Forecasting multivariate time series with boosted configuration networks

Thierry Moudiki

ISFA - Laboratoire SAF, Université Lyon I, 69007 Lyon, France

Abstract

This paper contributes to a development of Randomized Neural Networks, and more specifically to a development of the Stochastic Configuration Networks (SCNs). We present a family of learning algorithms based on the SCNs and on ensembles of single layer feedforward networks (SLFNs). They are close to Gradient Boosting, and to Matching Pursuit algorithms, and are denoted here as boosted configuration networks (BCNs). In the BCN framework, as with SCNs, the networks' hidden layers are chosen in a supervised way, ensuring that the universal approximation property of Neural Networks is met. Though, the learning mechanism of the BCNs incorporates a learning rate, that allows for a slower learning of the model's residuals. It also applies a subsampling of the models' explanatory variables that decorrelates the base learners. The BCNs are compared to other ensembles of Randomized Neural Networks and to other forecasting techniques, on real world multivariate time series data. An interesting direction for a future work would be to apply the BCNs to regression/classification data which are not time series.

Keywords: Neural Networks, Gradient Boosting, universal approximation property, Randomized Neural Networks, forecasting

2010 MSC: 00-01, 99-00

*Corresponding author

Email address: thierry.moudiki@gmail.com (Thierry Moudiki)

1. Introduction

The goal of ensemble learning is to combine astutely two or more individual statistical/machine learning models – the base learners – into one, with the expectation of an improved out-of-sample error of the ensemble over the base learners.

Gradient Boosting (see Friedman (2001), Bühlmann & Yu (2003), Hothorn et al. (2010)) is an ensemble learning procedure, whose general idea is to fit the model's in-sample residuals with the base learners, iteratively and slowly, and to stop learning those residuals just before the out-of-sample error of the model starts to increase.

In this paper, we discuss a family of statistical/machine learning algorithms based on the Stochastic Configuration Networks (SCNs) from Wang & Li (2017b), and on the boosting of Single Layer Feedforward Networks (SLFNs) (specifically on the LS_Boost from Friedman (2001)). We denote this family of algorithms as *Boosted Configuration Networks* (BCNs).

The idea of the SCNs from Wang & Li (2017b) followed some related works on constructive Neural Networks (Friedman & Stuetzle (1981), Jones et al. (1992), Barron (1993) and Kwok & Yeung (1997)) and constructive Random Basis Approximators (Igelnik & Pao (1995) and Li & Wang (2017)). The common philosophy behind these references was to construct learning algorithms for target functions incrementally, with bounded functions of the explanatory variables (of the target) as base learners, until the residual error of the resulting model falls under a certain level of tolerance. The universal approximation property (Hornik et al. (1989)); the ability of the constructive model to converge (in L_2) towards the target function as the number of base learners grows, is also a general concern examined in these references. Other resources containing insights on this body of work are Vincent & Bengio (2002) and Mallat & Zhang (1993), who put a greater emphasis on Matching Pursuit algorithms (Friedman & Stuetzle (1981)).

Wang & Li (2017b) introduced three different algorithms for SCN-learning, denoted as SC-I, SC-II, SC-III. In these algorithms, the base learners are SLFNs and each one of the three algorithms verifies the universal approximation property. SC-I does successive orthogonal projections of the target function on the base learners. The parameters of the base learners (nodes of the hidden layers) are chosen in a supervised way, ensuring that the universal approximation property of the additive *basis* expansion of SC-I is met. SC-III is a modified version of SC-I, in which all the *weights* of the target function in the additive expansion are recalculated, at each learning iteration of the residuals, and not just the *output weight* (the most recent weight of the additive expansion) as in SC-I.

SC-III provides the best results of the three algorithms presented in Wang & Li (2017b), but is said to be less suited than SC-I for large-scale data – notably because of its recalculation of all the weights of the expansion. SC-II mixes ingredients from SC-I and SC-III, to serve as a compromise between them, when both accuracy and scalability are needed. Another contribution to the SCN literature is Wang & Li (2017a), in which a version of the SCN is introduced; more robust to noisy samples and to outliers.

In the BCN framework introduced here, the SCNs from Wang & Li (2017b) are brought closer to the LS_Boost from (Friedman (2001)). The base learners $(\mathbf{x}, \mathbf{a}) \mapsto h(\mathbf{x}, \mathbf{a})$ (parametrized functions of the input variables $\mathbf{x}$, characterized by parameters $\mathbf{a} = \{a_1, a_2, \ldots\}$) of the LS_Boost are SLFNs, and the parameters $\mathbf{a}$ of the hidden layers are chosen in a supervised way that guarantees the universal approximation property of the BCN expansion. Contrary to the SCNs, and similarly to Gradient Boosting machines, the BCNs incorporate both a learning rate that allows for a slow learning of the residuals, and a subsampling of the models' explanatory variables that achieves an increased diversity of the base learners.

The contributions of this paper are thus twofold:

- define the BCN algorithms as supervised and automated ways of con-

structing new model's features, a more general version of SCNs, a Gradient Boosting procedure that verifies the universal approximation property of Neural Networks.

- employ the BCNs for multivariate time series (MTS) forecasting and compare it to other ensembles of randomized Neural Networks and usual forecasting techniques, on various real world MTS datasets.

The BCNs are described in details in Section 2, and their forecasting capabilities are examined in Section 3, on nine different MTS datasets. In Section 2, we notably demonstrate the universal approximation property of the BCNs, after including a learning rate and a subsampling of the covariates to the SCNs. We also discuss how some parameters of the model influence the convergence of the BCN expansion towards its target.

2. The boosted configuration networks

We start with a general remark that holds for all the models described in this paper, including the BCNs. For the construction of training/testing/validation sets, we always consider $p \in \mathbb{N}^*$ time series $(X_t^{(j)})_{t\geq 0}$ for $j \in \{1, \ldots, p\}$ observed at $n \in \mathbb{N}^*$ discrete dates, and we are interested in obtaining simultaneous forecasts of these time series at time $n + h$, $h \in \mathbb{N}^*$. Each series is allowed to be influenced by the others, in the spirit of VAR models (Lütkepohl (2005); Pfaff (2008)).

