Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 17(1): 636–653.
DOI: 10.3934/mbe.2020033
Received: 29 May 2019
Accepted: 09 October 2019
Published: 22 October 2019

Research article

Nonlinear autoregressive sieve bootstrap based on extreme learning machines

Michele La Rocca and Cira Perna*

Department of Economics and Statistics, University of Salerno, via Giovanni Paolo II, Fisciano 84084, Italy

* **Correspondence:** Email: perna@unisa.it; Tel: +3989962209; Fax: +3989962048.

Abstract: The aim of the paper is to propose and discuss a sieve bootstrap scheme based on Extreme Learning Machines for non linear time series. The procedure is fully nonparametric in its spirit and retains the conceptual simplicity of the residual bootstrap. Using Extreme Learning Machines in the resampling scheme can dramatically reduce the computational burden of the bootstrap procedure, with performances comparable to the NN-Sieve bootstrap and computing time similar to the AR-Sieve bootstrap. A Monte Carlo simulation experiment has been implemented, in order to evaluate the performance of the proposed procedure and to compare it with the NN-Sieve bootstrap. The distributions of the bootstrap variance estimators appear to be consistent, delivering good results both in terms of accuracy and bias, for either linear and nonlinear statistics (such as the mean and the median) and smooth functions of means (such as the variance and the covariance).

Keywords: sieve bootstrap; nonlinear time series; extreme learning machines; neural networks; Monte Carlo

1. Introduction

In the last decades, in the context of time series, several bootstrap procedures have been proposed to infer properties of a statistic of interest, including both parametric and non-parametric resampling schemes. Anyway, the effectiveness of the different bootstrap procedures is related to their ability to capture the dependent probabilistic structure of the underlying stochastic process under study, and to the analytical properties of the particular statistic considered.

When it is reasonable to impose parametric type assumptions on the process, the residual bootstrap is usually a sensible choice. In this general bootstrap scheme, the dependence structure of the series is modelled explicitly by using a parametric model and the bootstrap sample is drawn from the fitted model. This approach can be easily implemented, leading to very efficient results, but it is very sensitive

to model misspecification, in which case it leads to bootstrap estimators, which might be not consistent. In this latter case, or when it is possible to account only for some mixing or weak dependence structure, more complex and fully non-parametric bootstrap methods are required. In this context, sieve bootstrap schemes have become useful and powerful tools to capture the dependence structure of the data without imposing any rigid parametric model specification.

The basic idea of this approach is to use non-parametric estimators as sieve approximators. In particular, the stochastic process under study is approximated by a family of (semi-) parametric models which, in a proper sense, contains the original process. Fixed an appropriate model selection rule, a model is picked from the set of the identified family and estimated on the observed dataset. Residual bootstrap is then implemented on the previously estimated model. In the context of sieve bootstrap schemes, the most used approach is the AR-Sieve bootstrap procedure [1–3]. It is based on the method of autoregressive process sieve in which an $AR(p_T)$ model is fitted to the observed data and a bootstrap sample is generated by resampling from the centred residuals. This resampling scheme retains the simplicity of the classical residual bootstrap while being a nonparametric bootstrap scheme. It enjoys the properties of a plug-in rule. Moreover, it does not exhibit artefacts in the dependence structure like in the blockwise bootstrap, and there is no need for 'pre-vectorizing' the original observations. For these reasons, the AR-Sieve bootstrap has been largely used in the literature for constructing prediction intervals for linear processes [4–9]; for unit root testing [10] and stationarity testing [11], for fractionally integrated and non-invertible processes [12, 13]. More recently, it has been used also in the context of functional time series [14] and spatial processes [15].

However, the AR-Sieve bootstrap performs better than other bootstrap techniques if the data generating process is linear and representable as an AR(∞) process. Moreover, for quite general processes, it is expected to deliver consistent results if the asymptotic distribution of a given statistic is determined solely by the first and second order moment structure [16] or if the original time series is transformed and a more complex residual bootstrap scheme is applied [17].

For general nonlinear processes, an approach based on the use of feedforward Neural Networks (NN) as sieve approximators has been proposed [18]. The NN-Sieve resampling scheme, which is non-parametric in its spirit, retains the conceptual simplicity of the AR-Sieve bootstrap and it delivers consistent results for quite general nonlinear processes. Moreover, it performs similarly to the AR-Sieve bootstrap for linear processes while it outperforms the AR-Sieve bootstrap and the moving block bootstrap for nonlinear processes, both in terms of bias and variability [19].

However, despite their proven theoretical capabilities of non-parametric data-driven universal approximation of general nonlinear functions, NNs face challenging issues concerning the computational burden, which can be very heavy especially for complex non-linear generating processes. This might be a serious concern, even with computational power available nowadays, when using computer intensive model selection techniques (such as cross-validation) to tune the neural network hyper-parameters (such as hidden layer size, weight decay, etc.) in a resampling scheme involving neural networks. Moreover, in many applications of the bootstrap, to incorporate uncertainty due to model estimation, the statistical model needs to be estimated on each bootstrap resampled data. This makes the use of the bootstrap unfeasible when applied to complex estimation procedures. Finally, many statistical problems could be solved in practice only by using iterative bootstrap, which of course requires very efficient estimation procedures.

To overcome these problems, our proposal is to estimate the neural network model in the NN-Sieve

procedure by using learning without iterative tuning. This approach, known as Extreme Learning Machines (ELM) in the computational intelligence and machine learning literature [20, 21], has been extensively studied and remarkable contributions have been made both in theories and applications. By using ELMs, a nonlinear autoregressive sieve bootstrap scheme can be implemented for general nonlinear time series. This scheme has the advantage to dramatically reduce the computational burden of the overall procedure; moreover, it has performances comparable to the NN-Sieve bootstrap and computing time comparable to the AR-Sieve bootstrap.

The paper is organized as follows. In section 2 a brief review of neural networks for time series is introduced in the context of Nonlinear Autoregressive (NAR) time series. In section 3 the extreme learning machine approach is presented and discussed, highlighting the advantages of its use with respect to the classical neural network approach. In section 3, NAR Sieve bootstrap based on ELMs is proposed, emphasizing the improvement with respect to alternative bootstrap schemes. In section 4 a simulation experiment is performed in order to evaluate the performance of the proposed bootstrap procedure and to compare it with the NN approach. Some remarks close the paper.

2. Neural networks for time series analysis

Let $\{Y_t, t \in \mathbb{Z}\}$ be a real valued stochastic process modeled as a nonlinear autoregressive process with exogenous components:

$$Y_t = m\left(Y_{t-1}, \ldots, Y_{t-p}, \mathbf{X}_t\right) + \varepsilon_t \tag{2.1}$$

where $m(\cdot)$ is an unknown (possibly nonlinear) function, ε_t are i.i.d. innovations with mean 0 and finite variance and $\mathbf{X}_t$ is a d-dimensional stochastic process representing other explicative variables , useful in predicting Y_t or other functionals related to Y_t.

By denoting with $\mathcal{I}_t$ the information set available at time t and using the abbreviation $\mathbf{Z}_t = (Y_{t-1}, \ldots, Y_{t-p}, \mathbf{X}_t)$ the conditional expectation of Y_t, given the information set $\mathcal{I}_t$ is given by:

$$\mathbb{E}(Y_t|\mathcal{I}_t) = m(\mathbf{Z}_t) \tag{2.2}$$

There are many different parametric approaches to modelling the function m and they can give quite different answers in the range of perhaps most interest to practitioners. In many cases, they are not willing to assume any parametric form (avoiding, in this way, model mispecification errors) and this motivates a nonparametric approach, because of the greater fexibility in functional form thereby allowed. For models with simple lag structure and without the presence of exogenous variables, the problem has been thoroughly investigated using local smoothers, such as in [22].

