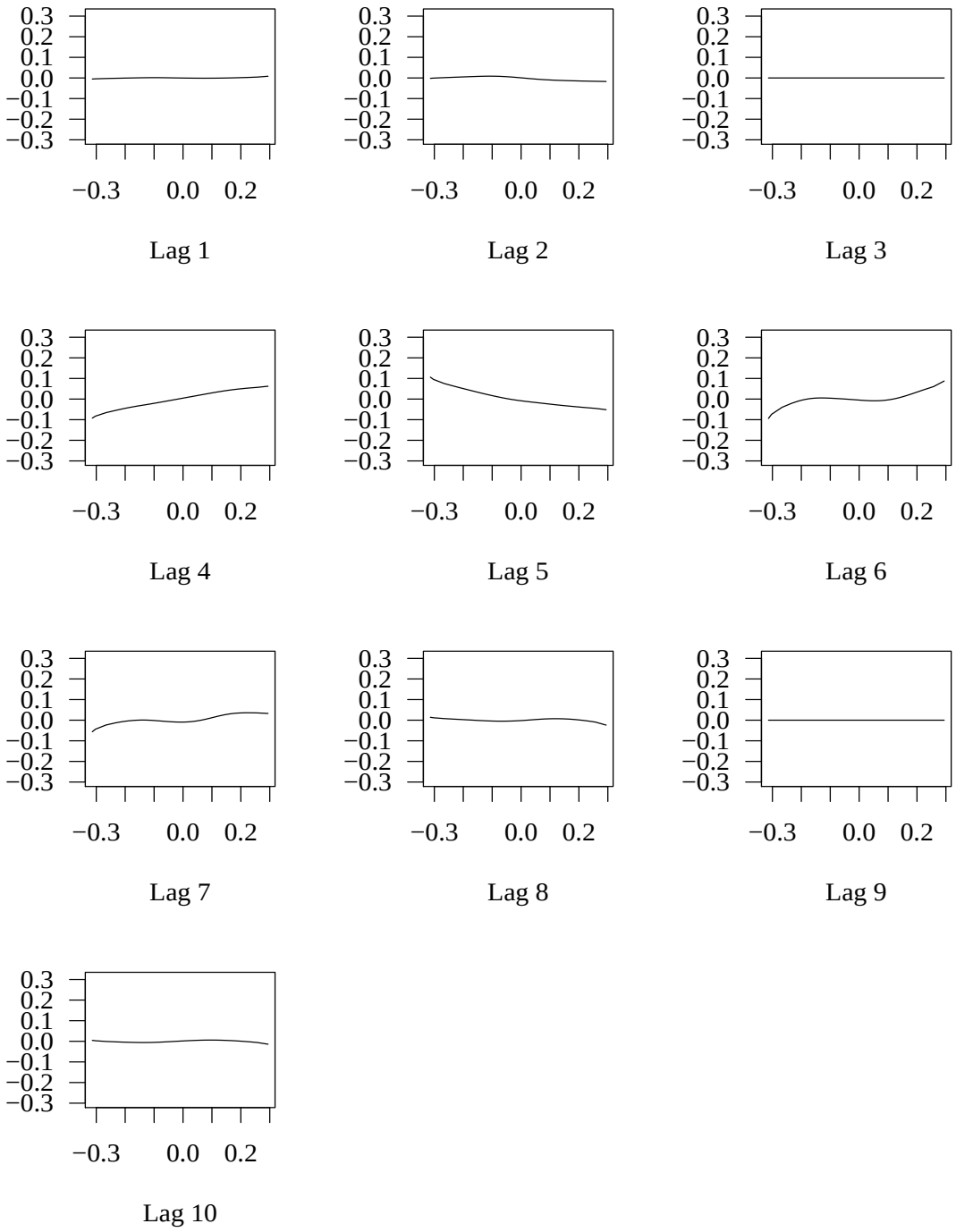


Figure 18: *NLAR4 with true lags: 4,5,6,7*

D APPENDIX: R-Code

In the enclosed CD most of the functions are documented with sufficient amount of comment lines, which facilitates their further use. Therefore, they are only briefly described here. Since I also used the novel technique by Huang and Yang (2004), not implemented in any statistical software by the time of writing this thesis, below I propose my version of the corresponding algorithm in full detail (see function `stepwise()`). Now we summarize the usage of the basic functions and helpers on which the present thesis was grounded.