For the purpose of forecasting these p variables, $k < n$ lags of each time series are used. Therefore, the output variables to be explained are:

$$Y^{(j)} = \left(X_n^{(j)}, \ldots, X_{k+1}^{(j)}\right)^T \tag{1}$$

for $j \in \{1, \ldots, p\}$. $X_n^{(j)}$ is the most recent observed value of the j^{th} time series, and $X_{k+1}^{(j)}$ was observed k dates earlier in time for $(X_t^{(j)})_{t\geq 0}$. These

output variables are stored in a matrix:

$$\mathbf{Y} \in \mathbb{R}^{(n-k)\times p} \tag{2}$$

and the original explanatory variables (before any nonlinear transformation) are stored in a matrix:

$$\mathbf{X} \in \mathbb{R}^{(n-k)\times(k\times p)} \tag{3}$$

$\mathbf{X}$ consists of p blocks with k time series lags each. For example, the j_0^{th} block of $\mathbf{X}$, for $j_0 \in \{1,\ldots,p\}$ contains in columns:

$$\left(X_{n-i}^{(j_0)},\ldots,X_{k+1-i}^{(j_0)}\right)^T \tag{4}$$

with $i \in \{1,\ldots,k\}$. It is also possible to add other regressors, such as dummy variables, indicators of special events, but as in Moudiki et al. (2018), we consider only the inclusion of lags of the observed time series. For example, if we have observed $p = 2$ time series $(X_{t_1}^{(1)},\ldots,X_{t_5}^{(1)})$ and $(X_{t_1}^{(2)},\ldots,X_{t_5}^{(2)})$ at $n = 5$ dates $t_1 < \ldots < t_5$, and would like to use $k = 2$ lags to construct the explanatory variables, the response variables will be stored in:

$$\mathbf{Y} = \begin{pmatrix} X_{t_5}^{(1)} & X_{t_5}^{(2)} \\ X_{t_4}^{(1)} & X_{t_4}^{(2)} \\ X_{t_3}^{(1)} & X_{t_3}^{(2)} \end{pmatrix} \tag{5}$$

And the original explanatory variables will be stored in:

$$\mathbf{X} = \begin{pmatrix} X_{t_4}^{(1)} & X_{t_3}^{(1)} & X_{t_4}^{(2)} & X_{t_3}^{(2)} \\ X_{t_3}^{(1)} & X_{t_2}^{(1)} & X_{t_3}^{(2)} & X_{t_2}^{(2)} \\ X_{t_2}^{(1)} & X_{t_1}^{(1)} & X_{t_2}^{(2)} & X_{t_1}^{(2)} \end{pmatrix} \tag{6}$$

In the sequel of the paper, we will sometimes work on only one of the p response variables and denote it as $y \in \mathbb{R}^{(n-k)}$. Hence, the observation at time t_i of y will be denoted as y_i, and y_i will be explained as a function of $\mathbf{x}_i$, the i^{th} line of matrix $\mathbf{X}$. All of the p response variables do share the same set of

predictors, in a multitask learning (Caruana (1998)) fashion. Their treatment is thus equivalent.

Now, borrowing from some notations of Friedman (2001), the problem that we would like to solve is finding F^* that verifies:

$$F^* = arg\, min_F \mathbb{E}_{\mathbf{X}} \left[\mathbb{E}_y \left[L(y, F(\mathbf{X})) \right] | \mathbf{X} \right] \tag{7}$$

With $(y, \mathbf{X}) \mapsto L(y, F(\mathbf{X})) = (y - F(\mathbf{X}))^2$ being a squared-error loss function. Having these introductory informations and notations, we will now present the BCNs.

2.1. General description of the boosted configuration networks (BCNs)

In the SCN framework from Wang & Li (2017b) and similarly to simple Matching Pursuit algorithms (see Vincent & Bengio (2002) and Mallat & Zhang (1993)), it is assumed that an approximation F_{L-1} of F (observation y_i) has already been constructed as:

$$F_{L-1}(\mathbf{x}_i) = \beta_0 + \sum_{l=1}^{L-1} \beta_l g(\mathbf{x}_i^T w_l + b_l) \tag{8}$$

With $F_0 = \beta_0$. The $L-1$ functions $g_l : (\mathbf{x}_i; w_l, b_l) \mapsto g(\mathbf{x}_i^T w_l + b_l)$ for $l \in \{1, \ldots, L-1\}$, with their respective parameters, constitute the base learners of the additive expansion F_{L-1}. g is an activation function that transforms the linear inputs into nonlinear features. The output weight β_l, the parameters $w_l \in \mathbb{R}^{k*p}$ and the bias parameter $b_l \in \mathbb{R}^*$ are to be optimized, as it will be shown at the end of the section. The current residual error between F and F_{L-1} at step $L-1$ is given by:

$$e_{L-1} = F - F_{L-1} \tag{9}$$

In order to iterate from step $L-1$ to step L, if $||e_{L-1}||_F$ (the Frobenius norm of the residuals) still exceeds a given tolerance level ϵ, we need to generate a new base learner $g_L : (\mathbf{x}_i, w_L^*, b_L^*) \mapsto g(\mathbf{x}_i^T w_L^* + b_L^*)$. That is, we need to find

optimal w_L^* and b_L^*, and evaluate the output weight β_L^*, so that the norm of the residuals is decreased and:

$$F_L = F_{L-1} + \beta_L^* g_L \tag{10}$$

Wang & Li (2017b) showed that if the activation function g is bounded by a constant $b_g > 0$ ($0 < ||g||_{L^2} < b_g$), and the following condition is fulfilled for each of the p response variables (here, the most recent observations of the time series):

$$< e_{L-1,q}, g_L >_{L^2} \geq b_g^2 \delta_{L,q},\ q = 1, \ldots, p \tag{11}$$

Then, when $L \to \infty$, the additive expansion will converge towards F:

$$F_L \longrightarrow_{L^2} F \tag{12}$$

With $0 < r < 1$, $\mu_L = \frac{1-r}{L+1}$, and $\delta_{L,q} = (1 - r - \mu_L)||e_{L-1,q}||^2$, $q = 1, \ldots, p$.

