CHAPTER 1

BOOSTING ALGORITHMS: with an application to bootstrapping multivariate time series

Peter Bühlmann and Roman W. Lutz
Seminar für Statistik, ETH Zürich
CH-8092 Zürich, Switzerland
E-mail: buhlmann@stat.math.ethz.ch

We describe boosting as an estimation method within specified parametric, potentially very high-dimensional models. Besides some short review and historical remarks, we focus on high-dimensional, multivariate linear models (overcomplete dictionaries) for independent and time series data. In particular, we propose a new bootstrap method for high-multivariate, linear time series. We demonstrate its usefulness and we describe relations to some of Bickel's contributions in time series and the bootstrap.

1. Introduction

Since its inception in a practical form in Freund and Schapire (1996), boosting has attracted a lot of attention both in the machine learning and statistics literature. This is in part due to its excellent reputation as a prediction method. The gradient descent view of boosting as articulated in Breiman (1998, 1999) and Friedman et al. (2000) provides a basis for the understanding and new variants of boosting. As an implication, boosting is not only a black-box prediction tool but also an estimation method in specified classes of models, allowing for interpretation of specific model-terms.

We focus here on boosting with the squared error loss, mainly for the multivariate case. Based on it, we propose here a new time series bootstrap method, and we will make a link to some of Bickel's contributions and ideas in time series and the bootstrap. The boosting approach for multivariate linear time series addresses one of the problems which the first author discussed with Peter Bickel a decade ago.

Throughout the paper, we assume that the data are realizations of ran-

dom variables

$$(X_1, Y_1), \ldots, (X_n, Y_n)$$

from a stationary process with p-dimensional predictor variables X_i and q-dimensional response variables Y_i; the jth component of a p-dimensional x will be denoted by $x^{(j)}$. Most often, we will consider $X_i \in \mathbb{R}^p$ and $Y_i \in \mathbb{R}^q$.

1.1. *AdaBoost*

AdaBoost (Freund and Schapire, 1996) is an ensemble algorithm for binary classification with $Y_i \in \{0, 1\}$. It is (still) the most popular boosting algorithm and it exhibits a remarkable performance in numerous empirical studies. It works by specifying a base classifier ("weak learner") which is repeatedly applied to iteratively reweighted data, yielding an ensemble of classifiers $\hat{g}^{[1]}(\cdot), \ldots, \hat{g}^{[m]}(\cdot)$, where each $\hat{g}^{[k]}(\cdot) : \mathbb{R}^p \to \{0, 1\}$. That is:

$$\begin{array}{lcl} \text{reweighted data 1} & \overset{\text{base procedure}}{\longrightarrow} & \hat{g}^{[1]}(\cdot) \\ \text{reweighted data 2} & \overset{\text{base procedure}}{\longrightarrow} & \hat{g}^{[2]}(\cdot) \\ \ldots & & \ldots \\ \text{reweighted data m} & \overset{\text{base procedure}}{\longrightarrow} & \hat{g}^{[m]}(\cdot) \end{array}$$

Finally, the AdaBoost classifier is

$$\hat{C}^{[m]}_{AdaBoost}(\cdot) = \text{sign}(\sum_{j=1}^{m} c_j \hat{g}^{[m]}(\cdot)), \tag{1}$$

where c_j are linear combination weights, depending on the in-sample performance of the classifier $\hat{g}^{[j]}(\cdot)$. Thus, the AdaBoost classifier is a weighted majority vote among the ensemble of individual classifiers. A key issue is how to reweigh the original data; once we have reweighted data, one simply applies the base procedure to it as if it would be the original dataset. A statistically motivated description can be found in Friedman et al. (2000).

From the description above, AdaBoost involves three specifications: (i) the base procedure ("weak learner"), (ii) the construction of reweighted data, (iii) the size of the ensemble m. Regarding (i), most popular are classification trees; issue (ii) is defined by the AdaBoost description (cf. Friedman et al. (2000)); and the value m in (iii) is a simple one-dimensional tuning parameter.

2. Boosting and functional gradient descent

Breiman (1998, 1999) showed that the somewhat mysterious AdaBoost algorithm can be represented as a steepest descent algorithm in function space which we call functional gradient descent (FGD). This great result opened the door to use boosting in other settings than classification. For simplicity, we focus first on the univariate case with 1-dimensional response variables Y_i ($q = 1$).

In the sequel, boosting and functional gradient descent (FGD) are used as a terminology for the same method or algorithm. The goal is to estimate a function

$$f_0(\cdot) = \text{argmin}_{f(\cdot)} \mathbb{E}[\rho(Y, f(X))] \tag{2}$$

where $\rho(\cdot, \cdot)$ is a real-valued loss function which is typically convex with respect to the second argument. The function class which we minimize over is not of interest for the moment and hence notationally omitted.

Examples of loss functions and their population minimizers are given in the following table; each case corresponds to a boosting algorithm, as explained in section 2.1.

	L_2Boosting	LogitBoost	AdaBoost
spaces	$y \in \mathbb{R}, f \in \mathbb{R}$	$y \in \{0,1\}, f \in \mathbb{R}$	$y \in \{0,1\}, f \in \mathbb{R}$
$\rho(y, f)$	$\lvert y - f\rvert^2$	$-\log_2(1 + \exp(-2(2y-1)f))$	$\exp(-(2y-1)f)$
f_0	$\mathbb{E}[Y\lvert X = x]$	$\frac{1}{2}\log\left(\frac{p(x)}{1-p(x)}\right)$	$\frac{1}{2}\log\left(\frac{p(x)}{1-p(x)}\right)$

For the last row, $p(x) = \mathbb{P}[Y = 1|X = x]$.

2.1. *The generic boosting algorithm*

Having specified a loss function $\rho(\cdot, \cdot)$ we pursue some sort of empirical minimization: instead of (2), we do some constrained minimization of the empirical risk

$$n^{1-} \sum_{i=1}^{n} \rho(Y_i, f(X_i)) \tag{3}$$

with respect to $f(\cdot)$. We emphasize here that the constraints will enter non-implicitly in terms of a (boosting) algorithm. This is in contrast to empirical risk minimization over suitable (small enough) function classes or by pursuing penalized empirical risk minimization using e.g. ℓ^2- or ℓ^1-penalties.

2.1.1. *The base procedure ("weak learner")*

Boosting or FGD pursues constrained minimization of (3) by iterative steepest descent in function space. To explain this, we elaborate a bit more on the notion of a base procedure, often called the "weak learner" in the machine learning community. Based on some (pseudo-) response variables $\mathbf{U} = U_1, \ldots, U_n$ and predictor variables $\mathbf{X} = X_1, \ldots, X_n$, the base procedure yields a function estimate

$$\hat{g}(\cdot) = \hat{g}_{(\mathbf{U},\mathbf{X})}(\cdot) : \mathbb{R}^p \to \mathbb{R}.$$

Note that we focus here on function estimates with values in $\mathbb{R}$, rather than classifiers with values in $\{0, 1\}$ as described in section 1.1.

Typically, the function estimate $\hat{g}(x)$ can be thought as an approximation of $\mathbb{E}[U|X = x]$. For example, the base procedure could be a nonparametric kernel estimator (if p is small) or a nonparametric statistical method with some structural restrictions (for $p \geq 2$) such as a regression tree (or class-probability estimates from a classification tree).