- `bbsformula()` builds formula for P-Spline base learners, read by `gamboost()`.
- `dev.on()` is a helper, which facilitates creation of nice Encapsulated PostScript files, used for the production of all figures in this thesis.
- `embed2()` is a wrapper of the `embed()` function. It embeds a time series into a low-dimensional Euclidean space and is properly adjusted to suit the data to direct forecasting purposes.
- `mspe()` defines the accuracy measure MSPE from Section 5.3.
- `myfcst()` facilitates multi-period ahead forecasts with different modelling strategies.
- `partial.fit()` extracts the lag functions of a fitted object, obtained by boosting, BRUTO, MARS or HaY.
- `p-spline()` is a replica of the example in `smooth.construct` (see. `?p.spline`) in the package `mgcv`. It allows application of the HaY algorithm with P-Splines.
- `setcheck()` provides the lag selection rule, as defined in Section 5.2.
- `simts()` defines the artificial times series from the simulation study (see Table 2 in Section 5).
- `stepwise()` is an implementation of the stepwise algorithm, provided by Huang and Yang (2004). See the function below for more details.

In addition, there are several R-files which had the following purposes.

- `forecasting.r` was used to carry out the forecasting results in the application section.
- `mc-bruto.r`, `mc-gamboost.r`, `mc-HaY.r` and `mc-mars.r` were used to carry out the Monte Carlo simulations. The files are named after the corresponding methods of interest.