In this paper, we use a slightly modified version of SCNs: the BCNs. The BCNs are close to Matching Pursuit algorithms, and possess two additional hyperparameters when compared to the SCNs:

- a learning rate $0 < \nu \leq 1$, so that the update at iteration L based on Eq. (10) is now:

$$F_L = F_{L-1} + \nu \beta_L^* g_L \tag{13}$$

- a subsampling percentage of the covariates, comprised between 0.5 and 1.

The learning rate ν will allow the model to learn the residuals slower than when ν is always equal to 1; whereas subsampling the covariates will increase the diversity of the base learners at each boosting iteration. Indeed, as mentioned in Friedman (2001), the learning rate controls the degree of fit. Considering it could be roughly seen as including smaller or larger steps in gradients' descent. The subsampling coefficient allows to randomly select a fraction of the covariates

at each boosting iteration, and reduce overfitting. If it's equal to 0.5, then only one half of the covariates are randomly selected at each boosting iteration. It must be chosen in such a way that, the number of covariates at each boosting iteration is always positive.

In Section 2.2, based on SCN framework, we show that the universal approximation property of the BCNs is verified.

2.2. Universal Approximation property of the boosted configuration networks (BCNs)

Including a learning rate ν to the SCN algorithms (Wang & Li (2017b)), in order to obtain the BCNs, leads to consider a new condition for the convergence of F_L towards F:

$$< e_{L-1,q}, g_L >_{L^2} \geq \frac{1}{\nu(2-\nu)} b_g^2 \delta_{L,q},\ q = 1, \ldots, p \tag{14}$$

With $0 < r < 1$, $\mu_L = \frac{1-r}{L+1}$, and $\delta_{L,q} = (1 - r - \mu_L)||e_{L-1,q}||^2$, $q = 1, \ldots, p$. Indeed, in this case, updating the residuals from step $L-1$ to step L is done by obtaining:

$$e_{L,q} = e_{L-1,q} - \nu \beta_{L,q} g_L,\ q = 1, \ldots, p \tag{15}$$

Consequently, by defining $\delta_L = \sum_{q=1}^{p} \delta_{L,q}$, and following the steps of the proof of Theorem 6 in Wang & Li (2017b), we have for the SC-I:

Proof.

$$\begin{aligned}
||e_L||^2 - (r + \mu_L)||e_{L-1}||^2 &= (1 - r - \mu_L)||e_{L-1}||^2 - \nu(2-\nu) \sum_{q=1}^{p} \frac{< e_{L-1,q}, g_L >_{L^2}^2}{||g_L||^2} \\
&= \delta_L - \nu(2-\nu) \sum_{q=1}^{p} \frac{< e_{L-1,q}, g_L >_{L^2}^2}{||g_L||^2} \\
&\leq \delta_L - \nu(2-\nu) \sum_{q=1}^{p} \frac{< e_{L-1,q}, g_L >_{L^2}^2}{b_g^2} \\
&\leq 0
\end{aligned}$$

□

$g : x \mapsto tanh(x)$ is used as an activation function for all the models presented in the paper. As required by the universal approximation property of the SCNs, $g : x \mapsto tanh(x)$ is indeed bounded.

We observe that this proof of the universal approximation property only depends on the boundedness of g and the condition from Eq. (14) (given $0 < r < 1$ and L sufficiently high). Therefore, it still holds when the columns of $\mathbf{x}_i$ are subsampled, as long as the condition from Eq. (14) is met. Similarly, for the adaptation of SC-III to the BCN framework, the universal approximation property holds, by using analogous arguments and Theorem 7 from Wang & Li (2017b).

The algorithms that we used for implementing the BCNs are described in the next section, 2.3. There would be three versions of BCN algorithms as in the SCN framework, but we only describe the adaptation of SC-I. SC-II and SC-III implementations would follow similar ideas.

2.3. Algorithm for the boosted configuration networks (BCNs)

Having chosen $0 < r < 1$; $\lambda > 0$: a regularization parameter for w_l and b_l; B: the budget number of boosting iterations; ϵ: a tolerance level for the Frobenius norm of the successive matrices of residuals, algorithm SC-I from Wang & Li (2017b) is hence modified as indicated in algorithm 1. With the following notations, relying on the original notations from Wang & Li (2017b):

$$e_{L-1}(\mathbf{X}) = [e_{L-1,1}(\mathbf{X}) \dots e_{L-1,p}(\mathbf{X})] \tag{16}$$

and $e_{L-1}(\mathbf{X}) \in \mathbb{R}^{(n-k)\times p}$. And:

$$e_{L-1,q}(\mathbf{X}) = (e_{L-1,q}(\mathbf{x}_1), \dots, e_{L-1,q}(\mathbf{x}_{n-k}))^T \tag{17}$$

$e_{L-1,q}(\mathbf{X}) \in \mathbb{R}^{n-k}$ for $q = 1, \dots, p$. Plus:

$$h_L(\mathbf{X}) = \left(g(\mathbf{x}_1^T w_L), \dots, g(\mathbf{x}_{n-k}^T w_L)\right)^T \tag{18}$$

$h_L(\mathbf{X}) \in \mathbb{R}^{n-k}$, and:

$$\xi_{L,q} = \nu(2-\nu)\frac{\left(e_{L-1,q}(\mathbf{X})^T h_L(\mathbf{X})\right)^2}{h_L(\mathbf{X})^T h_L(\mathbf{X})} - (1 - r - \mu_L) e_{L-1,q}(\mathbf{X})^T e_{L-1,q}(\mathbf{X}) \tag{19}$$

Algorithm 1 Algorithm for BCN learning

1: **procedure** BCN-I($\mathbf{Y}, \mathbf{X}, B, \eta, \lambda, r, \epsilon$) ▷ BCN learning of $\mathbf{Y}$ by $\mathbf{X}$