Componentwise linear least squares: For cases with $p \gg n$, a useful base procedure is componentwise linear least squares:

$$\hat{g}(x) = \hat{\gamma}_{\hat{\mathcal{S}}} x^{(\hat{\mathcal{S}})},$$
$$\hat{\gamma}_j = \frac{\sum_{i=1}^n U_i X_i^{(j)}}{\sum_{i=1}^n (X_i^{(j)})^2} \ (j = 1, \ldots, p), \quad \hat{\mathcal{S}} = \text{argmin}_{1 \leq j \leq p} \sum_{i=1}^n (U_i - \hat{\gamma}_j X_i^{(j)})^2.$$

This base procedure fits a linear regression with the one predictor variable which reduces residual sum of squares most.

2.1.2. *The algorithm*

The generic FGD or boosting algorithm is as follows.

Generic FGD algorithm

Step 1. Initialize $\hat{f}^{[0]} \equiv 0$. Set $m = 0$.

Step 2. Increase m by 1.
Compute the negative gradient and evaluate it at $f = \hat{f}^{[m-1]}(X_i)$:

$$U_i = -\frac{\partial}{\partial f} \rho(Y, f)|_{f = \hat{f}^{[m-1]}(X_i)}, \ i = 1, \ldots, n.$$

Step 3. Fit negative gradient vector $U_1, \ldots, U_n$ by using the base procedure, yielding the estimated function

$$\hat{g}^{[m]}(\cdot) = \hat{g}_{\mathbf{U},\mathbf{X}}(\cdot) : \mathbb{R}^p \to \mathbb{R}.$$

The function estimate $\hat{g}^{[m]}(\cdot)$ may be thought of as an approximation of the negative gradient vector $(U_1, \ldots, U_n)$.

Step 4. Do a one-dimensional numerical line-search for the best step-size:

$$\hat{\delta}^{[m]} = \text{argmin}_\delta \sum_{i=1}^{n} \rho(Y_i, \hat{f}^{[m-1]}(X_i) + \delta \hat{g}^{[m]}(X_i)).$$

Step 5. Up-date $\hat{f}^{[m]} = \hat{f}^{[m-1]}(\cdot) + \nu \cdot \hat{\delta}^{[m]} \hat{g}^{[m]}(\cdot)$ where $0 < \nu \leq 1$ is reducing the step-length for following the approximated negative gradient.

Step 6. Iterate Steps 2-5 until $m = m_{stop}$ is reached for some specified stopping iteration m_{stop}.

The factor ν which reduces the step-length in Step 5 should be chosen "small": our proposal for a default is $\nu = 0.1$. The FGD algorithm does depend on this factor ν, but its choice is not very crucial as long as it is taken to be "small". On the other hand, the stopping iteration m_{stop} is an important tuning parameter of boosting or FGD. Data-driven choices can be done by using cross-validation schemes; computationally much more attractive are internal estimates from information criteria such as AIC, see section 3.3.

By definition, the generic FGD algorithm yields a linear combination of base procedure estimates:

$$\hat{f}^{[m_{stop}]}(\cdot) = \nu \sum_{m=1}^{m_{stop}} \hat{g}^{[m]}(\cdot)$$

which can be interpreted as an estimate from an ensemble scheme, i.e. the final estimator is an average of individual estimates from the base procedure, similar to the formula for AdaBoost in (1). Thus, the boosting solution implies the following constraint for minimizing the empirical risk in (3): it is a linear combination of fits from the base procedure; in addition, it will be a regularized fit of such linear combinations, see section 2.2.2.

2.2. *Boosting with the squared error loss: : L_2Boosting*

When using the squared error loss $\rho(y, f) = |y - f|^2$, the generic FGD algorithm above takes the simple form of refitting the base procedure to residuals of the previous iteration, cf. Friedman (2001).

L_2Boosting

Step 1 (initialization and first estimate). Given data $\{(X_i, Y_i); i = 1, \ldots, n\}$, fit the base procedure

$$\hat{f}^{[1]}(\cdot) = \nu \cdot \hat{g}_{(\mathbf{Y},\mathbf{X})}(\cdot).$$

Set $m = 1$.

Step 2. Increase m by 1.
Compute residuals $U_i = Y_i - \hat{f}^{[m-1]}(X_i)$ $(i = 1, \ldots, n)$ and fit the base procedure to the current residuals. The fit is denoted by $\hat{g}^{[m]}(\cdot) = \hat{g}_{(\mathbf{U},\mathbf{X})}(\cdot)$. Up-date

$$\hat{f}^{[m]}(\cdot) = \hat{f}^{[m-1]}(\cdot) + \nu \cdot \hat{g}^{[m]}(\cdot),$$

where $0 < \nu \leq 1$ is a pre-specified step-size parameter.
(The line-search $\hat{\delta}^{[m]}$ is omitted; in fact, $\hat{\delta}^{[m]} = 1$ if the base procedure is fitted by least squares).

Step 3 (iteration). Repeat Step 2 until some stopping value m_{stop} for the number of iterations is reached.

2.2.1. *A glimpse of history*

With $m = 2$ (one boosting step), L_2Boosting has already been proposed by Tukey (1977) under the name "twicing".

C.F. Gauss

R.V. Southwell

J.W. Tukey

L_2Boosting with the componentwise least squares base procedure for a fixed collection of basis functions (and using $\nu = 1$) coincides with the matching pursuit algorithm of Mallat and Zhang (1993). Matching pursuit is also

known in computational mathematics under the name of "(weak) greedy algorithm" (DeVore and Temlyakov, 1996; Temlyakov, 2000). All these methods are also known under the keyword "Gauss-Southwell algorithm" whose origin goes back to Carl Friedrich Gauss, the "Princeps Mathematicorum", and to Sir Richard Southwell. While Gauss is famous, Southwell is less known. Sir Richard Southwell was a faculty member of the Engineering School of Oxford University. In the early 1940's, he developed a powerful iterative procedure, known as the relaxation method, which was successfully applied to a large variety of problems in engineering and physical science. Quoting from a newsletter of the Society of Oxford University Engineers: "Southwell was a first class lecturer and attendance at his lectures was a pleasure".

Gauss realized that a linear system of equations

$$A\beta = b,\ A \in \mathbb{R}^{n \times p},\ \beta \in \mathbb{R}^p,\ b \in \mathbb{R}^n$$

can be solved for β by iteratively pursuing the solution for one component of β while keeping all others fixed: the iteration goes over the component indices of β: $j = 1, 2, \ldots, p, 1, 2, \ldots, p, 1, 2, \ldots$ This is known as the Gauss-Seidel algorithm; the same idea is also used in backfitting (cf. Buja et al. (1989)), and Bickel et al. (1993) describe some of its (statistical) properties. Southwell's contribution has been to alter the way the iterations are done: instead of going systematically through the component-indices of β as described above, he argued for the greedy version: select the component such that a suitable error-norm is minimized. In our context, this translates to select the predictor variable such that residual sum of squares is minimized which is exactly what the componentwise linear least squares base procedure does.

Tukey's (1977) twicing seems to be the first proposal to formulate the Gauss-Southwell idea in the context of a nonparametric smoothing estimator, beyond the framework of linear models (dictionaries of basis functions).