- `plotting.application.r` contains the sequence of commands, used to produce all figures in the application section.
- `plotting.simulation.r` contains the sequence of commands, used to produce all figures in the simulation section.
- `plotting.r` contains the sequence of commands, used to produce all figures in the sections 2 to 4.
- `simulation.storage.r` reads, i.e.`source`, the processes, defined by `simts()` and stores them locally into a pre-specified directory.

```
#####################################################################
# Implementation of the Stepwise selection method proposed by Huang and Yang.
# INPUT:  'y'  - univariate time-series or data.frame with 'lags'.
#         'd'  - number of candidate covariates.
#         'Smax' - maximum number of covariates in one model.
#         'basis' - either 'ps' or 'Bs'
#         'fPath' - print forward and backward procedure.
# OUTPUT: A list of class "stepwise_HaY".
#####################################################################
stepwise <- function(y, d=10, Smax=10, basis="Bs", fPath=F)
     { if(all(basis!=c("Bs","ps"))) stop("'basis' must be 'Bs' or 'ps'.")
       if(is.null(dim(y)) || min(dim(y)==1))
       dat  <- embed2(y,dimension=d) else
       dat <- y
       endF <- forward(dat, Smax=Smax, basis, fPath)
       fset <- endF[[Smax]]$set
       dat2 <- dat[,c(names(dat)[1],fset)]
       endB <- backward(dat2, basis, fPath)
       final<- c(endF,endB)
       BICs <- sapply(seq(along=final), function(i) final[[i]]$BIC)
       index<- which(BICs==min(BICs))[1]
       result <- final[[index]]
       class(result) <- "stepwise_HaY"
       result
     }

#####################################################################
# NOTE:  'forward' doesn't compute intercept-model.
```

```
# Reason: this is a job of 'backward'.
#########################################################################
forward <- function(dat, Smax, basis, fPath=F)
{   n   <- NROW(dat)
    resp<- names(dat)[1]
    set <- names(dat)[-1]
    forwardset <- NULL
    endF <- vector("list", Smax); count <- 1
    while(length(forwardset) < Smax){
       models <- vector("list", length(set)); k <- 1
       for(z in set)  # z="Series1.L2"
          { work        <- buildModel(resp,c(forwardset,z),basis,n) # buildModel
            models[[k]] <- calc.MSE_BIC(work,dat=dat)
            k = k + 1
          }
       MSEs <- sapply(seq(along=models), function(j) models[[j]]$MSE)
       ind  <- which(MSEs==min(MSEs))[1]
       #BICs <- sapply(seq(along=models), function(j) models[[j]]$BIC)
       #ind  <- which(BICs==min(BICs))[1]
       forwardset    <- models[[ind]]$set
       endF[[count]] <- models[[ind]]; count=count+1
       set <- set[!set%in%forwardset]
       if(fPath) cat(paste(forwardset,collapse=" "),"\n")
      }
    endF
}

#########################################################################
# NOTE:  'backward' doesn't compute full-model and 'single-term model'.
# Reason: this is a job of 'forward'.
backward <- function(dat, basis, fPath=F)
{    n <- nrow(dat)
     resp <- names(dat)[1]
     set  <- names(dat)[-1]
     endB <- vector("list", length(set)-1)
     count <- 2
     workNull  <- buildModel(resp,NULL, basis, n)       # buildModel
     endB[[1]] <- calc.MSE_BIC(workNull, dat=dat)
```

```
     while(length(set)>2){
        models <- vector("list", length(set)); k <- 1
        for(z in set)  # z="Series1.L1"
           { work        <- buildModel(resp,set[!set%in%z], basis, n) # buildMode
             models[[k]] <- calc.MSE_BIC(work, dat=dat)
             k = k + 1
           }
        MSEs <- sapply(seq(along=models), function(j) models[[j]]$MSE)
        ind  <- which(MSEs==min(MSEs))[1]
       # BICs <- sapply(seq(along=models), function(j) models[[j]]$BIC)
       # ind  <- which(BICs==min(BICs))[1]
        endB[[count]] <- models[[ind]]; count=count+1
        set  <- models[[ind]]$set
        if(fPath) cat(paste(set,collapse=" "),"\n")
       }
    endB
}

#########################################################################
# INPUT for buildModel():
#          'predictors' is a set ot covariates
#          'basis' is a basis specification
# OUTPUT:  list() with builded 'formula' and 'set' for gam fitting.
# Example: predictors = c("lag1","lag3"); n=100
#          basis = "ps"
#          basis = "cr"
#          buildModel(predictors, basis, n)
#          buildModel(NULL, basis, n)
# Note:
# When evaluating gam(Y ~ s(formula, bs = "ps")) for penalized splines
# the model doesn't work without the function 'p-spline.r'.
#########################################################################
buildModel <- function(responce, predictors, basis, n)
{ if(is.null(predictors)) {
     res= list(formula=as.formula(paste(responce,"~ 1")),set="1")
     attr(res, "basis") <- basis
     class(res) <- "buildModel"
     return(res)
```

```
    }
  if( basis=="ps"){
     pred  <- paste("s(",predictors,",bs='",basis,"')",sep="",collapse="+")
     model <- as.formula(paste(responce,"~",pred))
    }
  if( basis=="Bs"){
     k     <- 2                          # tuning constant
     knots <- floor((k*n)^(1/5))         # obtain the number of knots ? problemati
     pred  <-  paste("s(",predictors,",bs='cr',k=",knots,",fx=T)",sep="",collapse
     model <- as.formula(paste(responce,"~",pred))
     }
   res   <- list()
   res$formula <- model
   res$set     <- predictors
   attr(res, "basis") <- basis
   class(res) <- "buildModel"
   return(res)
}

#############################################################################
# INPUT for calc.MSE_BIC():
#         'formula_and_set' should be OUTPUT of the function 'buildModel'
#############################################################################
calc.MSE_BIC <- function(formula_and_set, dat)
{  if(class(formula_and_set)!="buildModel") stop("Object must be from class 'buil
   require(splines)
   require(mgcv)
   res  <- list()
   fit  <- gam(formula_and_set$formula, data=dat)
   if( attributes(formula_and_set)$basis == "ps" ){
      n    <- length(fit$fitted)
      N    <- sum(fit$edf)
      MSE  <- mean((fit$resid)^2)
      BIC  <- log(MSE) + N/n*log(n)
     }
   if( attributes(formula_and_set)$basis == "Bs" ) {
    n    <- length(fit$fitted)
    N    <- 1 + sum(fit$edf) + length(coef(fit)) #sum(fit$edf)==sum(fit$hat)
```

```
    MSE  <- mean((fit$resid)^2)
    BIC  <- log(MSE) + N/n*log(n)
     }
   res$fit     <- fit
   res$set     <- formula_and_set$set
   res$MSE     <- MSE
   res$BIC     <- BIC
   return(res)
}
```