However, the use of local smoothers, due the so called "curse of dimensionality", does not allow complex lag structure or inclusion of other explanatory variables. These issues lead Franke and Diagne [23] to propose an alternative approach based on feedforward neural networks with one hidden layer. The function m can be approximated by using neural networks with a single output and additive nodes in the class:

$$\mathcal{F} = \left\{ f(\mathbf{z}, \boldsymbol{\eta}) : \mathbf{z} \in \mathbb{R}^{p+d}, \boldsymbol{\eta} \in \mathbb{R}^{r(p+d+2)} \right\} \tag{2.3}$$

with:

$$f_r(\mathbf{z}, \boldsymbol{\eta}) = \sum_{k=1}^{r} \beta_k \psi(\mathbf{a}_k' \mathbf{z} + b_k) \tag{2.4}$$

where r is the hidden layer size, $\psi(\cdot)$ is a sigmoidal activation function; $\boldsymbol{\eta} = (\beta_1, \ldots, \beta_r, \mathbf{a}_1', \mathbf{a}_2', \ldots, \mathbf{a}_r', b_1, \ldots, b_r)'$; $\{\mathbf{a}_k\}$ are the $(p + d)$ dimensional vectors of weights for the connections between input layer and hidden layer; $\{\beta_k\}$ are the weights of the link between the hidden layer and the output; $\{b_k\}$ are the bias terms of the hidden neurons.

Suppose that the processs Y_t and $\mathbf{X}_t$ are observed for T consecutive time periods generating the time series $y_1, y_2, \ldots, y_T$ and $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$ along with appropriate initial conditions. Let $\mathbf{u}_t = (y_t, \mathbf{x}_t)$ and $\mathbf{z}_t = (y_{t-1}, y_{t-2}, \ldots, y_{t-p}, \mathbf{x}_t)$. A consistent estimate of the regression function $m(\cdot)$ can be obtained as:

$$\hat{m} = \text{argmin}_{f \in \mathcal{F}} \left\| f(\mathbf{z}_t, \boldsymbol{\eta}) - \mathbf{y} \right\| \tag{2.5}$$

where $\| \cdot \|$ denotes the L_2-norm. Alternatively, to improve the stability of the network solution, a regularized version of the optimization problem 2.5 can be used:

$$\hat{m} = \text{argmin}_{f \in \mathcal{F}} \left\| f(\mathbf{z}_t, \boldsymbol{\eta}) - \mathbf{y} \right\| + \lambda \left\| \boldsymbol{\eta} \right\| \tag{2.6}$$

where the tuning parameter λ is usually fixed by cross-validation.

Neural networks provide an arbitrarily accurate approximation to the unknown target function which satisfy certain smoothness conditions. In particular, under quite general conditions on the activation function ψ, there exists a sequence of network functions $\{f_r\}$ approximating to any given continuous target function m with any expected learning error. A deterministic approximation rate (in L_2-norm) of $r_T^{-1/2}$ for NNs with sigmoid activation functions has been obtained in [24]. For better rates see [25–27]. If the network model is fitted to the data in such a way that complexity of the network is allowed to increase at a proper rate with the sample size, the resulting function estimator can then be viewed as a nonparametric sieve estimator [28, 29]. Moreover, estimation of hidden layer size (r_T) seems to be less critical than estimation of the window size in local nonparametric approaches. Finally, extension of the sieve bootstrap procedure to high dimensional models is (much) more straightforward than other nonparametric approaches (absence of 'curse of dimensionality').

However, despite their proven theoretical capabilities of non-parametric data driven universal approximation of a general class of nonlinear functions, NNs face other challenging issues. Firstly, the use of the NNs requires the specification of the network topology in accordance with the underlying structure of the series. It involves the specification of the size and the structure of the input layer, the size of the hidden layer, the signal processing within nodes (i.e., the choice of the activation function). Furthermore, in the context of time series analysis, the characteristics of the series and the presence of deterministic and/or stochastic components such as trend, seasonality, structural breaks and level shift impose also an accurate feature selection. Many strategies have been proposed to solve these problems (for example, [30, 31]) but the difficulty to find a unique method able to automatically identify the optimal NN still remains an open issue.

Moreover, once the neural network architecture is fixed, the estimation of the parameters can be made by using the backpropagation algorithm, which is essentially a first order gradient method for parameter optimization and suffers from slow convergence and local minimum problem. Also in this context, various ways to improve the efficiency or optimality in training a neural network have been

proposed. They include second order optimization methods [32], subset selection methods [33] or global optimization methods [34]. Although these methods lead to faster training speed and, in general, better generalization performance compared to the back propagation algorithm, most of them still cannot guarantee a global optimal solution.

3. Extreme learning machines in a nutshell

Recently, extreme learning machine (ELM) for training NNs has attracted the attention in the literature [35–37] as a possible method to overcome some challenges faced by the other techniques.

The essence of ELMs is that, unlike the other traditional learning algorithms, such as back propagation based neural networks, the hidden nodes weights are randomly generated and they need not to be tuned, so that the algorithm analytically determines the output weights of NNs.

Basically, ELM trains a NN in two main stages. In the first, ELM randomly initializes the hidden layer to map the input data into a feature space by some non linear functions. As in NN theory, they can be any nonlinear piecewise continuous functions such as the sigmoid or the hyperbolic functions. The hidden node parameters $(\mathbf{a}, b)$ are randomly generated according to any continuos probability distribution so that the matrix:

$$\mathbf{H} = \begin{bmatrix} \psi(\mathbf{a}_1'\mathbf{z}_1 + b_1) & \cdots & \psi(\mathbf{a}_r'\mathbf{z}_1 + b_r) \\ \vdots & \ddots & \vdots \\ \psi(\mathbf{a}_1'\mathbf{z}_T + b_1) & \cdots & \psi(\mathbf{a}_r'\mathbf{z}_T + b_r) \end{bmatrix} \tag{3.1}$$

is completely known. In the second stage, the output weights $\boldsymbol{\beta}$ are estimated by solving the following minimization problem:

$$\hat{\boldsymbol{\beta}} = \text{argmin}_{\boldsymbol{\beta}} \|\mathbf{H}\boldsymbol{\beta} - \mathbf{y}\| \tag{3.2}$$

where $\mathbf{y}$ is the training data target vector and $\|\cdot\|$ denotes the L_2-norm.

If $\mathbf{H}^\dagger$ denotes the Moore-Penrose generalized inverse of matrix $\mathbf{H}$, the optimal solution to the previous optimization problem is:

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^\dagger \mathbf{y} \tag{3.3}$$

The matrix $\mathbf{H}^\dagger$ can be calculated by using one of the numerous methods proposed in the literature which include orthogonal projection, orthogonalization method, iterative method and the single value decomposition, the last one being the most general.

The estimation of the parameter vector $\boldsymbol{\beta}$ can also be obtained via regularized ELM [38]. If $\mathbf{H}$ has more rows than columns ($T > r$), which is usually the case when the number of training data is larger than the number of hidden neurons, the following closed form solution can be obtained:

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{H}'\mathbf{H} + \frac{\mathbf{I}}{C}\right)^{-1} \mathbf{H}'\mathbf{y} \tag{3.4}$$

where $\mathbf{I}$ is an identity matrix of dimension r and C is a proper chosen constant.