2: $e_0 \leftarrow \mathbf{Y}$; $\beta_0 \leftarrow \bar{\mathbf{Y}}$; $e_0 \leftarrow e_0 - \beta_0$; $L = 1$; ▷ Initialization

3: **while** $L \leq B$ and $||e_0||_F > \epsilon$ **do** ▷ Loop until tolerance is reached

4: Obtain

$$w_L^*, b_L^* \leftarrow arg\, max_{w_L, b_L \in [-\lambda, \lambda]^{k*p}} \left[\left(\sum_{q=1}^{p} \xi_{L,q} \right) \mathbb{1}_{min(\xi_{L,1}, \ldots, \xi_{L,p}) \geq 0} \right]$$

5: Obtain h_L^*, $\xi_{L,q}^*$, based on Eq. (18) and (19), and $\mu_L := (1-r)/(L+1)$

6: Do

$$\beta_L^* \leftarrow \texttt{lsfit}(\texttt{response} = e_{L-1}, \texttt{covariates} = h_L^*)$$

▷ least squares regression of e_{L-1} on h_L^*: obtain β_L^*, output weight.

7: $e_L \leftarrow e_{L-1} - \nu \beta_L^* h_L^*$ ▷ Update the residuals.

8: $e_0 \leftarrow e_L; L \leftarrow L + 1$ ▷ Next iteration.

In algorithm (1), Step 4 is typically achieved by using a derivative-free optimization. In Section 2.4, we discuss the convergence rate of the BCN expansion towards its target.

2.4. On the convergence of boosted configuration networks (BCNs)

In Section 2.1, we showed that for fixed $L > 0$, $0 < r < 1$, and under condition 14, we have:

$$||e_L||^2 \leq (r + \mu_L)||e_{L-1}||^2 \tag{20}$$

Because the target function to be explained is assumed to be square-integrable, this leads to the following inequality:

$$||e_L||^2 \leq C(r + \mu_L)^L \tag{21}$$

Where $C > 0$ is a positive constant. A sufficient condition for the conver-

gence of F_L towards F (in L^2) and $||e_L||^2$ towards 0 when $L \to \infty$ is:

$$0 < r + \mu_L < 1 \tag{22}$$

Which is always true when $0 < r < 1$ and $L > 0$. In figure 1, we present the function $(r, L) \mapsto (r + \mu_L)^L$; the convergence rate of $||e_L||^2$ towards 0, for 100 values or $r \in [0.01, 1]$, and 25 values of $L \in \{1, 2, 3, \ldots, 25\}$

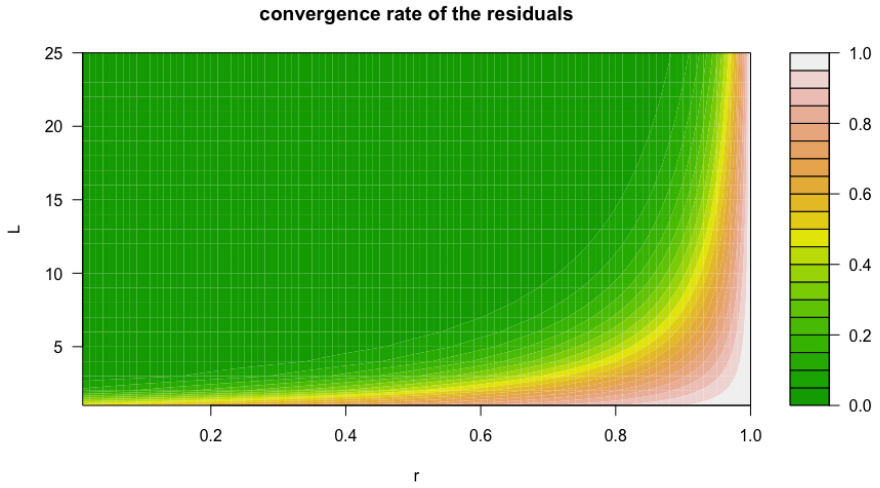

Figure 1: Convergence rate of the residuals of BCNs towards 0, as a function of r and L.

Figure 1 suggests that a very high value of L would not necessarily be required for the convergence of the residuals to occur. r must be chosen adequately: relatively close to 1 to prevent overfitting, but *not too close*, to avoid a divergence of the BCN expansion. r and L are both chosen along with the other hyperparameters of the BCNs by cross-validation, as demonstrated in Section 3.

3. Numerical examples

In this section, we examine the forecasting capabilities of the BCNs on 9 real world MTS datasets, presented in Section 3.2. Each dataset was partitioned into

two parts:

1. A training/testing containing 75% of the observations.
2. A validation set that contains the remaining 25% of the observations, unseen at point 1, and used to assess the out-of-sample accuracy.

Based on the 9 datasets presented in Section 3.2, we compare the out-of-sample accuracy of the BCNs (on the validation set) to:

- the accuracy of two *naive* forecasting methods: random walk and sample mean.
- the accuracy of other models based on Randomized Neural Networks.
- the accuracy of other usual forecasting techniques.

All these competing models are described in Section 3.1. Their *optimal* hyperparameters, used to produce forecasts on the validation set, are obtained through a rolling forecasting methodology (Bergmeir et al. (2015)) applied to the training/testing set.

The rolling forecasting methodology proceeds as follows: a fixed window of length `training` is set for training the model, and another window, contiguous to the first one, and of length `testing` is set for testing it. The origin of the training set is then advanced of 1 observation, and the training/testing procedure is repeated until no more training/testing set can be constructed.