References

Akaike, H. (1973). Information theory and an extension of the maximum likelihood. *Second International Symposium on Information Theory. Akademiai Kaido, Budapest, Hungary*, 267–281.

Akaike, H. (1974). A new look at the statistical identification model. *IEEE Transactions on Automatic Control 19*(3), 716–723.

Bellman, R. (1961). Adaptive control processes: A guided tour.(A RAND Corporation Research Study.

Benner, J. and C. Meier (2004). Prognosegüte alternativer frühindikatoren für die konjunktur in deutschland. *Jahrbücher für Nationalökonomie und Statistik 224*(6), 637–652.

Binder, H. (2006). *GAMBoost: Generalized additive models by likelihood based boosting*. R package, version 0.9-3.

Box, G. and G. Jenkins (1976). Time Series Analysis: Forecasting and Control, rev. ed. *San Francisco*.

Breiman, L. (1996). Bagging Predictors. *Machine Learning 24*(2), 123–140.

Breiman, L. (1998). Arcing Classifiers. *The Annals of Statistics 26*(3), 801–824.

Brockwell, P. and R. Davis (1991). *Time Series: Theory and Methods*. Springer.

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

Bühlmann, P. and B. Yu (2003). Boosting With the L2 Loss: Regression and Classification. *Journal of the American Statistical Association 98*(462), 324–339.

Chan, K. and H. Tong (1986). On estimating thresholds in autoregressive models. *Journal of Time Series Analysis 7*(3), 179–190.

Chen, R. and R. Tsay (1993). Nonlinear Additive ARX Models. *Journal of the American Statistical Association 88*(423), 955–967.

Chevillon, G. and D. Hendry (2005). Non-parametric direct multi-step estimation for forecasting economic processes. *International Journal of Forecasting 21*(2), 201–218.

Claveria, O., E. Pons, and R. Ramos (2007). Business and consumer expectations and macroeconomic forecasts. *International Journal of Forecasting 23*(1), 47–69.

Clements, M., P. Franses, and N. Swanson (2004). Forecasting economic and financial time-series with non-linear models. *International Journal of Forecasting 20*(2), 169–183.

De Boor, C. (1978). *A practical guide to splines*. Springer, Berlin.

De Boor, C. (2001). *A Practical Guide to Splines*. Springer.

Dolado, J. and H. Lütkepohl (1996). Making wald tests work for cointegrated VAR systems. *Econometric Reviews 15*(4), 369–386.

Dreger, C. and C. Schumacher (2005). Out-of-sample Performance of Leading Indicators for the German Business Cycle. Single vs Combined Forecasts. *Journal of Business Cycle Measurement and Analysis 2*(1), 71–88.

Eilers, P. and B. Marx (1996). Flexible Smoothing with B-splines and Penalties. *Statistical Science 11*(2), 89–102.

Elliot, G. and A. Timmermann (2008). Economic Forecasting. *Journal of Economic Literature 66(1)*, 3–56.