2.2.2. *L_2Boosting, Lasso and LARS*

Efron et al. (2004) made an intriguing connection between L_2Boosting with componentwise linear least squares and the Lasso (Tibshirani, 1996) which is an ℓ^1-penalized least squares method for linear regression. They consider a version of L_2Boosting, called forward stagewise linear regression (FSLR), and they show that FSLR with infinitesimally small step-sizes produces a set of solutions which is approximately equivalent to the set of Lasso

solutions when varying the regularization parameter in Lasso (see also (4) below). The approximate equivalence is derived by representing FSLR and Lasso as two different modifications of their computationally clever least angle regression (LARS) algorithm. In special cases where the design matrix satisfies a "positive cone condition", FSLR, Lasso and LARS all coincide (Efron et al., 2004; p.425).

As Efron et al. (2004; sec. 8) write, their LARS procedure is not directly applicable to more general base procedures (e.g. regression trees) and to problems which we will present in section 3.

During the iterations of L_2Boosting, we get an interesting set of solutions $\{\hat{f}^{[m]}(\cdot); m = 1, 2, \ldots\}$ and corresponding regression coefficients $\{\hat{\beta}^{[m]} \in \mathbb{R}^p;\ m = 1, 2, \ldots\}$. Heuristically, due to the results in Efron et al. (2004), it is "similar" to the set of Lasso solutions $\{\hat{\beta}_\lambda \in \mathbb{R}^p;\ \lambda \in \mathbb{R}^+\}$ when varying the penalty parameter λ, where

$$\hat{\beta}_\lambda = \text{argmin}_{\beta \in \mathbb{R}^p} \sum_{i=1}^{n} (Y_i - \sum_{j=1}^{p} \beta^{(j)} X_i^{(j)})^2 + \lambda \sum_{j=1}^{p} |\beta^{(j)}|. \tag{4}$$

Computing the set of boosting solutions $\{\hat{f}^{[m]};\ m = 1, 2 \ldots\}$ is computationally quite cheap since every boosting step is typically simple: hence, estimating a good stopping iteration m_{stop} via e.g. cross-validation is computationally attractive, and the computational gain is even more impressive when using an internal information criterion such as AIC, see section 3.3. (Of course, for the special case of linear regression, LARS (Efron et al., 2004) is computationally even more efficient than boosting). On the other hand, regularized boosting with e.g. an ℓ^1 penalty term (cf. Lugosi and Vayatis (2004)) requires for tuning via cross-validation (selecting the penalty parameter) that the whole algorithm is run repeatedly for many candidate penalty parameters and all training-/test-sets from cross-validation.

2.3. *A selective review of theoretical results for boosting*

The difficulty to analyse some boosting method lies in the fact that one has to understand the statistical properties of an algorithm. This is in contrast to theoretical analysis of more explicit estimators such as ℓ^1-penalized versions of boosting or the Lasso. Regarding the latter, in case of linear regression, we have an explicit estimation functional as in (4), and the theoretical analysis is *not* considering the numerical algorithm for computing the solution of the convex minimization above.

Consistency results for boosting algorithms with early stopping as described in section 2.1 have been given by Jiang (2004) for AdaBoost, Bickel and Ritov (2004) for general loss functions, Zhang and Yu (2003) for general loss functions, and Bühlmann (2004) for L_2Boosting; Bühlmann and Yu (2003) have shown minimax optimality of L_2Boosting in the toy problem of one-dimensional curve estimation. There are quite a few other theoretical analyses of boosting-type methods which use an ℓ^1-penalty instead of early stopping for regularization. We have outlined in the last paragraph of section 2.2.2 a computational advantage of early-stopped boosting which distinguishes itself – as a method – from ℓ^1-regularized boosting.

3. L_2Boosting for high-dimensional multivariate regression

We are describing here a boosting method for multivariate data, including seemingly unrelated (Zellner, 1962; 1963) structures. Consider the multivariate linear regression model with n observations of a q-dimensional response and a p-dimensional predictor. In matrix notation:

$$\mathbf{Y} = \mathbf{X}\mathbf{B} + \mathbf{E}, \tag{5}$$

with $\mathbf{Y} \in \mathbb{R}^{n \times q}$, $\mathbf{X} \in \mathbb{R}^{n \times p}$, $\mathbf{B} \in \mathbb{R}^{p \times q}$ and $\mathbf{E} \in \mathbb{R}^{n \times q}$. The multivariate case requires an extension of the notation. We denote by $\mathbf{Y}_i \in \mathbb{R}^q$ the i-th sample point of the response variable (row-vector of $\mathbf{Y}$) and by $\mathbf{Y}^{(k)} \in \mathbb{R}^n$ the k-th component of the response (column-vector of $\mathbf{Y}$); and analogously for $\mathbf{X}$, $\mathbf{B}$ and $\mathbf{E}$. For each $\mathbf{Y}^{(k)}$, $k = 1, \ldots, q$, we have a univariate regression model with the predictor matrix $\mathbf{X}$ and the coefficient vector $\mathbf{B}^{(k)}$. For the row-vectors of the error matrix $\mathbf{E}_i$, $i = 1, \ldots, n$, we assume $\mathbf{E}_i$ i.i.d. $\sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$. Assuming that $\mathbf{X}$ has full rank p (in particular $p \leq n$), the ordinary least squares estimator exists:

$$\hat{\mathbf{B}}_{OLS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

which equals the ordinary least squares estimate for each of the q univariate regressions. In particular, it is independent of $\mathbf{\Sigma}$.

3.1. *The implementing loss function*

In many examples, p and q are large relative to sample size n and we would like to do a sparse model fit. We construct a boosting method by specifying a loss function and a base procedure. Regarding the former, we use the

Gaussian negative log-likelihood:

$$-\ell_{\mathbf{\Sigma}}(\mathbf{B}) = -\log((2\pi)^{nq/2}\det(\mathbf{\Sigma})^{n/2}) + \frac{1}{2}\sum_{i=1}^{n}(\mathbf{Y}_i^T - \mathbf{X}_i^T\mathbf{B})\mathbf{\Sigma}^{-1}(\mathbf{Y}_i^T - \mathbf{X}_i^T\mathbf{B})^T.$$

The first term on the right-hand side is a constant (w.r.t. $\mathbf{B}$): we drop it and with $\mathbf{\Gamma}^{-1} = \mathbf{\Sigma}^{-1}$, the loss function becomes

$$L(\mathbf{B}) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{Y}_i^T - \mathbf{X}_i^T\mathbf{B})\mathbf{\Gamma}^{-1}(\mathbf{Y}_i^T - \mathbf{X}_i^T\mathbf{B})^T. \tag{6}$$

We distinguish here between $\mathbf{\Gamma}$ and $\mathbf{\Sigma} = \mathrm{Cov}(\mathbf{E}_i)$: $\mathbf{\Gamma}$ is the implementing covariance matrix for the loss function. We may use for it an estimate of $\mathbf{\Sigma}$ (e.g. from another model-fit such as univariate boosting for each response separately) or we can choose something simpler, e.g. $\mathbf{\Gamma} = \mathbf{I}_q$ (in particular if q is large). In case of the latter, the loss function may still be reasonable (if the q components are on the same scale) and the statement in Theorem 1 is then with $\mathbf{\Gamma} = \mathbf{I}_q$.