Typically, for all the competing models we used a fixed rolling window of length `training` = 18 points for the training, and testing sets of increasing lengths `testing` = 3, 6, 9, and 12 points. The measure of accuracy comparing a forecast $\hat{y}$ to the observed data y for each series is the Root Mean Squared Error (RMSE):

$$RMSE = \sqrt{\frac{1}{H}\sum_{h=1}^{H}\left(y_{n+h} - \hat{y}_{n+h}\right)^2} \tag{23}$$

For the optimization of the training/testing RMSE, we use a bayesian optimization algorithm of each model's hyperparameters. The bayesian optimization algorithm employed is the one described as `GP EI Opt` in Snoek et al. (2012), with 250 iterations and 10 repeats. The bounds for `GP EI Opt`'s hyperparameters search are those given in appendix 5.1, and the detailed cross-validation results can be found at `https://github.com/thierrymoudiki/ins-sirn-2018`. The rolling forecasting methodology from (Bergmeir et al. (2015)) is presented in figure 2:

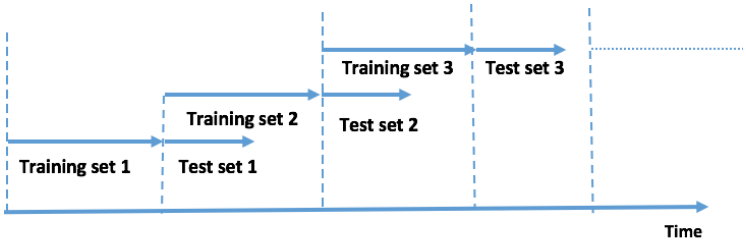

Figure 2: Time series cross-validation procedure illustrated.

3.1. Competing models

The first forecasting models that we compared to the BCNs (BCN-I and BCN-III) on MTS forecasting are two *naive* ones: a random walk denoted here and in the Github repo as `rw`, and a sample mean denoted as `mean`. The random walk would obtain h-steps-ahead forecasts of a time series by using its last observed value:

$$\hat{y}_{n+h} = y_n, \; h > 0 \tag{24}$$

Whereas for the sample mean, we would have for each time series:

$$\hat{y}_{n+h} = \bar{y}_n,\ h > 0 \tag{25}$$

The sample mean is calculated on rolling windows of length `training`. Then, other popular models were considered: an unrestricted VAR (Lütkepohl (2005); Pfaff (2008)) denoted as `VAR` and a Lasso VAR (see Cavalcante et al. (2017), Davidson et al. (2004), Nicholson et al. (2014)) denoted as `lassoVAR`. As a Lasso VAR model, we used Fu (1998) to implement the row Lasso VAR (rLV) model presented with details in Cavalcante et al. (2017).

To finish, three models based on Randomized Neural Networks and ensembles of Randomized Neural Networks were considered:

- The MTS forecasting model from Moudiki et al. (2018), based on Quasi-Randomized Neural Networks, and denoted here as `ridge2`.

- A model based on Hothorn et al. (2010) and Hothorn et al. (2017), applied to MTS forecasting, denoted as `glmboost`. In this model (using ideas from Section 2.1), after B boosting iterations, F_B has the form:

$$F_B(\mathbf{x}_i) = \beta_0 + \sum_{l=1}^{B} \sum_{m=1}^{M} \nu \beta_l^{(m)} g_l(\mathbf{x}_i; W_l^{(m)}, b_l^{(m)}) \tag{26}$$

  $W_l \in \mathbb{R}^{(k*p)\times M}$, for each boosting iteration $l \in \{1, \ldots, B\}$, are matrices of model's parameters drawn from a quasi-random Sobol sequence (see Niederreiter (1992) and Joe & Kuo (2008)), as in Moudiki et al. (2018). These matrices W_l, with columns $W_l^{(m)}$, give the number of nodes in the hidden layer (which is equal to M, and denoted as `nb_hidden`), and create a new set of M features from $\mathbf{x}_i$. The number of nodes in the hidden layer M, is an hyperparameter of the whole procedure (fixed in advance), and the functions $(\mathbf{x}_i; W_l, b_l) \mapsto g_l(\mathbf{x}_i; W_l, b_l)$ are the base learners. The

general form of g_l is:

$$g_l : (\mathbf{x}_i, W_l, b_l) \mapsto g(\mathbf{x}_i^T W_l + b_l) \tag{27}$$

Where g is the hyperbolic tangent (tanh), because the SCNs and BCNs require a bounded activation function for their convergence.

- The Partial Least Squares (PLS) algorithm (see Friedman et al. (2001)) denoted as `pls`. As stated in Friedman et al. (2001): the PLS seeks directions that have high variance and high correlation with the response, contrary to principal components regression which focuses only on the variance. If we denote $\mathbf{T}$ the matrix of explanatory variables with M columns, and y the response variable, then for the PLS we have:

  1. Standardize each column of $\mathbf{T}$ to have mean 0 and variance 1. Set $\hat{y}_0 = \bar{y}\mathbb{1}$ and $\mathbf{T}_0^{(j)} = \mathbf{T}^{(j)}$, $j = 1, \ldots, M$ (the columns of $\mathbf{T}$ are $\mathbf{T}^{(j)}, j = 1, \ldots, M$)
  2. For $m = 1, \ldots, M$
     - $\mathbf{z}_m = \sum_{j=1}^{M} \hat{\phi}_{mj} \mathbf{T}_{m-1}^{(j)}$, where $\hat{\phi}_{mj} = < \mathbf{T}_{m-1}^{(j)}, y >$
     - $\hat{\theta}_m = \frac{<\mathbf{z}_m, y>}{||\mathbf{z}_m||^2}$
     - $\hat{y}_m = \hat{y}_{m-1} + \hat{\theta}_m \mathbf{z}_m$
  3. Output the sequence of fitted vectors $\{\hat{y}_m\}_{m=1}^{p}$

  The algorithm can be stopped before the loop on m (the number of PLS directions) reaches M. Here, we use the SIMPLS algorithm from De Jong (1993) to a set of new features obtained by transformations of the original ones:

$$g(\mathbf{x}_i^T W + b) \tag{28}$$

  We start by choosing a fixed number of nodes in the hidden layer M (denoted as `nb_hidden` in the results), then obtain a quasi-random Sobol sequence for $W \in \mathbb{R}^{(k*p)\times M}$. A new set of M features are obtained with formula 28, and stored in a matrix $\mathbf{T}$, the matrix of transformed predictors. M is an hyperparameter for the algorithm, and the PLS is applied to $\mathbf{T}$.