If the number of training data is less than the number of hidden neurons ($T < r$), an estimate for $\boldsymbol{\beta}$ can be obtained as:

$$\hat{\boldsymbol{\beta}} = \mathbf{H}' \left(\mathbf{H}\mathbf{H}' + \frac{\mathbf{I}}{C} \right)^{-1} \mathbf{y}$$

where $\mathbf{I}$ is an identity matrix of dimension T this time.

It can be shown that Eq. 3.4 actually aims at minimizing:

$$\hat{\boldsymbol{\beta}} = \text{argmin}_{\beta} \|\mathbf{H}\boldsymbol{\beta} - \mathbf{y}\| + \frac{1}{C} \|\boldsymbol{\beta}\|$$

Comparing to standard ELM, in which the target is to minimize $\|\mathbf{H}\boldsymbol{\beta} - \mathbf{y}\|$, an extra penalty term $\frac{1}{C}\|\boldsymbol{\beta}\|$ is added to the target of standard ELM. This is actually consistent to the theory that smaller output weights $\boldsymbol{\beta}$ play an important role for ELM in achieving better generalization ability.

The ELM approach have several advantages. Firstly, it has good generalization performance in the sense that it reaches the small training error and, contemporaneously, the smallest norm of output weights. Secondly, learning can be done without iteratively tuning the hidden nodes which can be independent of training data. Moreover, ELMs, like NNs, enjoy the property of being universal approximators. Given any non constant piecewise continuous function ψ, if

$$\text{spam} \left\{ \psi(\mathbf{a}, b, \mathbf{z}) : (\mathbf{a}, b) \in \mathbb{R}^{p+d} \times \mathbb{R} \right\}$$

is dense in L^2, for any continuous target function m and any sequence $\psi(\mathbf{a}_k'\mathbf{z} + b_k)$ for $k = 1, \ldots, r$ randomly generated according to any continuous sampling distribution and if the output weights $\hat{\boldsymbol{\beta}}$ are determined by ordinary least square to minimize:

$$\left\| m(\mathbf{z}) - \sum_{k=1}^{r} \hat{\boldsymbol{\beta}} \psi(\mathbf{a}_k'\mathbf{z} + b_k) \right\|$$

it can be shown [39–41] that, with probability one, it is:

$$\lim_{r \to \infty} \|m - f_r\| = 0 \tag{3.5}$$

This result states the universal approximation capability of ELMs without imposing any restrictive assumption on the activation function as in the case of NN paradigm in which, on the contrary, a continuous and differentiable activation function is needed. In practice, being the hidden layer randomly generated, ELMs usually require more hidden neurons than NNs to obtain a given performance. However, this does not seem to be a serious problem due to the computational efficiency of ELMs. Moreover, ELMs are well suited for large data processing and, even if a model selection process is implemented for an optimal structure searching, the running time of ELMs is always lower than other competing strategies. In any case, parallel and cloud computing techniques [42] can also be used for even faster implementation of ELMs.

4. NAR Sieve bootstrap based on ELM

Given a general real valued stochastic process $\{Y_t, t \in \mathbb{Z}\} \sim M_0$ modeled as in Eq. 2.1, the basic idea of a sieve bootstrap scheme is to approximate the process by a family of (semi-) parametric models $\mathcal{M} = \{M_j, j \in \mathbb{N}\}$ such that $\cup_{j=1}^{\infty} M_j$ contains (in some sense) the original process M_0. Fixed a proper

Algorithm 1 The NAR-Sieve bootstrap scheme.

Fix B, the number of bootstrap runs

Consider the time series $\{\mathbf{u}_t, t = 1, 2, \ldots, T\}$ with $\mathbf{u}_t = (y_t, \mathbf{z}_t)$

Let $\mathbf{z}_t = \left\{(y_{t-1}, y_{t-2}, \ldots, y_{t-p}, \mathbf{x}_t), t = p+1, \ldots, T\right\}$

Fix n_1, the observations to be discarded in order to make negligible the effect of starting values.

Let $N = n_1 + T$

Estimate $m(\cdot)$ by using an ELM, obtaining $\hat{m}(\cdot)$

Compute the residuals $\hat{\varepsilon}_t = y_t - \hat{m}(\mathbf{z}_t)$

Compute the centered residuals

$$\tilde{\varepsilon}_t = \hat{\varepsilon}_t - (T-p)^{-1} \sum_{t=p+1}^{T} \hat{\varepsilon}_t.$$

Denote the empirical distribution function of the centered residuals $\tilde{\varepsilon}_t$ by

$$F_{\tilde{\varepsilon}}(x) = (T-p)^{-1} \sum_{t=p+1}^{T} \mathbb{I}(\tilde{\varepsilon}_t \leq x)$$

where $\mathbb{I}(\cdot)$ is the indicator function.

for $b = 1, 2, \ldots, B$ **do**

Resample for $t = p+1, p+2, \ldots, N$

$$\varepsilon^*_{(b,t)} \overset{iid}{\sim} F_{\tilde{\varepsilon}}$$

Fix $y^*_{(b,t)} = \bar{y}$ for $t = 1, \ldots, p$. Define $\mathbf{z}^*_{(b,t)} = (y^*_{(b,t-1)}, \ldots, y^*_{(b,t-p)}, \mathbf{x}_t)$.

Define $y^*_{(b,t)}$ by recursion

$$y^*_{(b,t)} = \hat{m}\left(\mathbf{z}^*_{(b,t)}\right) + \varepsilon^*_{(b,t)};\ t = p+1, \ldots, N$$

Compute $\hat{\theta}^*_{(b,T)} = q\left(\mathbf{u}^*_{(b,1)}, \mathbf{u}_{(b,2)}, \ldots, \mathbf{u}^*_{(b,T)}\right)$

end for

Use the empirical distribution function

$$\hat{F}^*(x) = B^{-1} \sum_{b=1}^{B} \mathbb{I}\left(\hat{\theta}^*_{(b,T)} \leq x\right)$$

to approximate the unknown sampling distribution of the estimator $\hat{\theta}_n$.

model selection rule, a model is picked from the set $\mathcal{M}$ and estimated. Residual bootstrap is based on the previous estimated model. Hence, a key issue is the selection of a proper model family.

To elaborate, let $\{\mathbf{u}_t = (y_t, \mathbf{x}_t), t = 1, 2, \ldots, T\}$ be the observed time series, and let θ a finite dimensional parameter of interest and $\hat{\theta}_T = q(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_T)$ a scalar-, vector- or curve-valued estimator, which is a measurable function of the data. Inference on θ can be gained by using the NN-Sieve bootstrap approach. The procedure, proposed in [18], exploits the good properties of neural network modelling and it is shown to be asymptotically justified, delivering consistent results for quite general non linear models, and it yields satisfactory results for finite sample size [19].

Here, we propose to approximate the unknown function $m(\cdot)$ with the class of ELMs with fixed input neurons and hidden layer size going to infinity with the time series length at a proper rate. The resampling procedure is detailed in algorithm 1.

The proposed resampling scheme has some advantages which make it effective in many applicative fields. As the NN-Sieve bootstrap, this approach is asymptotically justified and it delivers consistent results for quite general nonlinear processes [19]. Moreover, it does not suffer for the so-called 'curse of dimensionality'. Theoretically, ELMs are expected to perform better than other approximation methods since the approximation form is not so sensitive to the increasing dimension, at least within the confines of particular classes of functions.