The right hand-side can be written as $\sum_{i=1}^{n}\rho(\mathbf{Y}_i^T, \mathbf{B})$ (implicitly involving $\mathbf{\Gamma}^{-1}$ and $\mathbf{X}_i^T$), very much like in (3).

The maximum likelihood estimator of $\mathbf{B}$ is the same as the OLS solution and is therefore independent of $\mathbf{\Sigma}$. The covariance matrix $\mathbf{\Sigma}$ becomes only relevant in the seemingly unrelated regressions (SUR; Zellner (1962, 1963)) where some covariates influence only a few components of the q-dimensional response.

3.2. *The base procedure*

The input data for the base procedure is the design matrix $\mathbf{X}$ and a pseudo-response matrix $\mathbf{U} \in \mathbb{R}^{n\times q}$ (not necessarily equal to $\mathbf{Y}$). The base procedure fits the linear least squares regression with one selected covariate (column of $\mathbf{X}$) and one selected pseudo-response (column of $\mathbf{U}$) so that the loss function in (6), with $\mathbf{U}$ instead of $\mathbf{Y}$, is reduced most.

Thus, the base procedure fits one selected matrix element of $\mathbf{B}$:

$$\begin{aligned}
&\hat{\mathbf{B}}_{jk} = 0,\ (jk) \neq (\hat{s}\hat{t}), \quad \hat{\mathbf{B}}_{\hat{s}\hat{t}} = \hat{b}_{\hat{s}\hat{t}},\\
&\hat{b}_{jk} = \frac{\sum_{v=1}^{q}(\mathbf{U}^{(v)})^T\mathbf{X}^{(j)}\mathbf{\Gamma}_{vk}^{-1}}{(\mathbf{X}^{(j)})^T\mathbf{X}^{(j)}\mathbf{\Gamma}_{kk}^{-1}},\\
&(\hat{s}\hat{t}) = \mathrm{argmin}_{1\le j\le p, 1\le k\le q}\{L(\mathbf{B}); \mathbf{B}_{jk} = \hat{b}_{jk}, \mathbf{B}_{rs} = 0\ (rs \neq jk)\}\\
&\qquad = \mathrm{argmax}_{1\le j\le p, 1\le k\le q}\frac{\left(\sum_{v=1}^{q}(\mathbf{U}^{(v)})^T\mathbf{X}^{(j)}\mathbf{\Gamma}_{kv}^{-1}\right)^2}{(\mathbf{X}^{(j)})^T\mathbf{X}^{(j)}\mathbf{\Gamma}_{kk}^{-1}}.
\end{aligned} \tag{7}$$

Corresponding to the parameter estimate, there is a function estimate $\hat{\mathbf{g}}(\cdot) : \mathbb{R}^p \to \mathbb{R}^q$ defined as follows:

$$(\hat{\mathbf{g}})_k(\mathbf{x}) = \begin{cases} \hat{b}_{\hat{s}\hat{t}} x^{(\hat{s})} & \text{if } k = \hat{t} \\ 0 & \text{if } k \neq \hat{t}, \end{cases} \quad k = 1, \ldots, q.$$

From (7) we see that the coefficient $\hat{b}_{jk}$ is not only influenced by the k-th response but also by other responses, depending on the partial correlations of the errors (via $\mathbf{\Gamma}^{-1}$ if $\mathbf{\Gamma}$ is a reasonable estimate for $\mathbf{\Sigma}$) and the correlations of the other responses with the j-th covariate (i.e. $(\mathbf{U}^{(v)})^T \mathbf{X}^{(j)}$).

3.3. *The algorithm*

Our multivariate L_2Boosting algorithm is defined as follows.

Multivariate L_2Boosting
with componentwise linear least squares

Step 1 (initialization and first estimate). Fit the base procedure

$$\hat{\mathbf{f}}^{[1]}(\cdot) = \nu \cdot \hat{\mathbf{g}}_{\mathbf{Y},\mathbf{X}}(\cdot),$$

where $\hat{\mathbf{g}}_{\mathbf{Y},\mathbf{X}}(\cdot)$ is the base procedure based on data $\mathbf{Y}, \mathbf{X}$.
Set $m = 1$.

Step 2. Increase m by 1.
Compute current residuals

$$\mathbf{U}_i^{[m]} = \mathbf{Y}_i - \hat{\mathbf{f}}^{[m-1]}(\mathbf{X}_i) \ (i = 1, \ldots, n)$$

and fit the base procedure from (7) to $\mathbf{U}$ and $\mathbf{X}$. The fit is denoted by $\hat{\mathbf{g}}^{[m]}(\cdot)$.
Up-date

$$\hat{\mathbf{f}}^{[m]}(\cdot) = \hat{\mathbf{f}}^{[m-1]}(\cdot) + \nu \cdot \hat{\mathbf{g}}^{[m]}(\cdot), \ 0 < \nu < 1.$$

Step 3 (iteration). Repeat Step 2 until a stopping iteration m_{stop} is met.

We also obtain a sequence of estimates $\hat{\mathbf{B}}^{[m]}$ which correspond to $\hat{\mathbf{f}}^{[m]}(\mathbf{x}) = (\hat{\mathbf{B}}^m)^T \mathbf{x}$, $\mathbf{x} \in \mathbb{R}^p$. Multivariate L_2Boosting with componentwise linear least squares resembles very much the univariate L_2Boosting analogue described in section 2.2. The difference is that we search in addition for the best response-component $k \in \{1, \ldots, q\}$ to improve the loss function $L(\cdot)$. As in the univariate case, the step-size ν should be chosen small, e.g. $\nu = 0.1$.

The number of iterations m_{stop} is a tuning parameter and can be estimated by e.g. cross validation or an AIC criterion. The latter is computationally attractive. It relies on the fact that the base procedure in (7) involves a linear hat operator and an optimization over the best pair of indices $(\hat{s}\hat{t})$. Then, the boosting fit at iteration m can be represented as a hat operator $\mathcal{B}^{[m]} : \mathbb{R}^{nq} \to \mathbb{R}^{nq}$ which maps the response $\mathbf{Y}$ to the fitted values $\hat{\mathbf{Y}}$. Neglecting the fact that a search over the best pair of indices $(\hat{s}\hat{t})$ has taken place in the repeated use of the base procedure in (7), $\mathcal{B}^{[m]}$ becomes a linear operator and its trace can serve as the number of degrees of freedom (d.f.):

$$\text{d.f.} = \text{trace}(\mathcal{B}^{[m]}).$$

With this notion of degrees of freedom, one can then use the corrected AIC (AIC_c), cf. Hurvich et al. (1998), to estimate the optimal stopping iteration m_{stop}. Using model selection criteria for stopping the boosting iterations has been successfully demonstrated in Bühlmann (2004) and Bühlmann and Yu (2005); in the context of multivariate boosting, the details are described in Lutz and Bühlmann (2005).

3.4. *Properties of multivariate L_2Boosting*

We summarize here some of the results from Lutz and Bühlmann (2005).