3.2. Datasets used for benchmarking the models

9 datasets are used for comparing the algorithms from Section 3.1 to the BCNs:

- The `usconsumption` dataset. 2 series, 164 observations. The quaterly percentage changes of real personal expenditure and the real personal disposable income in the United States, from March 1970 to December 2010. The source for this dataset is the Federal Reserve Bank of St Louis: `http://data.is/AnVtzB` and `http://data.is/wQPcjU`. `usconsumption` is also available in R package **fpp** (Hyndman (2013)). No transformation is applied to the dataset for stationarity.

- The `Canada` dataset. 4 series, 83 observations of economic indicators observed in Canada from 1980 (first quarter) to 2000 (fourth quarter). The source for this dataset is the OECD: `http://www.oecd.org`. `Canada` is also available in R package **vars** (Pfaff (2008)). We transform each one of the original time series $(I_t^{(i)})_t, j = 1, \ldots, 4$ as $log\left(I_{t+1}^{(j)}/I_t^{(j)}\right), j = 1, \ldots, 4$.

- The `ausmacro` dataset used in Jiang et al. (2017). 35 series, 121 observations of macroeconomic indicators, available at `http://ausmacrodata.org/research.php` (accessed on August 4th, 2018). No transformation is applied to the dataset for stationarity.

- `usexp` a dataset from Makridakis et al. (2008). 2 series, 87 observations. Available in R package **fma** under the name `capital`: Seasonally adjusted quarterly capital expenditure and appropriations in U.S. manufacturing from 1953 to 1974. The times series are transformed as `Canada` for stationarity.

- `germancons` dataset. 3 series, 91 observations. Quarterly seasonally adjusted West German Fixed investment Disposable income and Consumption expenditures in billions of Deutsche Marks, from March 1960 to December 1982. Available at `https://datamarket.com` (accessed on August 4th, 2018). The times series are transformed as `Canada` for stationarity.

- Table F2.2: 10 series, 51 observations, the U.S. Gasoline Market from 1953 to 2004 (`tableF2_2` in the results and Github data). And Table F5.2: 9 series, 203 observations, macroeconomics data set, 1950I to 2000IV (`tableF5_2` in the results and Github data). Table F2.2 and Table F5.2 datasets both come from the 7^{th} and 8^{th} edition of Greene (2003), and are available at `http://pages.stern.nyu.edu/~wgreene/Text/econometricanalysis.htm` (accessed on August 4th, 2018) with their respective sources. The times series are transformed as `Canada` for stationarity.

- `housing`: 3 series, 81 observations. Monthly housing starts, construction contracts and average new home mortgage rates (from January 1983 to October 1989). Available in `R` package **fma**. The times series are transformed as `Canada` for stationarity.

- `ips`. 7 series, 575 observations. Industrial production data in the US, from 1959:I to 2006:IV. Obtained from Stock & Watson (2009) and the Global Insights Basic Economics Database, with the content column labeled 'E.F.?' equal to 1. The times series are transformed as `Canada` for stationarity.

3.3. Results

This section presents the rankings of the competing models based on their out-of-sample RMSE, and Diebold-Mariano tests (Diebold & Mariano (1995)) of the forecasting accuracies.

3.3.1. Ranking the models by out-of-sample RMSE

Once the *optimal* hyperparameters are chosen by cross validation on the training/testing set (75%) for each dataset, they are used on the validation set (25%) to determine which model has the lowest RMSE on unseen data. The detailed cross-validation results on training, testing and validation set, can be found at `https://github.com/thierrymoudiki/ins-sirn-2018`.

Since the datasets are not on the same scale, the RMSEs are not directly comparable accross the competing models. Instead, we compare the rank of each model among the eight others, based on the out-of-sample RMSE. In table 1 and figure 3, we report the average rankings of each model calculated on all the 9 datasets.

Table 1: Average RMSE rankings and standard deviation on the 9 validation sets

	horizon = 3	horizon = 6	horizon = 9	horizon = 12	Avg. rank
glmboost	5.11 +/- 1.90	5.33 +/- 2.55	5.88 +/- 2.37	4.88 +/- 2.62	5.30
lassoVAR	4.44 +/- 2.55	4.22 +/- 1.09	3.77 +/- **1.39**	3.66 +/- **1.50**	4.03
mean	4.00 +/- 2.87	4.11 +/- 2.09	**3.11** +/- 2.31	**3.00** +/- 2.24	3.55
pls	4.55 +/- 1.88	4.66 +/- 2.64	6.11 +/- 1.54	6.22 +/- 2.33	5.39
ridge2	**2.66** +/- 1.50	**2.88** +/- 2.15	3.88 +/- 2.20	4.00 +/- 1.58	**3.36**
rw	7.00 +/- 3.16	8.44 +/- **1.01**	8.11 +/- 1.96	7.88 +/- 2.62	7.86
bcnI	5.44 +/- 2.13	4.66 +/- 2.06	3.55 +/- 1.94	3.44 +/- 1.59	4.28
bcnIII	4.00 +/- 2.18	3.55 +/- 2.24	3.77 +/- 1.92	4.66 +/- 2.24	4.00
VAR	7.77 +/- **1.30**	7.11 +/- 2.42	6.77 +/- 2.86	7.22 +/- 1.99	7.22