Estrella, A. and G. Hardouvelis (1991). The Term Structure as a Predictor of Real Economic Activity. *The Journal of Finance 46*(2), 555–576.

Eves, H. (1980). *Elementary Matrix Theory*. Courier Dover Publications.

Fahrmeir, L. and G. Tutz (2001). *Multivariate Statistical Modelling Based on Generalized Linear Models*. Springer.

Freund, Y. and R. Schapire (1996). Experiments with a new boosting algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference 148*, 156.

Friedman, J. (1991). Multivariate Adaptive Regression Splines. *The Annals of Statistics 19*(1), 1–67.

Friedman, J. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics 29*(5), 1189–1232.

Friedman, J. and B. Silverman (1989). Flexible parsimonious smoothing and additive modelling (with discussion). *Technometrics 31*, 3–39.

Granger, C. and A. Andersen (1978). *An Introduction to Bilinear Time Series Models*. Göttingen.

Gu, C. (2002). *Smoothing Spline Anova Models*. Springer.

Hamilton, J. (1989). A New Approach to the Economic Analysis of Nonstationary Time Series and the Business Cycle. *Econometrica 57*(2), 357–384.

Hamilton, J. (1994). *Time Series Analysis*. Princeton University Press.

Hannan, E. and B. Quinn (1979). The Determination of the Order of an Autoregression. *Journal of the Royal Statistical Society. Series B (Methodological) 41*(2), 190–195.

Hastie, T. and R. Tibshirani (1990). *Generalized Additive Models*. Chapman & Hall/CRC.

Hastie, T., R. Tibshirani, and J. Friedman (2001). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

Hornik, K., F. Leisch, and B. D. . Ripley (2006). *mda: Mixture and flexible discriminant analysis*. R package, version 0.3-2.

Hothorn, T., P. Bühlmann, T. Kneib, and M. Schmid (2008). *mboost: Model-Based Boosting*. R package, version 1.0-1.

Huang, J. and L. Yang (2004). Identification of non-linear additive autoregressive models. *Journal of the Royal Statistical Society Series B(Statistical Methodology) 66*(2), 463–477.

Huber, P. (1964). Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics 35*(1), 73–101.

Hüfner, F. and M. Schröder (2002). Prognosegehalt von ifo-geschäftserwartungen und zew-konjunkturerwartungen: Ein ökonometrischer vergleich. *Jahrbücher für Nationalökonomie und Statistik 222*(3), 316–336.

Hurvich, C., J. Simonoff, and C. Tsai (1998). Smoothing parameter selection in nonparametric regression using an improved Akaike information criterion. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 60*(2), 271–293.

Hyndman, R. and A. Koehler (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting 22*(4), 679–688.

Kneib, T. (2003). Bayes-Inferenz in Generalizierten Geoadditiven Gemischten Modellen. Diplomarbeit, University of Munich.

Lewis, P. and J. Stevens (1991). Nonlinear Modeling of Time Series Using Multivariate Adaptive Regression Splines (MARS). *Journal of the American Statistical Association 86*(416).

Lütkepohl, H. (1991). *Introduction to Multiple Time Series Analysis*. Springer, Berlin.

Lütkepohl, H. (2006). *New Introduction to Multiple Time Series Analysis*. Springer.

Lütkepohl, H. and M. Krätzig (2004). *Applied Time Series Econometrics*. Cambridge University Press.

Lutz, R. and P. Bühlmann (2006). Boosting for high-multivariate responses in high-dimensional linear regression. *Statistica Sinica 16*(2), 471–494.

Mallat, S. and Z. Zhang (1993). Matching pursuit in a time-frequency dictionary. *IEEE Transactions on Signal Processing 41*(12), 3397–3415.

Marcellino, M. and C. Schumacher (2007). Factor nowcasting of german gdp with ragged-edge data. a model comparison using midas projections. Technical report, Bundesbank Discussion Paper, Series 1, 34/2007.