3.4.1. *An asymptotic result*

First, we describe a consistency result for multivariate L_2Boosting in linear regression where the number of predictors $p = p_n$ and the dimension of the response $q = q_n$ are allowed to grow very fast as sample size n increases. Consider the model

$$\begin{aligned} &\mathbf{Y}_i = \mathbf{f}(\mathbf{X}_i) + \mathbf{E}_i,\ i = 1, \ldots, n, \qquad \mathbf{Y}_i, \mathbf{E}_i \in \mathbb{R}^{q_n},\ \mathbf{X}_i \in \mathbb{R}^{p_n}, \\ &\mathbf{f}(\mathbf{x}) = \mathbf{B}^T\mathbf{x}, \qquad \mathbf{B} \in \mathbb{R}^{p_n \times q_n},\ \mathbf{x} \in \mathbb{R}^{p_n}, \\ &\mathbf{X}_i \text{ i.i.d. and } \mathbf{E}_i \text{ i.i.d. with } \mathbb{E}[\mathbf{E}_i] = \mathbf{0} \text{ and } \text{Cov}(\mathbf{E}_i) = \mathbf{\Sigma}. \end{aligned} \tag{8}$$

Because p_n and q_n are allowed to grow with n, also the predictors and the responses depend on n, but we often ignore this notationally. To identify the magnitude of $b_{jk} = b_{jk,n} = \mathbf{B}_{n;jk}$, we assume $\mathbb{E}|\mathbf{X}_1^{(j)}|^2 = 1,\ j = 1, \ldots, p_n$.

We make the following assumptions:

(A1) The dimension of the predictor and the response in model (8) satisfies $p_n = O(\exp(Cn^{1-\xi}))$, $q_n = O(\exp(Cn^{1-\xi}))$ $(n \to \infty)$, for some $0 < \xi < 1, 0 < C < \infty$.

(A2) $\sup_{n\in\mathbb{N}} \sum_{j=1}^{p_n} \sum_{k=1}^{q_n} |b_{jk,n}| < \infty$.
(A3) For the implementing $\mathbf{\Gamma}^{-1}$ in (6):
$\sup_{n\in\mathbb{N},1\le k\le q_n} \sum_{\ell=1}^{q_n} |\mathbf{\Gamma}^{-1}_{k\ell,n}| < \infty, \quad \inf_{n\in\mathbb{N},1\le k\le q_n} \mathbf{\Gamma}^{-1}_{kk} > 0.$
(A4) $\sup_{1\le j\le p_n} \|(\mathbf{X}_1^{(j)}\|_\infty < \infty$, where $\|x\|_\infty = \sup_{\omega\in\Omega} |x(\omega)|$ (Ω denotes the underlying probability space).
(A5) $\sup_{1\le k\le q_n} \mathbb{E}|\mathbf{E}_1^{(k)}|^s < \infty$ for some $s > 4/\xi$ with ξ from (A1).

Assumption (A1) allows for very large predictor and response dimensions relative to sample size n. Assumption (A2) is an ℓ^1-norm sparseness condition. Assumption (A4) could be weakened to existence of sufficiently high moments, at the expense of a slower growth in (A1) which could still be as fast as $O(n^\beta)$ for any $0 < \beta < \infty$ (see also section 4).

Theorem 1: Consider the model (8) satisfying (A1)-(A5). Then, the multivariate L_2Boosting estimate $\hat{\mathbf{f}}^{[m_n]}$ with the component-wise linear learner from (7) satisfies: for some sequence $(m_n)_{n\in\mathcal{N}}$ with $m_n \to \infty$ ($n \to \infty$) sufficiently slowly,

$$\mathbb{E}_{\mathbf{x}} \left| \left(\hat{\mathbf{f}}^{[m_n]}(\mathbf{x}) - \mathbf{f}(\mathbf{x})\right)^T \mathbf{\Gamma}^{-1} \left(\hat{\mathbf{f}}^{[m_n]}(\mathbf{x}) - \mathbf{f}(\mathbf{x})\right) \right| = o_p(1) \ (n \to \infty),$$

where $\mathbf{x}$ denotes a new observation, independent of and with the same distribution as the training data.

A proof is given in Lutz and Bühlmann (2005). Theorem 1 says that multivariate L_2Boosting recovers the true sparse regression function even if the dimensions of the predictor or response grow essentially exponentially with sample size n. For the univariate linear model analogue, such a result has been shown for L_2Boosting in Bühlmann (2004) and for the Lasso in Greenshtein and Ritov (2004).

3.4.2. *A summary of some empirical results*

In Lutz and Bühlmann (2005), multivariate L_2Boosting has been compared with multivariate forward variable selection and with individual L_2Boosting where each of the response components are fitted separately. A crude summary is as follows.

(i) Multivariate forward variable selection was often worse than boosting. The reason can be attributed to the fact that it often pays off to use a method which does variable selection *and* shrinkage such as boosting or Lasso. This has been observed in various contexts, cf. Tibshirani (1996) for

the Lasso, Friedman (2001) and Bühlmann (2004) for boosting. Examples where forward variable selection perform better than boosting are of the following type: only very few effective predictor variables contributing a strong signal and many non-effective predictors with no influence on the responses: e.g., the coefficient matrix $\mathbf{B}$ has only very few rows with large entries and all others are zero.

(ii) Multivariate L_2Boosting was found to be better than individual boosting if the errors are highly correlated, i.e. $\mathbf{\Sigma}$ is (strongly) non-diagonal. In contrast to individual boosting and other individual methods like OLS for multivariate regression, multivariate L_2Boosting allows to take an estimate of $\mathbf{\Sigma}$ into account via the loss function in (6). For three real data sets, where we do not know whether $\mathbf{\Sigma}$ is strongly non-diagonal, multivariate boosting and individual boosting were comparable in terms of a cross-validation score; larger differences may be masked by a substantial noise variance which enters because we consider differences $\hat{\mathbf{Y}} - \mathbf{Y} = (\hat{\mathbf{f}} - \mathbf{f}) - \mathbf{E}$, whereas for simulated data-sets we measure the discrepancy $\hat{\mathbf{f}} - \mathbf{f}$ directly.

4. L_2Boosting for multivariate linear time series

Obviously, the boosting method from section 3 can be used for vector autoregressive processes

$$\mathbf{X}_t = \sum_{j=1}^{p} \mathbf{A_j} \mathbf{X}_{t-j} + \mathbf{E}_t, \quad t \in \mathbb{Z}, \tag{9}$$

where $\mathbf{X}_t \in \mathbb{R}^q$ is the q-dimensional observation at time t, $\mathbf{A_j} \in \mathbb{R}^{q \times q}$ and $\mathbf{E}_t \in \mathbb{R}^q$ i.i.d. with $\mathbb{E}[\mathbf{E}_t] = \mathbf{0}$ and $\mathrm{Cov}(\mathbf{E}_t) = \mathbf{\Sigma}$. The model is stationary and causal if all roots of $\det(\mathbf{I} - \sum_{j=1}^{p} \mathbf{A_j} z^j)$ ($z \in \mathbb{C}$) are greater than one in absolute value.