As neural networks, ELMs are global nonparametric methods and their use could stress different features and data structures when compared to local nonparametric methods. With respect the NN-Sieve bootstrap, this scheme dramatically reduces the computational burden of the overall procedure having a computing time comparable to the AR-Sieve bootstrap.

5. Simulation results

In this section, we discuss the results of a simulation experiment performed in order to evaluate the performance of the proposed procedure and to compare it with the NN approach. All computations were implemented in the language R (version 3.6.0) using the package nnet (for feedforward neural networks with regularization) and ad hoc implementation by the authors for ELMs and ELMs with regularization. This latter implementation is based on the package ridge [43] which also includes the selection of the regularization parameter. All computations were made by using two different workstations: the first running MacOS (version 10.14.4) with 4 GHz Intel Core i7 and 16 GB 1600 MHz DDR3 memory, using the standard R math library and, the second running Ubuntu Linux (version 18.04) with a 3.50 Ghz Intel Xeon E5–1650 and 32 Gb ECC DDR4 memory, using OpenBlas.

In the first place, the computational advantage of ELMs is evaluated in terms of computing time of the learning process. In Tables 1 and 2 some descriptive statistics for the computing time for NNs, ELMs and ELMs with regularization are reported for a regression problem with two levels of complexity. The first model has 2 predictors, 2 neurons in the hidden layer and it has been estimated on a sample of 300 observations. In this case, the median execution time for the learning process of a full neural network (with ten random restarts) is 54 times slower with respect to the learning process of ELM with regularization and about 300 times slower with respect to a learning process based on the Moore-Penrose generalized inverse. The second model is much more complex, with 16 predictors, 40 neurons in the hidden layer and it has been estimated on a sample of 2000 observations. In this case, the median execution time for the learning process of a full neural network (with ten random

restarts) is about 478 times slower with respect to the learning process of ELM with regularization and about 1575 times slower with respect to a learning process based on the Moore-Penrose generalized inverse. The more complex the model, the higher the advantage of using ELMs. As a remark, note that it is well known that the standard R math library does not deliver the best performance for linear algebra operations. So the figures reported in Tables 1 and 2 could be significantly improved using better Basic Linear Algebra System (BLAS) implementations such as the OpenBlas or Intel's Math Kernel Library (MKL), as evident from the results reported in the following.

Table 1. Descriptive statistics of computing time (in milliseconds) for feedforward neural networks (NN), extreme learning machines (ELM) and extreme learning machines with regularization (Reg), for a sample size of 300 observations, input size = 2, hidden layer size = 2. Computations made on a MacOS with 4 GHz Intel Core i7 and 16 GB 1600 MHz DDR3, using the standard R math library.

Method	min	1^{st} quart.	mean	median	3^{rd} quart.	max	runs
NN	37.54	48.22	54.14	54.75	59.02	73.29	100
Reg	0.75	0.95	1.40	1.02	1.13	24.56	100
ELM	0.11	0.16	0.40	0.18	0.20	21.99	100

Table 2. Descriptive statistics of computing time (in milliseconds) for feedforward neural networks (NN), extreme learning machines (ELM) and extreme learning machines with regularization (Reg), for a sample size of 2,000 observations, input size = 16, hidden layer size = 40. Computations made on a MacOS with 4 GHz Intel Core i7 and 16 GB 1600 MHz DDR3, using the standard R math library.

Method	min	1^{st} quart.	mean	median	3^{rd} quart.	max	runs
NN	15886.80	17575.38	17906.21	17791.85	18252.19	20371.12	100
Reg	32.86	35.11	41.46	37.18	43.88	256.92	100
ELM	8.62	10.85	11.97	11.29	11.77	22.29	100

In Figures 1 and 2, the relative execution time between NN, ELM and ELM with regularization is reported, for different number of input neurons $d \in \{4, 8, 12, 16\}$, different hidden layer size $r = 2, 3, \ldots, 40$ and samples of different size $\{300, 500, 1000, 2000\}$. In all cases the computational gain when using ELM is very significant, ranging from about 300 times (for the simplest model with two input neurons and two hidden neurons) to 6,000 times for the most complex model considered (16 input neurons and 40 hidden neurons) by using OpenBlas. Note that to make the comparison between the OpenBlas implementation and the standard R math library implementation (which is single threaded) fair, we have forced the OpenBlas to use a single thread. Therefore, even better performance might be expected using a multi-threaded OpenBlas implementation. All neural networks have been estimated by restarting the learning process ten times, to avoid being trapped in local minima (an operation not necessary when using ELMs). In many applications, it is a standard choice to use 50 random restarts, making the advantage in using ELM even more effective. As a further remark, note that when using the ELM with the regularization learning process, the computational advantage is reduced by a factor of 4 on average. However, this is still an important figure. In the NN learning process, the regularization

parameter has not been estimated on each dataset but fixed on the base of ad hoc choices. The selection of this tuning parameter for NNs is usually based on cross-validation, making the computational burden heavy and in many applications unfeasible.

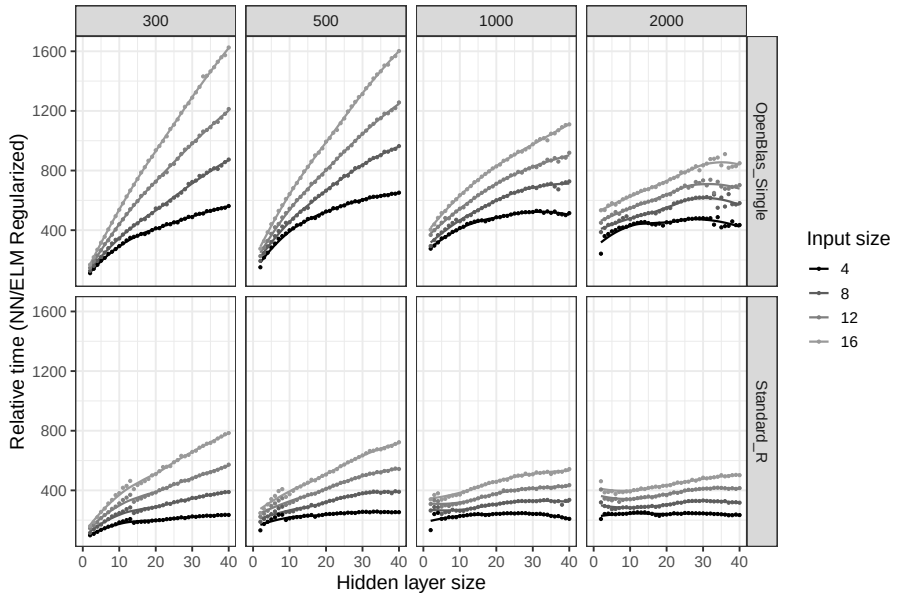

Figure 1. NN vs ELM with regularization relative estimation computing time: $d = 4, 6, 8, 10$; $T = 300, 500, 1000, 2000$; number of random restarts of NN = 10; maximum number of iteration for NN = 200; ratios of the average estimation time over 100 Monte Carlo runs.