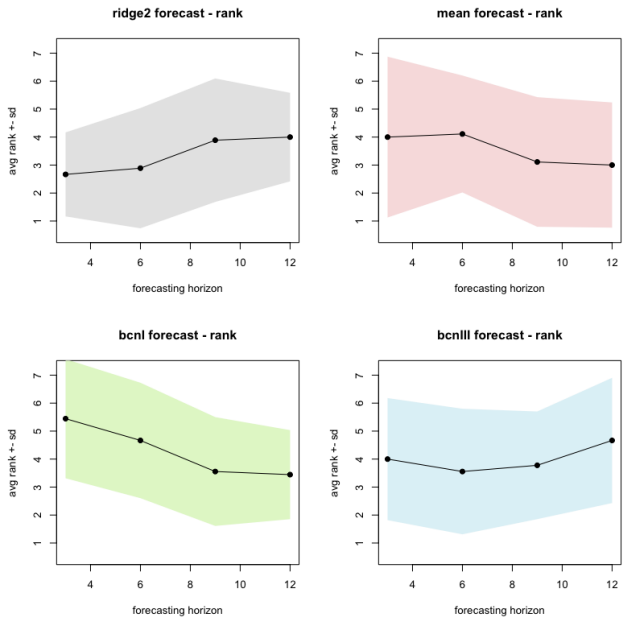

Figure 3: Average RMSE rankings and standard deviation for `ridge2`, `mean`, `bcnI`, `bcnIII`

Based on the results in table 1, no method is uniformly superior to the other. For short-term forecasts, `ridge2` (Moudiki et al. (2018)) is doing better than the other methods, and is followed by `bcnIII`. The sample mean is winning on long-term forecasts, and is followed by `bcnI`. The `bcn` models are therefore highly competitive on time series data.

The unrestricted VAR is well-known to overfit the data, because its learning mechanism uses a lot of parameters that are not constrained. But the lassoVAR largely improves on it, by forcing some of its parameters to be equal to 0.

Overall, `ridge2` has the best average rank of all the methods. And the very good rank obtained by the `mean` confirms some observations of Jiang et al. (2017), who remarked that it is difficult to outperform the naive sample mean, especially on long-term forecasts here.

Though, contrary to `ridge2` or `lassoVAR` for example, the sample mean model provides almost no insights to the analyst (on how the covariates influence

the response, how a shock on a covariate would affect the response, etc.), and mostly serves as a benchmark.

It might be beneficial to construct ensembles combining methods that perform well on short-term forecasts, `ridge2` and `bcnIII`, with methods that perform well on long-term forecasts, `mean` and `bcnI`.

3.3.2. Diebold-Mariano tests of `bcnIII` vs `mean`

In this section, we compare the forecasts of `bcnIII` and `mean` on two datasets:

- a dataset on which `bcnIII` has a better average rank than `mean`: `usexp`
- a dataset on which `mean` has a better average rank than `bcnIII`: `ips`

Details on the rank of each model among the competing models presented in Section 3.1 can be found at `https://github.com/thierrymoudiki/ins-sirn-2018`.

In order to compare the forecasts of `bcnIII` and `mean`, we use the Diebold-Mariano test (Diebold & Mariano (1995)) on residuals of the validation set, and obtain some information about the significance of the difference between model's forecast (at 5% and 10%). Negative values of the statistic in tables 2 and 3 indicate superiority of `bcnIII` forecasts over the sample mean forecasts. The null hypothesis is that the two forecasts have, on average, the same accuracy. One asterisk denotes significance relative to the asymptotic null distribution at the 10% level, and two asterisks denote significance relative to the asymptotic null distribution at the 5% level.

Table 2: DM statistic for out-of-sample forecasting accuracy comparison, on `usexp`

series	`horizon` = 3	`horizon` = 6	`horizon` = 9	`horizon` = 12
`Appropriations`	0.04	-0.11	-3.38**	-1.88*
`Expenditure`	-1.88*	0.79	-0.75	-2.05**

Table 3: DM statistic for out-of-sample forecasting accuracy comparison, on `ips`

series	`horizon` = 3	`horizon` = 6	`horizon` = 9	`horizon` = 12
`IPS13`	0.01	0.62	0.86	1.25
`IPS18`	1.81*	2.29**	2.40**	3.33**
`IPS25`	-1.03	-1.58	-1.83*	-1.24
`IPS34`	-0.68	-0.37	0.15	0.86
`IPS38`	0.66	1.86*	2.52**	3.00**
`IPS43`	-0.77	-0.70	-0.39	0.35
`IPS306`	1.78*	2.38**	2.80**	3.61**

Looking at each dataset/series individually can lead to more nuanced observations than reported in Section 3.3.1. Typically on `usexp`, `bcnIII` forecasts are always superior to the `mean`'s for long-term horizons; the difference is significant 3 times out of 4. Also, on `ips` for `IPS25` series, `bcnIII` is always superior to `mean`; the two forecasts being significantly different for `horizon` = 9.

4. Conclusion

In this paper, we discussed the Boosted Configuration Networks (BCNs) as a learning algorithm derived from the Stochastic Configuration Networks (SCNs) (Wang & Li (2017b)), the LS_Boost from (Friedman (2001)), and Matching Pursuit algorithms (Vincent & Bengio (2002)). The hidden layers of the BCNs are chosen in a supervised way, to ensure that the universal approximation property of Neural Networks is met, as with the SCNs. But contrary to the SCNs, the BCNs incorporates both a learning rate that allows for a slow learning of the residuals, and a subsampling of the models' explanatory variables that decorrelates the predictors and reduces overfitting. The optimization of the hidden layers is also carried out by using derivative-free optimization.

The results obtained here on various multivariate time series datasets by the BCNs are promising. Interestingly, the overall performance of the BCNs on these specific datasets is superior to the performance of unrestricted VAR and lasso VAR. As mentioned in Exterkate et al. (2016) and the conclusion of Makridakis et al. (2018) though, time series data are a very specific type of data, in which there is a serial dependence between the observations, and a strong correlation between the covariates. These characteristics could lead relatively complex models to fail, whereas relatively simple models would produce accurate forecasts. This could be part of the reason why the BCNs are not doing as good as the model from Moudiki et al. (2018) on these datasets, except in given cases (based on out-of-sample RMSE ranks).

In a future work, it could be interesting to test other bounded activation functions against the hyperbolic tangent. In addition, combining methods that performed well here on short-term forecasts (`ridge2` and `bcnIII`), with methods that performed well on long-term forecasts (`mean` and `bcnI`) could be beneficial. Finally, we would also want to assess how well the BCNs do on regression/classification problems not based on time series data.