For observations $\mathbf{X}_t$ ($t = 1, \ldots, n$), the equation in (9) can be written as a multivariate regression model as in (5) with $\mathbf{Y} = [\mathbf{X}_{p+1}, \ldots, \mathbf{X}_n]^T \in \mathbb{R}^{(n-p) \times q}$, $\mathbf{B} = [\mathbf{A_1}, \ldots, \mathbf{A_p}]^T \in \mathbb{R}^{qp \times q}$ and $\mathbf{X} \in \mathbb{R}^{(n-p) \times qp}$ the corresponding design matrix. The consistency result from Theorem 1 carries over to the time series case. We consider the following $q = q_n$-dimensional VAR(∞) model:

$$\mathbf{X}_t = \sum_{j=1}^{\infty} \mathbf{A_j} \mathbf{X}_{t-j} + \mathbf{E}_t, \quad t \in \mathbb{Z}, \tag{10}$$

with $\mathbf{E}_t \in \mathbb{R}^{q_n}$ i.i.d. with $\mathbb{E}[\mathbf{E}_t] = \mathbf{0}$, $\mathrm{Cov}(\mathbf{E}_t) = \mathbf{\Sigma}$ and $\mathbf{E}_t$ independent of $\{\mathbf{X}_s;\ s < t\}$. Again, we ignore notationally that the model and its terms

depend on n due to the growing dimension q_n. Assume that:

(B1) $\{\mathbf{X}_t\}_{t\in\mathbb{Z}}$ in (10) is strictly stationary and α-mixing with mixing coefficients $\alpha(\cdot) = \alpha_n(\cdot)$.
(B2) The dimension satisfies: $q = q_n = O(n^\beta)$ for some $0 < \beta < \infty$.
(B3) $\sup_{n\in\mathbb{N}} \sum_{j=1}^{\infty} \sum_{k,r=1}^{q_n} |a_{k,r;j,n}| < \infty, \quad a_{k,r;j,n} = (\mathbf{A}_{j;n})_{kr}$.
(B4) The mixing coefficients and moments satisfy: for some $s \in \mathbb{N}$ with $s > 2(1+\beta) - 2$ (β as in (B2)) and $\gamma > 0$

$$\sum_{k=1}^{\infty} (k+1)^{s-1} \alpha_n(k)^{\gamma/(2s+\gamma)} < \infty,$$

$$\sup_{1\le j\le q_n, n\in\mathbb{N}} \mathbb{E}|\mathbf{X}_t^{(j)}|^{4s+2\gamma} < \infty, \quad \sup_{1\le j\le q_n, n\in\mathbb{N}} \mathbb{E}|\mathbf{E}_t^{(j)}|^{2s+\gamma} < \infty.$$

Theorem 2: Assume the model (10), satisfying the assumptions (B1)-(B4), and require that (A3) holds. Consider multivariate L_2Boosting with componentwise linear least squares (as in section 3) using $p = p_n$ lagged variables (as in model (9)) with $p_n \to \infty$, $p_n = O(n^{1-\kappa})$ $(n \to \infty)$, where $2(1+\beta)/(s+2) < \kappa < 1$. Then, the assertion from Theorem 1 holds with $f(\mathbf{x}) = \sum_{j=1}^{\infty} \mathbf{A}_j \mathbf{x}_{t-j}$, $\hat{\mathbf{F}}^{(m_n)}(\mathbf{x}) = \sum_{j=1}^{p_n} \hat{\mathbf{A}}_j^{(m_n)} \mathbf{x}_{t-j}$ and $\mathbf{x}$ a new realization from (10), independent from the training data.

A proof is given in Lutz and Bühlmann (2005). Multivariate L_2Boosting is thus a consistent method for vector autoregressive models of infinite order. Assuming invertibility of the autoregressive representation, i.e. $\det(\mathbf{I} - \sum_{j=1}^{\infty} \mathbf{A_j} z^j) \neq 0$ for $|z| \le 1$, the range of consistency of multivariate L_2Boosting covers all causal, linear multivariate processes.

4.1. *The modification for stationary model fitting*

It is a desirable feature that a time series fit yields a stationary model. This property holds e.g. for the Yule-Walker estimator in autoregressive models (cf. Brockwell and Davis (1991)) and it allows to simulate stationary processes from the fitted model. The latter will be important for bootstrapping stationary time series data, see section 4.3. Ensuring a stationary model fit is not so easily possible with the Lasso or Ridge regression which are well known regularization methods for regression.

With boosting, it is straightforward to modify the algorithm so that stationarity of the fitted model is ensured. In every iteration, we only allow an up-date $\hat{f}^{[m]}(\cdot)$ which is stationary. This can be achieved as follows.

Modified multivariate L_2Boosting for stationary model fitting

We modify Step 2 of the algorithm in section 3.3.

Modified Step 2 (a). Check if $\hat{f}^{[m]}(\cdot)$ corresponds to a stationary process. To do so, denote the corresponding estimates by $\hat{\mathbf{A}}_j^{[m]}$ and consider the condition:

$$\det(I - \hat{\mathbf{A}}_1^{[m]} z - \ldots - \hat{\mathbf{A}}_p^{[m]} z^p) \neq 0 \text{ for } |z| \leq 1. \tag{11}$$

If (11) holds, then accept this $\hat{f}^{[m]}(\cdot)$ and proceed to Step 3 in the multivariate L_2Boosting algorithm from section 3.3.
If (11) fails, then go to the modified Step 2 (b) below.

Modified Step 2 (b). Consider the set $V = (\{1, \ldots, p\} \times \{1, \ldots q\}) \setminus (\hat{s}, \hat{t})$, where $(\hat{s}, \hat{t})$ have been chosen by the base procedure from (7) in Step 2, yielding $\hat{\mathbf{g}}^{[m]}$ and a violation of (11). Use the best argument $(\hat{s}_{new}, \hat{t}_{new})$ in the base procedure in (7) from the restricted set V and up-date to get a new $\hat{f}^{[m]}(\cdot)$. Go back to Step 2 (a).

4.2. *Two problems discussed with Peter Bickel around 1995*

What is a linear process? This question has been raised by Peter Bickel when the first author has been staying at Berkeley. We then realized that the class of linear processes is surprisingly large and we pointed out the difficulties for testing whether a process is linear (Bickel and Bühlmann, 1996; 1997). While the latter is more a negative result, the former "supports" the idea of approximations with linear processes in an automatic way. The notion of an automatic approximation, i.e. learned by a machine, has been particularly motivated for bootstrapping linear time series.

We certainly do not rule out the possibility that some time series data should be modelled using nonlinear processes, ideally by incorporating mechanistic understanding about the underlying scientific problem.

How should we model higher-order Markov transition probabilities? When sticking to automatic approximations for a stationary process, it is most common to do this in a Markovian framework of higher order. The transition probabilities or densities for X_t given the previous $X_{t-1}, \ldots$ are then of the form

$$p(x_t | x_{t-1}, \ldots) = p(x_t | x_{t-1}, \ldots, x_{t-p}). \tag{12}$$

Obviously, if p is large we run into the curse of dimensionality.

In the context of categorical processes with values in a finite space, the idea of variable length Markov chains $\{X_t\}_{t\in\mathbb{Z}}$ (Rissanen, 1983; Bühlmann and Wyner, 1999) is

$$p(x_t|x_{t-1},\ldots) = p(x_t|x_{t-1},\ldots,x_{t-\ell}),$$
$$\ell = \ell(x_{t-1},\ldots) = \min\{r; \mathbb{P}[X_t = x_t|X_{-\infty}^{t-1} = x_{-\infty}^{t-1}] = \mathbb{P}[X_t = x_t|X_{t-r}^{t-1} = x_{t-r}^{t-1}] \text{ for all } x_t\},$$

where $\ell \equiv 0$ corresponds to independence.