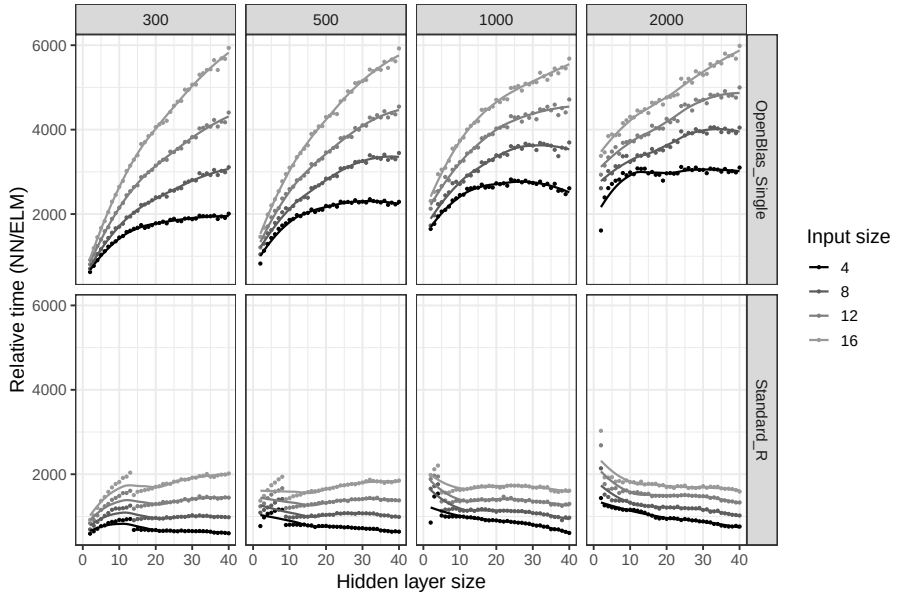

Figure 2. NN vs ELM relative estimation computing time: $d = 4, 6, 8, 10$; $T = 300, 500, 1000, 2000$; number of random restarts of NN = 10; maximum number of iteration for NN = 200; ratios of the average estimation time over 100 Monte Carlo runs.

The computational advantage in using ELM (with or without regularization) in bootstrap schemes is so high that the whole bootstrap distribution for a given statistic of interest could be estimated using basically the same computational time that it is needed for a single neural network estimation (by considering 1000/2000 bootstrap runs to approximate the bootstrap distribution). Moreover, consider that in several applications, the neural network model needs to be re-estimated on each bootstrap run. This is the case, for example, when computing bootstrap prediction intervals, where model re-estimation is needed to incorporate uncertainty due to model estimation. In this latter case, using NNs is almost unfeasible, while using ELMs makes the overall bootstrap procedure possible, with very reasonable computing time.

As a further step in the simulation design, we wish to evaluate and compare the performance of the bootstrap procedure using NNs and ELMs, in order to check if there is any loss in accuracy for the bootstrap inference process. The experimental setup is based on datasets generated by different nonlinear models, with different degrees on nonlinearity.

We consider the class of STAR models as specified in [45]:

$$Y_t = \phi_1 Y_{t-1} - (\phi_1 - \phi_2) F\left(Y_{t-1}, \gamma\right) Y_{t-1} + \varepsilon_t$$

with $\varepsilon_t \sim N(0,1)$. The function F determines different transition processes. If $F(u,\gamma) = 1 - \exp(-\gamma u^2)$ we get exponential STAR model (ESTAR) while using $F(u,\gamma) = \frac{1}{1+\exp(-\gamma u)}$ we get a logistic smooth transition model (LSTAR). These models, very popular in several applications in different fields, have been chosen as representing different kind of dynamic behaviour, since their flexibility allows generation of quite different time series structures.

The degree of non-linearity in the LSTAR/ESTAR models is controlled by the parameter γ in the transition function. When $\gamma \to 0$, the transition function tends towards 0, and the model will be a simple autoregressive process. When $\gamma \to \infty$, the transition function converges towards unity, which implies that the model is a different autoregressive model with coefficients equal to the mean of the autoregressive parameters of the two regimes.

The parameters ϕ_1, ϕ_2 and γ have been fixed according to the values in Table 3, leading to four different specifications for each class of models (denoted as v1, v2, v3 and v4).

Table 3. Parameter values for the LSTAR/ESTAR models chosen according to the combinations v1, v2, v3 and v4.

	v1	v2	v3	v4
ϕ_1	0.1	0.6	0.1	0.6
ϕ_2	-0.1	-0.6	-0.1	-0.6
γ	5	5	25	25

In order to evaluate the distribution of either linear estimators or simple nonlinear estimators, we consider, for all the models, four statistics: the sample mean, the sample median, the sample variance and the sample autocovariance (at lag 1). Let $\hat{\theta}$ be the statistic of interest and let $\mathcal{R}^* = (\hat{\theta}^* - \mathbb{E}_*[\hat{\theta}^*])/\sigma_T$ the bootstrap counterpart of the root $\mathcal{R} = (\hat{\theta} - \mathbb{E}[\hat{\theta}])/\sigma_T$ where $\sigma_T = \sqrt{\mathrm{Var}(\hat{\theta})}$ denotes the true standard deviation. The true standard errors σ_t have been estimated by using a Monte Carlo simulation with

100,000 runs. The distribution of $\mathcal{R}^*$ has been generated by using NAR-Sieve bootstrap based on NNs, ELMs and regularized ELMs.

All simulations are based on $N = 500$ Monte Carlo runs with time series of length T with $T \in \{300, 500, 1000, 2000\}$. The bootstrap distributions have been estimated using $B = 1,000$ bootstrap replicates. The final experiment is based on 384 design points (2 classes of models, 4 model specifications, 4 statistics, 4 different time series length and three different bootstrap implementations).

The results are reported in Figures 3 and 4. For all the statistics considered, the performance of the bootstrap based on ELMs is comparable with that of the bootstrap based on NNs or, even, slightly better in some cases. For linear functionals, such as the mean, the performance of the three methods is very close, with low variability, showing excellent accuracy when using the bootstrap to estimate the true unknown standard error. For nonlinear functionals, such as the median, the results are very similar and remain stable for the two class of models and different degrees of nonlinearity. For the case of complex functionals, which can be expressed as a function of means such as the variance and the covariance, the results remain consistent, but they show a lower accuracy for shorter time series.

Moreover, the degree of nonlinearity of the models might reduce the accuracy of the bootstrap standard error estimation. However, ELMs appear to deliver better results in these latter cases both in terms of accuracy and bias. Anyhow, using ELMs requires only a fraction of the computational time needed for NNs. Finally, all estimators appear to be consistent, with an apparent convergence to the reference value and a decreasing variability, while the time series length increases.