Marcellino, M., J. Stock, and M. Watson (2006). A Comparison of Direct and Iterated Multistep AR Methods for Forecasting Macroeconomic Time Series. *Journal of Econometrics 135*(1-2), 499–526.

Nelder, J. and R. Wedderburn (1972). Generalized Linear Models. *Journal of the Royal Statistical Society. Series A (General) 135*(3), 370–384.

Priestley, M. (1980). State-dependent models: A general approach to non-linear time series analysis. *Journal of Time Series Analysis 1*(1), 47–71.

R Development Core Team (2008). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. ISBN 3-900051-07-0.

Reinsch, C. (1967). Smoothing by spline functions. *Numerische Mathematik 10*(3), 177–183.

Robinzonov, N. and K. Wohlrabe (2008). Freedom of Choice in Macroeconomic Forecasting: An Illustration with German Industrial Production and Linear Models. Ifo working paper No. 57, Munich.

Schmid, M. and T. Hothorn (2007). Boosting Additive Models using Component-wise P-Splines. *Department of Statistics: Technical Reports, No.2*.

Schwarz, G. (1978). Estimating the Dimension of a Model. *The Annals of Statistics 6*(2), 461–464.

Shafik, N. and G. Tutz (2007). Boosting Nonlinear Additive Autoregressive Time Series. *Department of Statistics: Technical Reports, No.6*.

Sims, C. (1980). Macroeconomic and Reality. *Econometrica 48*(1), 1–48.

Stock, J. and M. Watson (2003). Forecasting Output and Inflation: The Role of Asset Prices. *Journal of Economic Literature 41*(3), 788–829.

Stone, C., M. Hansen, C. Kooperberg, and Y. Truong (1997). Polynomial splines and their tensor products in extended linear modeling: 1994 Wald memorial lecture. *Ann. Statist 25*(4), 1371–1470.

Teräsvirta, T., D. van Dijk, and M. Medeiros (2005). Linear models, smooth transition autoregressions, and neural networks for forecasting macroeconomic time series: A reexamination. *International Journal of Forecasting 21*(4), 755–774.

Toda, H. and P. Phillips (1993). Vector Autoregressions and Causality. *Econometrica 61*(6), 1367–1393.

Toda, H. and T. Yamamoto (1995). Statistical inference in vector autoregressions with possibly integrated processes. *Journal of Econometrics 66*(1), 225–250.

Tong, H. (1993). *Non-linear Time Series: A Dynamical System Approach*. Oxford University Press.

Tong, H. and K. Lim (1980). Threshold autoregression, limit cycles and cyclical data. *Journal of the Royal Statistical Society 42*(3), 245–292.

Tsay, R. (2005). *Analysis of financial time series*. Wiley, New York.

Tschernig, R. and L. Yang (2000). Nonparametric Lag Selection for Time Series. *Journal of Time Series Analysis 21*(4), 457–487.

Tukey, J. (1977). *Exploratory data analysis.* Addison-Wesley, Reading, MA.

Tutz, G. and H. Binder (2006). Generalized additive modelling with implicit variable selection by likelihood based boosting. *Biometrics 62*(4), 961–71.

Wahba, G. (1990). *Spline Models for Observational Data*. Society for Industrial Mathematics.

Wood, S. (2006). *Generalized Additive Models: An Introduction with R*. Chapman & Hall/CRC.

Wood, S. (2007). *mgcv: GAMs with GCV smoothness estimation and GAMMs by REML/PQL*. R package, version 1.3-29.

Zellner, A. (1962). An Efficient Method of Estimating Seemingly Unrelated Regressions and Tests for Aggregation Bias. *Journal of the American Statistical Association 57*(298), 348–368.

Erklärung

Hiermit versichere ich, dass ich die vorliegende Master-Thesis selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe.

München, den 2. Mai 2008

Nikolay Robinzonov