Here, we denote by $x_{t-r}^{t-1} = x_{t-1},\ldots,x_{t-r}$. Thus, the memory-length function $\ell = \ell(\cdot)$ depends on how the past actually looks like. It can be conveniently represented by a tree: Figure 1 shows an example for a binary process. The memory-length function can be read top-down from the tree:

$$\ell(x_{t-1},\ldots) = \begin{cases} 2 \text{ if } x_{t-1} = 1, x_{t-2} \in \{0,1\}, x_{-\infty}^{t-3} = \text{ arbitrary}, \\ 2 \text{ if } x_{t-1} = 0, x_{t-2} = 1, x_{-\infty}^{t-3} = \text{ arbitrary}, \\ 3 \text{ if } x_{t-1} = 0, x_{t-2} = 0, x_{t-3} \in \{0,1\}, x_{-\infty}^{t-4} = \text{ arbitrary}. \end{cases}$$

Each terminal node in the tree from Figure 1 corresponds to a state in the variable length Markov chain. It becomes clear from the tree repre-

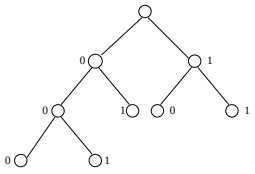

Fig. 1. Tree representation of a variable length Markov chain with binary values. The memory-length function can be read top-down from the tree. Each terminal node corresponds to a state in the Markov chain.

sentation that the model for the memory is hierarchical: Peter Bickel kept asking whether one could allow for "holes" in the tree and proceed non-hierarchically. This issue has also shown up in the context of motif finding in computational biology by the need to model long range interactions: for example, Zhao et al. (2004) propose permuted variable length Markov models.

From the perspective of boosting or also the Lasso, we can do a computationally efficient approach for modelling non-hierarchical dependence. Take p in (12) large and do a sparse model fit: as a result $p(x_t|x_{t-1},\ldots,x_{t-p}) = p(x_t|\{x_{t-j}; j \in \mathcal{A}\})$ where $\mathcal{A} \subseteq \{1,\ldots,p\}$, i.e. only *some* of the lagged variables will be effective for modelling x_t. What seems quite natural today has not been so obvious 10 years ago. We have described in section 4.1 a boosting approach for non-hierarchical, higher order Markov modelling in the context of multivariate, linear time series; from a methodological point of view, the multivariate-ness causes no additional major complication for our sparse model fitting, while for more classical approaches, the extension to high-multivariate models is often complicated.

4.3. L_2Boost-Bootstrap

We propose here a new bootstrap for stationary, multivariate time series, based on vector autoregressive model fitting. It works as follows.

L_2Boost-Bootstrap

Step 1. Specify the maximal lag p.

Step 2. Use multivariate L_2Boosting for the vector-autoregressive model of order p as in (9), with the modification from section 4.1. This yields estimated coefficient matrices $\hat{\mathbf{A}}_1,\ldots,\hat{\mathbf{A}}_p$.
Compute residuals $\hat{\mathbf{E}}_t = \mathbf{X}_t - \sum_{j=1}^p \hat{\mathbf{A}}_j \mathbf{X}_{t-j}$ and consider the centered versions $\tilde{\mathbf{E}}_t = \hat{\mathbf{E}}_t - (n-p)^{-1}\sum_{t=p+1}^n \hat{\mathbf{E}}_t$ $(t = p+1,\ldots,n)$.

Step 3. Do i.i.d. resampling, more than n times, from the empirical c.d.f. of $\tilde{\mathbf{E}}_t$ which yields $\mathbf{E}_1^*,\ldots,\mathbf{E}_{n+k}^*$. Then, generate

$$\mathbf{X}_1^* = \ldots = \mathbf{X}_p^* = 0,$$
$$\mathbf{X}_t^* = \sum_{j=1}^p \hat{\mathbf{A}}_j \mathbf{X}_{t-j}^* + \mathbf{E}_t^*, \quad t = p+1,\ldots,n+k.$$

Use the last stretch $\mathbf{X}_{k+1}^*,\ldots,\mathbf{X}_{n+k}^*$ as a (approximately) stationary realization from the fitted model.

Step 4. Define the bootstrapped estimator by the plug-in rule:

$$\hat{\theta}^* = h_n(\mathbf{X}_{k+1}^*,\ldots,\mathbf{X}_{n+k}^*),$$

where $h_n(\cdot)$ is the function defining the estimator $\hat{\theta} = h_n(\mathbf{X}_1,\ldots,\mathbf{X}_n)$.

The maximal lag p should be chosen "large": due to sparse model fitting, there is often no big decrease in performance when choosing a too large p. Theorems 1 and 2 support this proposal;. The value k should be chosen "large" to ensure that the initial conditions for the simulation in Step 3 have negligible influence.

Bootstrap inference can now be done in a standard way. Note that in contrast to model-based time series bootstraps, the block bootstrap (Künsch, 1989) is not using the plug-in rule as in Step 4; for a discussion, see also Bühlmann (2002).

4.3.1. *Some numerical examples*

It is not difficult to present examples where the L_2Boost-Bootstrap is superior than a vector-autoregressive, model based bootstrap using the (non-parsimonious) Yule-Walker estimator with the AIC-order selection (AR-YW Bootstrap). The gains of the L_2Boost-Bootstrap over the AR-YW method are most pronounced if q is large relative to sample size and the true model is sparse; but already for moderate q, we see substantial improvements (see below). Sparseness includes dependence structures with "holes": e.g. the true coefficients are $\mathbf{A}_1 \neq \mathbf{0},\ \mathbf{A}_2 = \mathbf{0},\ \mathbf{A}_3 \neq \mathbf{0}$, corresponding to a sparse VAR(3).

We present in Figure 2 variance estimates with the L_2Boost-Bootstrap and the AR-YW bootstrap for sample partial autocorrelation estimators

$$\widehat{\text{Parcor}}(\mathbf{X}_t^{(r)}, \mathbf{X}_{t-j}^{(s)}) \tag{13}$$

for some lags $j \in \{1, 2, 3\}$ and some $r, s \in \{1, \ldots, q\},\ r \neq s$. The sample partial correlation is the sample correlation after having subtracted the linear effects (from OLS regression) of all variables $\mathbf{X}_{t-1}, \ldots, \mathbf{X}_{t-j+1}$ in between. The true underlying model is a 5-dimensional VAR(3) ($q = 5$) as in (9) with $\mathbf{A}_2 = \mathbf{0}$ and $\mathbf{A}_1,\ \mathbf{A}_3$ sparse having only 7 non-zero entries (out of 25) each. Sample size is chosen as $n = 50$. The number of bootstrap samples is 1000. The reductions in mean squared error of the L_2Boost-Bootstrap variance estimates over the AR-YW bootstrap for the five situations from left to right in Figure 2 are substantial:

MSE reductions: 41.9% 62.7% 64.7% 74.4% 61.6%

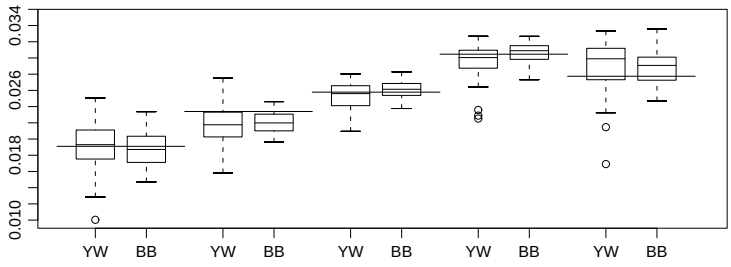

Fig. 2. Boxplots of bootstrap variance estimates of sample partial autocorrelations as in (13) for 5 combinations of various lags j and components r, s: AR-YW bootstrap (YW) and L_2Boost-Bootstrap with AIC_c stopping (BB); true variances indicated by horizontal line. 50 model simulations.