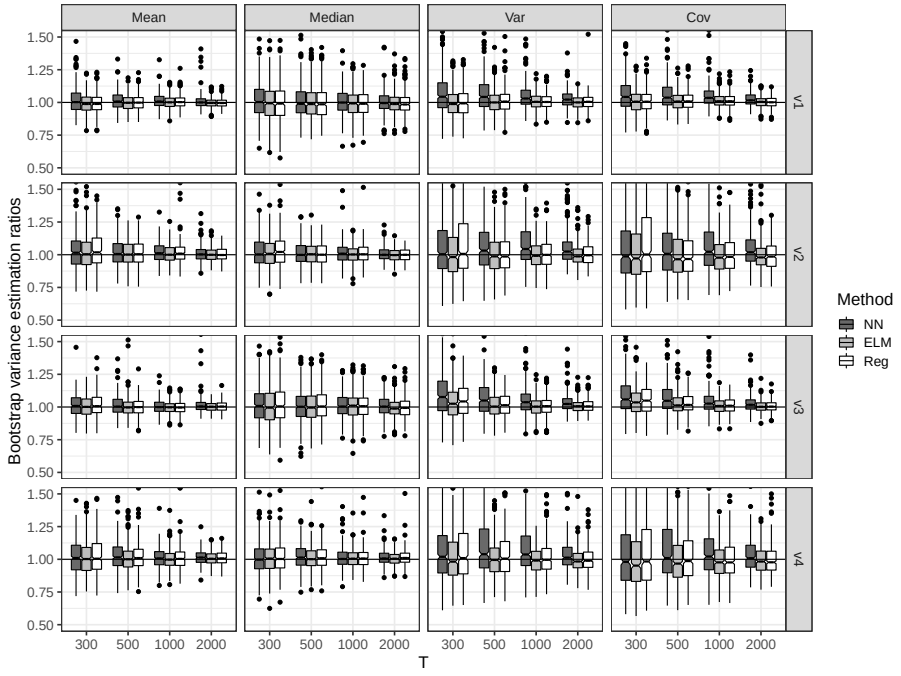

Figure 3. LSTAR model. Variance estimation of $(\hat{\theta} - \mathbb{E}[\hat{\theta}])/\sigma_T$ by $(\hat{\theta}^* - \mathbb{E}_*[\hat{\theta}^*])/\sigma_T$ [with $\sigma_T = \sqrt{\text{Var}(\hat{\theta})}$] for $\hat{\theta} = \{\text{mean}, \text{median}, \text{var}, \text{cov}\}$. Boxplots for NN-Sieve (NN), NN-sieve with ELM (ELM) and NN-sieve with ELM Ridge (Reg), with target indicated by the horizontal line. 500 simulation runs, 1000 bootstrap replicates per simulation run.

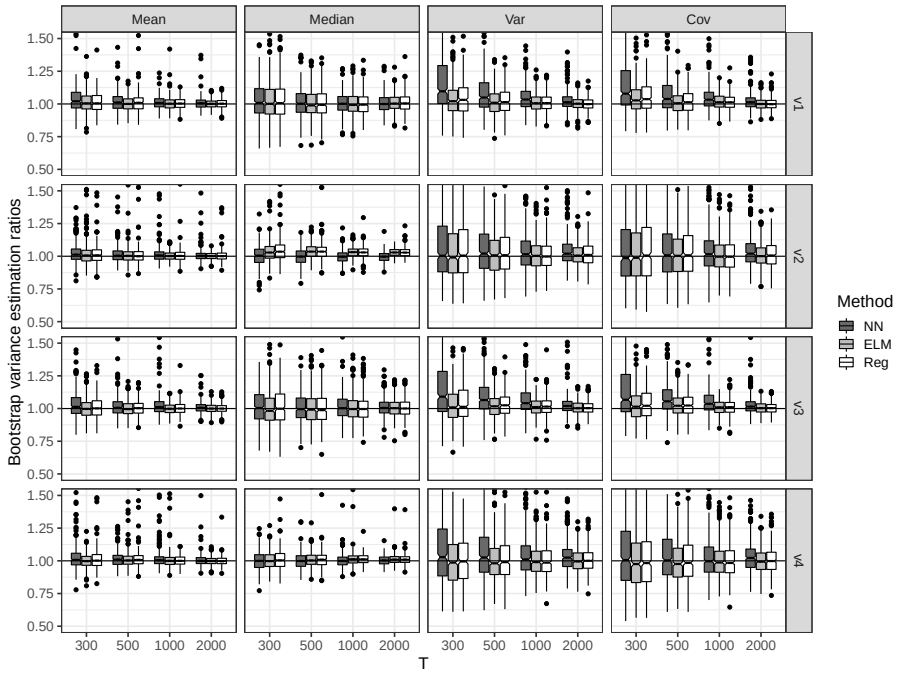

Figure 4. ESTAR model. Variance estimation of $(\hat{\theta}-\mathbb{E}[\hat{\theta}])/\sigma_T$ by $(\hat{\theta}^*-\mathbb{E}_*[\hat{\theta}^*])/\sigma_T$ [with $\sigma_T = \sqrt{\text{Var}(\hat{\theta})}$] for $\hat{\theta} = \{\text{mean}, \text{median}, \text{var}, \text{cov}\}$. Boxplots for NN-Sieve (NN), NN-sieve with ELM (ELM) and NN-sieve with ELM Ridge (Reg), with target indicated by the horizontal line. 500 simulation runs, 1000 bootstrap replicates per simulation run.

6. An application to real data

As an application to real data, we consider the normalized tree-ring widths in dimensionless units. The data were recorded by Donald A. Graybill, 1980, from Gt Basin Bristlecone Pine 2805M, 3726–11810 in Methuselah Walk, California. It is a univariate time series with 7981 yearly observations from 6000 BC to 1979. Tree-ring data are of great importance in ecology in general and in climate change studies in particular. Other fields of interest include archaeology (for dating materials and artefacts made from wood), chemists (where tree rings based methods are used to calibrate radiocarbon dates) and dendrology (which also includes forestry management and conservation).

In this application, we limit the analysis to the period ranging from 1001 to 1979 (about the last 1000 years). The time plot of the data and its autocorrelation function (for the first 20 lags) are reported in Figure 5. The data generating process appears to be stationary with a decreasing autocorrelation function where the first five lags are statistically significant at the level of 5%. To gain inference on the true autocorrelations, given the observed time series, the sampling distribution of the estimates is derived by using the sieve bootstrap based both on NNs and ELMs. The kernel density estimation of the bootstrap distributions for the first six lags is reported in Figure 6. Clearly, the bootstrap estimates seem to be able to capture the asymmetry of the true sampling distribution of autocorrelations. There is a slight difference between NN and ELM sieve bootstrap for the first lag, but in all other cases, the two approaches considered in the paper appear to deliver similar results.

Given the bootstrap sampling distribution, confidence intervals with nominal level equal to 95%

are derived and plotted in Figure 7 for the first 20 lags. Both procedures identify the first four lags as significant, while all other lags cannot be considered different from zero at the given confidence level. The different conclusion on lag five can be explained by the higher accuracy of the bootstrap-based inference with respect to the normal approximation used to construct the confidence intervals reported in Figure 5.

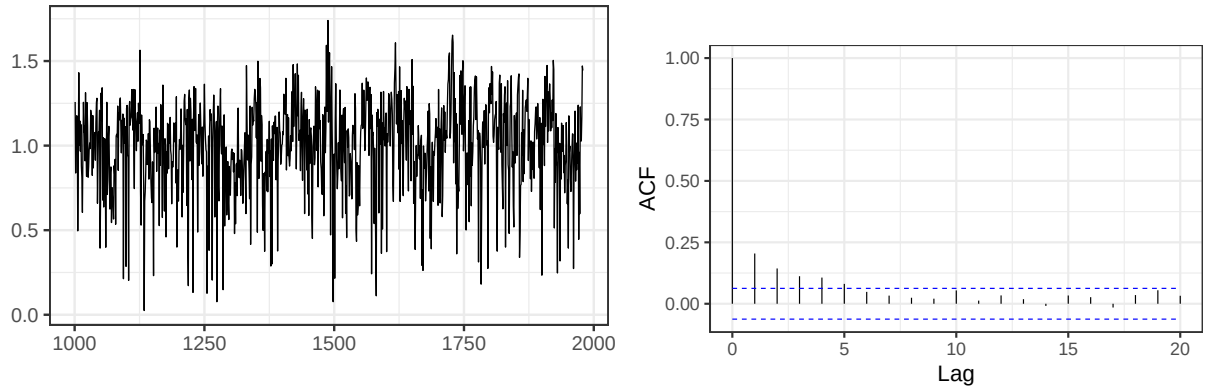

Figure 5. Tree-ring width yearly time series (on the left) from 1001 to 1979 and ACF (on the right) for the first 20 lags.