4.4. *Graphical modelling for stochastic processes: an outlook*

Graphical models are very useful for describing conditional dependencies for multivariate observations. For multivariate stationary stochastic processes $\{\mathbf{X}_t\}_{t\in\mathbb{Z}}$, an object of interest is the conditional dependence structure among the components $\{\mathbf{X}_t^{(j)}\}_{t\in\mathbb{Z}}$ for $j = 1, \ldots, q$. Various notions of conditional dependencies, associations and causality exist, cf. Brillinger (1996) or Dahlhaus and Eichler (2003).

Without having some mechanistic understanding about the underlying process $\{\mathbf{X}_t\}_{t\in\mathbb{Z}}$, and if q is large relative to sample size n, the Gaussian framework often yields a useful first approximation. It requires knowledge of second order moments of the process $\{\mathbf{X}_t\}_{t\in\mathbb{Z}}$ only, and this may be approximated by (potentially high-order) VAR(p) models as in (9). One specific graphical model within this framework is the partial correlation graph: Dahlhaus and Eichler (2003) give a precise description for VAR(p) models in terms of zero-elements in the coefficient matrices $\mathbf{A}_j$. When using our multivariate L_2Boosting from sections 3.3 and 4.1, we get a sparse VAR(p) model fit, and we can then immediately read off a boosting estimate for a partial correlation graph. We remark that such a straightforward graph estimate is only possible due to the sparseness of the VAR(p)-model fit with many estimated zeroes.

So far, we have not analyzed some properties of the boosting estimate for (high-dimensional) partial correlation graphs for stochastic processes. But we think that the boosting methods have an interesting potential for graphical modelling, both for multivariate stationary stochastic processes

as outlined above as well as for multivariate i.i.d. data.

References

1. Bickel, P.J. and Bühlmann, P. (1996). What is a linear process? Proc. Nat. Acad. Sci. USA **93**, 12128–12131.
2. Bickel, P.J. and Bühlmann, P. (1997). Closure of linear processes. J. Theor. Probab. **10**, 445–479.
3. Bickel, P.J., Klaassen, C.A.J., Ritov, Y. and Wellner, J.A. (1993). *Efficient and Adaptive Estimation for Semiparametric Models.* Johns Hopkins University Press, Baltimore.
4. Bickel, P.J. and Ritov, Y. (2004). The golden chain. Discussion on three papers on boosting (auths: Jiang, W.; Lugosi, G. and Vayatis, N.; Zhang, T). Ann Statist. **32**, 91–96.
5. Breiman, L. (1998). Arcing classifiers. Ann. Statist. **26**, 801–849 (with discussion).
6. Breiman, L. (1999). Prediction games & arcing algorithms. Neural Computation **11**, 1493-1517.
7. Brillinger, D.R. (1996). Remarks concerning graphical models for time series and point processes. Brazil. Rev. Econometrics **16**, 1–23.
8. Brockwell,P.J. and Davis,R.A. (1991). *Time Series: Theory and Methods.* 2nd ed. Springer, New York.
9. Bühlmann, P. (2002). Bootstraps for time series. Statistical Science **17**, 52–72.
10. Bühlmann, P. (2004). Boosting for high-dimensional linear models. To appear in Ann. Statist.
11. Bühlmann, P. and Wyner, A.J. (1999). Variable length Markov chains. Ann. Statist. **27**, 480–513.
12. Bühlmann, P. and Yu, B. (2003). Boosting with the L_2loss: regression and classification. J. Amer. Statist. Assoc. **98**, 324–339.
13. Bühlmann, P. and Yu, B. (2005). Boosting, model selection, lasso and nonnegative garrote. Preprint.
14. Buja, A., Hastie, T. and Tibshirani, R. (1989). Linear smoothers and additive models. Ann. Statist. **17**, 453–555 (with discussion).
15. Dahlhaus, R. and Eichler, M. (2003). Causality and graphical models for time series. In *Highly structured stochastic systems* (P. Green, N. Hjort, and S. Richardson eds.). Oxford University Press, Oxford.
16. DeVore, R.A. and Temlyakov, V.N. (1996). Some remarks on greedy algorithms. Adv. Comp. Math. **5**, 173–187.
17. Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. (2004). Least angle regression. Ann. Statist. **32**, 407–499 (with discussion).
18. Freund, Y. and Schapire, R.E. (1996). Experiments with a new boosting algorithm. In Machine Learning: Proc. Thirteenth International Conference, pp. 148–156. Morgan Kauffman, San Francisco.
19. Friedman, J.H. (2001). Greedy function approximation: a gradient boosting machine. Ann. Statist. **29**, 1189–1232.

20. Friedman, J.H., Hastie, T. and Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting. Ann. Statist. **28**, 337–407 (with discussion).
21. Greenshtein, E. and Ritov, Y. (2004). Persistence in high-dimensional linear predictor selection and the virtue of overparametrization. Bernoulli **10**, 971–988.
22. Hurvich, C.M., Simonoff, J.S. and Tsai, C.-L. (1998). Smoothing parameter selection in nonparametric regression using an improved Akaike information criterion. J. Roy. Statist. Soc., Ser. B, **60**, 271–293.
23. Jiang, W. (2004). Process consistency for AdaBoost. Ann. Statist. **32**, 13–29 (disc. pp. 85–134).
24. Künsch, H.R. (1989). The jackknife and the bootstrap for general stationary observations. Ann. Statist. **17**, 1217–1241.
25. Lugosi, G. and Vayatis, N. (2004). On the Bayes-risk consistency of regularized boosting methods. Ann. Statist. **32**, 30–55 (disc. pp. 85–134).
26. Lutz, R.W. and Bühlmann, P. (2005). Boosting for high-multivariate responses in high-dimensional linear regression. To appear in Statistica Sinica.
27. Mallat, S and Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries. IEEE Trans. Signal Proc. **41**, 3397–3415.
28. Rissanen, J. (1983). A universal data compression system. IEEE Trans. Inform. Theory **IT-29**, 656–664.
29. Temlyakov, V.N. (2000). Weak greedy algorithms. Adv. Comp. Math. **12**, 213–227.
30. Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. J. Roy. Statist. Soc., Ser. B, **58**, 267–288.
31. Tukey, J.W. (1977). *Exploratory data analysis.* Addison-Wesley, Reading, MA.
32. Zellner, A. (1962). An efficient method of estimating seemingly unrelated regressions and tests for aggregation bias. J. Amer. Statist. Assoc. **57**, 348–368.
33. Zellner, A. (1963). Estimators for seemingly unrelated regression equations: some exact finite sample results. J. Amer. Statist. Assoc. **58**, 977–992 (Corr. **67**, 255).
34. Zhang, T. and Yu, B. (2003). Boosting with early stopping: convergence and consistency. To appear in Ann. Statist.
35. Zhao, X., Huang, H. and Speed, T.P. (2004). Finding short DNA motifs using permuted Markov models, RECOMB'04.