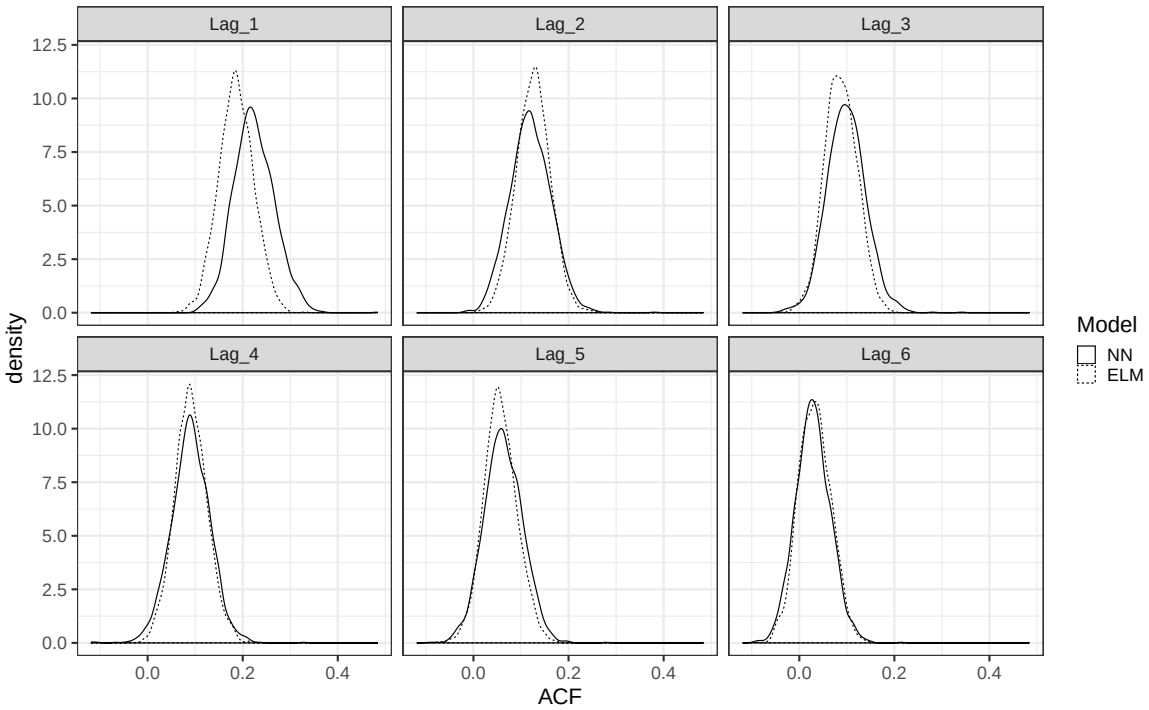

Figure 6. Kernel densities estimation of the bootstrap distributions (bootstrap runs = 1,999) obtained by using the sieve bootstrap based on neural networks (solid line) and extreme learning machines (dashed line) for the first 6 lags.

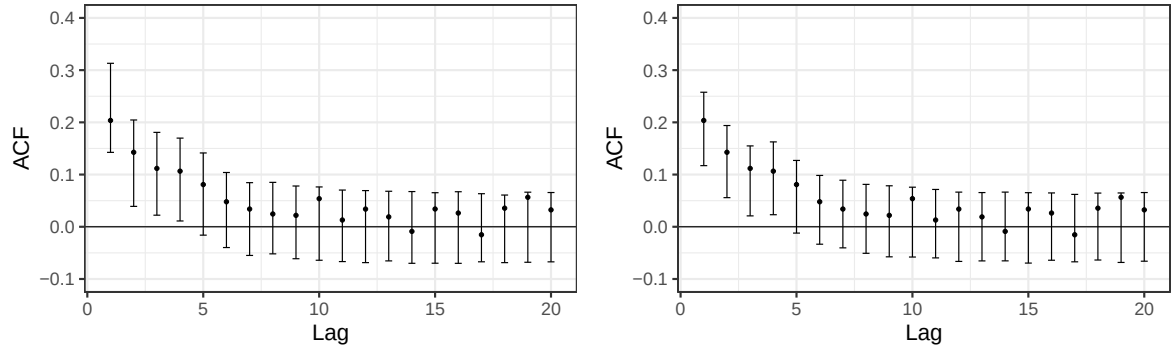

Figure 7. Percentile bootstrap confidence intervals (nominal level = 0.95, bootstrap runs=1,999) with sieve bootstrap based on neural networks (on the left) and based on extreme learning machines (on the right) for the first 20 lags.

7. Concluding remarks

In this paper, a novel nonlinear autoregressive sieve bootstrap scheme based on the use of ELMs have been proposed and discussed. To evaluate the performance of the proposed approach, a Monte Carlo simulation experiment has been implemented. Bootstrap schemes based on neural networks appear to be an encouraging solution to extend sieve bootstrap techniques to nonlinear time series. In this framework, ELMs can dramatically reduce the computational burden of the overall procedure, with performances comparable to the NN-Sieve bootstrap and computing time comparable to the AR-Sieve bootstrap.

Moreover, alternative algorithms for ELMs estimation can be considered. Although the orthogonal projection method can be efficiently used to calculate the Moore-Penrose inverse and the solution can be easily and fast obtained, regularized versions of least squares have proved to be stabler. These techniques regularize the coefficients (controlling how large they grow) by penalizing their magnitude along with minimizing the error between predicted and actual observations. Adding a penalty term can improve the stability of ELMs reducing the variance of the estimates. Even when using the regularized versions based on the Tsybakov (L_2) regularization the computational advantage of ELMs over classical NNs is maintained. Extensions of the bootstrap resampling scheme by using L_1 (such the LASSO) or a combination of L_1 and L_2 regularization (such as the elastic net) is still under study.

The performance of the sieve bootstrap based on ELMs depends on the Basic Linear Algebra System (BLAS) implementation. Using OpenBlas in place of the standard R library (even in single thread mode) can significantly improve the performance with better-scaling properties for bigger size problems. Other BLAS implementations that could be considered are the Intel's Math Kernel Library (MKL) and the ATLAS library which is a general purpose tunable library.

However, several different aspects should be further explored through a more extensive simulation study. These aspects include the sensitivity and the stability of the NAR-Sieve bootstrap to lag structure misspecification and to the choice of the hidden layer size. These topics are still under investigation and out of the scope of this paper. As a final remark, note that ELMs have been extended to deal with large size data problems effectively. The feasibility of bootstrap resampling schemes in this framework is still under investigation

Acknowledgements

The authors wish to thank the Associate Editor and the anonymous referees for their helpful comments and suggestions.

Conflict of interest

The authors declare no conflict of interest.

References

1. J. P. Kreiss, Bootstrap procedures for AR(∞)-processes, in *Bootstrapping and Related Techniques* (eds. K.-H. Jockel, G. Rothe and W. Sendler), Springer, Heidelberg, (1992), 107–113.
2. P. Bühlmann, Sieve bootstrap for time series, *Bernoulli*, **3** (1997), 123–148.
3. P. J. Bickel and P. Bühlmann, A new mixing notion and functional central limit theorems for a sieve bootstrap in time series, *Bernoulli*, **5** (1999), 413–446.
4. A. M. Alonso, D. Peña and J. Romo, Forecasting time series with sieve bootstrap, *J. Stat. Plann. Infer.*, **100** (2002), 1–11.
5. A. M. Alonso, D. Peña and J. Romo, On sieve bootstrap prediction intervals, *Stat. Probabili. Lett.*, **65** (2003), 13–20.
6. A. Zagdanski, On the construction and properties of bootstrap-t prediction intervals for stationary time series, *Probab. Math. Stati. PWN*, **25** (2005), 133–154.
7. A. M. Alonso and A. E. Sipols, A time series bootstrap procedure for interpolation intervals, *Comput. Stat. Data Anal.*, **52** (2008), 1792–1805.
8. P. Mukhopadhyay and V. A. Samaranayake, Prediction intervals for time series: a modified sieve bootstrap approach, *Commun. Stat. Simul. Comput.*, **39** (2010), 517–538.
9. G. Ulloa, H. Allende-Cid and H. Allende Robust sieve bootstrap prediction intervals for contaminated time series, *Int. J. Pattern Recognit. Artif. Intell.*, **28** (2014).
10. Y. Chang and J. Y. Park, A sieve bootstrap for the test of a unit root, *J. Time Ser. Anal.*, **24** (2003), 379–400.
11. Z. Psaradakis, Blockwise bootstrap testing for stationarity, *Stat. Probabili. Lett.*, **76** (2006), 562 –570.
12. D. S. Poskitt, Properties of the sieve bootstrap for fractionally integrated and non-invertible processes, *J. Time Ser. Anal.*, **29** (2008), 224–250.
13. D. S. Poskitt, G. M. Martin and S. D. Grose, Bias reduction of long memory parameter estimators via the pre-filtered sieve bootstrap, *arXiv preprint arXiv*, **2014** (2014).
14. E. Paparoditis, Sieve bootstrap for functional time series, *Ann. Stat.*, **46** (2018), 3510–3538.
15. M. Meyer, C. Jentsch and J. P. Kreiss Baxter's inequality and sieve bootstrap for random fields, *Bernoulli*, **23** (2017), 2988–3020.

16. J. P. Kreiss, E. Paparoditis and D. N. Politis, On the range of validity of the autoregressive sieve bootstrap, *Ann. Stat.*, **39** (2011), 2103–2130.

17. M. Fragkeskou and E. Paparoditis, Extending the Range of Validity of the Autoregressive (Sieve) Bootstrap, *J. Time Ser. Anal.*, **39** (2018), 356–379.

18. F. Giordano, M. La Rocca and C. Perna, Forecasting nonlinear time series with neural network sieve bootstrap, *Comput. Stat. Data Anal.*, **51** (2007), 3871–3884.

19. F. Giordano, M. La Rocca and C. Perna, Properties of the neural network sieve bootstrap, *J. Nonparametr. Stat.*, **23** (2011), 803–817.

20. G. B. Huang, Q. Y. Zhu and C. K. Siew, Extreme learning machine: theory and applications, *Neurocomputing*, **70** (2006), 489–501.

21. G. B. Huang, H. Zhou, X. Ding, et al., Extreme learning machine for regression and multiclass classification, *IEEE Trans. Syst. Man Cybern. Part B*, **42** (2012), 513–529.

22. W. Haerdle and A. Tsybakov, Local polynomial estimators of the volatility function in nonparametric autoregression, *J. Econometrics*, **81** (1997), 223–242.

23. J. Franke and M. Diagne, Estimating market risk with neural network, *Stat. Decisions*, **24** (2006), 233–253.

24. A. R. Barron, Universal approximation bounds for superpositions of a sigmoidal function, *IEEE Trans. Inf. Theory*, **39** (1993), 930–945.

25. K. Hornik, M. Stinchcombe and P. Auer, Degree of approximation results for feedforward networks approximating unknown mappings and their derivatives, *Neural Comput.*, **6** (1994), 1262–1275.

26. Y. Makovoz, Random approximates and neural networks, *J. Approximation Theory*, **85** (1994), 98–109.

27. X. Chen and H. White, Improved Rates and Asymptotic Normality for Nonparametric Neural Network Estimators, *IEEE Trans. Inf. Theory*, **45** (1999), 682–691.

28. X. Chen and X. Shen, Asymptotic Properties of Sieve Extremum Estimates for Weakly Dependent Data with Applications, *Econometrica*, **66** (1998), 299–315.

29. J. Zhang, Sieve Estimates via Neural Network for Strong Mixing Processes, *Stat. Inference Stochastic Processes*, **7** (2004), 115–135.

30. S. F. Crone and N. Kourentzes, Feature selection for time series prediction–A combined filter and wrapper approach for neural networks, *Neurocomputing*, **7** (2010), 1923–1936.

31. C. Wang, Y. Qi, M. Shao, et al., A fitting model for feature selection with fuzzy rough sets, *IEEE Trans. Fuzzy Syst.*, **25** (2017), 741–753.

32. D. Yu and L. Deng, Efficient and effective algorithms for training single hidden- layer neural networks, *Pattern Recognit. Lett.*, **33** (2012), 554–558.

33. K. Li, J. X. Peng and G. W. Irwin, A fast nonlinear model identification method, *IEEE Trans. Autom. Control*, **50** (2005), 1211–1216.

34. X. Yao, A review of evolutionary artificial neural networks, *Int. J. Intell. Syst.*, **8** (1993), 539–567.

35. G. B. Huang, D. H. Wang and Y. Lan, Extreme learning machines: a survey, *Int. J. Mach. Learn. Cybern.*, **2** (2011), 107–122.

36. S. Ding, H. Zhao, Y. Zhang, et al. Extreme learning machine: algorithm, theory and applications, *Artif. Intell. Rev.*, **44** (2015), 103–115.

37. G. Huang, G. B. Huang, S. Song, et al., Trends in extreme learning machines: A review, *Neural Networks*, **61** (2015), 32–48.

38. G. H. Huang, H. Zhou, X. Ding, et al., Extreme learning machine for regression and multiclass classification, *IEEE Trans. Syst. Man Cybern. Part B*, **42** (2012), 513–529.

39. G. B. Huang, L. Chen and C. K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, *IEEE Trans. Neural Networks*, **17** (2006), 879–892.

40. G. B. Huang and L. Chen, Convex incremental extreme learning machine, *Neurocomputing*, **70** (2007), 3056–3062.

41. G. B. Huang and L. Chen, Enhanced random search based incremental extreme learning machine, *Neurocomputing*, **71** (2008), 3460–3468.

42. J. Lin, J. Yin, Z. Cai, et al., A secure and practical mechanism of outsourcing extreme learning machine in cloud computing, *IEEE Intell. Syst.*, **28** (1999), 35–38.

43. E. Cule and S. Moritz, ridge: Ridge Regression with Automatic Selection of the Penalty Parameter, *R package version*, (2019), https://CRAN.R-project.org/package=ridge.

44. Z. Cai, J. Fan and Q. Yao, Functional-coefficient regression models for nonlinear time series, *J. Am. Stat. Assoc.*, **95** (2000), 941-956.

45. H. Kuswanto and P. Sibbertsen, Can we distinguish between common nonlinear time series models and long memory?, *Discussion papers//School of Economics and Management of the Hanover Leibniz University.*, (2007).

AIMS Press