5. Appendix

5.1. Bounds for hyperparameters search

For choosing the hyperparameters, we use Bayesian Optimization (Snoek et al. (2012)'s `GP EI Opt`), with the following bounds for each model:

Table 4: Bounds for hyperparameters search: `glmboost`

	B	ν	$lags$	`nb_hidden`
Lower bound	1	0.01	1	2
Upper bound	10	0.5	4	100

Table 5: Bounds for hyperparameters search: `lassoVAR`

	$lags$	λ
Lower bound	1	1e-02
Upper bound	4	1e04

Table 6: Bounds for hyperparameters search: `pls`

	B	$lags$	`nb_hidden`
Lower bound	1	1	2
Upper bound	10	4	100

Table 7: Bounds for hyperparameters search: `ridge2`

	$lags$	`nb_hidden`	λ_1	λ_2
Lower bound	1	2	1e-02	1e-02
Upper bound	4	100	1e04	1e04

Table 8: Bounds for hyperparameters search: `bcn`

	B	$lags$	ν	λ	r	ϵ	`col_sample`
Lower bound	2	1	0.01	1e-02	0.8	1e-06	0.5
Upper bound	10	4	0.5	1e04	0.99	1e-02	1

Where:

- B: number of boosting iterations for `glmboost`, `bcnI`, `bcnIII`.
- ν: learning rate for `glmboost`, `bcnI`, `bcnIII`.
- $lags$: number of lags of each time series included in the regression (see details at the beginning of Section 2).
- `nb_hidden`: number of nodes in the hidden layer (M in Section 2) for `ridge2`, `glmboost`, `pls`.
- λ: regularization parameter for `lassoVAR` (see rLV in Cavalcante et al. (2017)), `bcnI`, `bcnIII` (see algorithm 1).
- λ_1, λ_2: regularization parameters for `ridge2` (see Moudiki et al. (2018)).
- ϵ: level of tolerance for the Frobenius norm of residuals, for `bcnI`, `bcnIII`.
- `col_sample`: percentage of the covariates used at each boosting iteration, for `bcnI`, `bcnIII`.

The detailed cross-validation results can be found at `https://github.com/thierrymoudiki/ins-sirn-2018`.

References

Barron, A. R. (1993). Universal Approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, *39*, 930–945.

Bergmeir, C., Hyndman, R. J., Koo, B. et al. (2015). A note on the validity of cross-validation for evaluating Time Series prediction. *Monash University, Department of Econometrics and Business Statistics, Tech. Rep.*, .

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

Caruana, R. (1998). Multitask Learning. In *Learning to Learn* (pp. 95–133). Springer.

Cavalcante, L., Bessa, R. J., Reis, M., & Browell, J. (2017). Lasso Vector Autoregression structures for very short-term wind power forecasting. *Wind Energy*, *20*, 657–675.

Davidson, R., MacKinnon, J. G. et al. (2004). *Econometric Theory and Methods* volume 5. Oxford University Press New York.

De Jong, S. (1993). SIMPLS: an alternative approach to Partial Least Squares Regression. *Chemometrics and intelligent laboratory systems*, *18*, 251–263.

Diebold, F., & Mariano, R. (1995). Comparing Predictive Accuracy. *Journal of Business & Economic Statistics*, *13*, 253–63.

Exterkate, P., Groenen, P. J., Heij, C., & van Dijk, D. (2016). Nonlinear forecasting with many predictors using Kernel Ridge Regression. *International Journal of Forecasting*, *32*, 736–753.

Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The Elements of Statistical Learning* volume 1. Springer Series in Statistics New York.

Friedman, J. H. (2001). Greedy function approximation: a Gradient Boosting Machine. *Annals of statistics*, (pp. 1189–1232).

Friedman, J. H., & Stuetzle, W. (1981). Projection Pursuit Regression. *Journal of the American statistical Association*, *76*, 817–823.

Fu, W. J. (1998). Penalized Regressions: the Bridge versus the Lasso. *Journal of computational and graphical statistics*, *7*, 397–416.

Greene, W. H. (2003). Econometric Analysis, 5th. *Ed.. Upper Saddle River, NJ*, (pp. 89–140).

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer Feedforward Networks are Universal Approximators. *Neural networks*, *2*, 359–366.

Hothorn, T., Bühlmann, P., Kneib, T., Schmid, M., & Hofner, B. (2010). Model-based Boosting 2.0. *Journal of Machine Learning Research*, *11*, 2109–2113.

Hothorn, T., Bühlmann, P., Kneib, T., Schmid, M., & Hofner, B. (2017). mboost: Model-Based Boosting. R package version 2.8-1. *URL: http://CRAN. R-project. org/package= mboost*, .

Hyndman, R. J. (2013). *fpp: Data for "Forecasting: principles and practice"*. URL: `https://CRAN.R-project.org/package=fpp` r package version 0.5.

Igelnik, B., & Pao, Y.-H. (1995). Stochastic choice of basis functions in adaptive function approximation and the functional-link net. *IEEE Transactions on Neural Networks*, *6*, 1320–1329.

Jiang, B., Athanasopoulos, G., Hyndman, R. J., Panagiotelis, A., Vahid, F. et al. (2017). Macroeconomic forecasting for Australia using a large number of predictors. *Monash Econometrics and Business Statistics Working Papers*, *2*, 17.

Joe, S., & Kuo, F. (2008). Notes on generating Sobol sequences. `http://web.maths.unsw.edu.au/~fkuo/sobol/joe-kuo-notes.pdf`.

Jones, L. K. et al. (1992). A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and Neural Network training. *The annals of Statistics*, *20*, 608–613.

Kwok, T.-Y., & Yeung, D.-Y. (1997). Objective functions for training new hidden units in constructive Neural Networks. *IEEE Transactions on Neural Networks*, *8*, 1131–1148.

Li, M., & Wang, D. (2017). Insights into randomized algorithms for Neural Networks: Practical issues and common pitfalls. *Information Sciences*, *382*, 170–178.

Lütkepohl, H. (2005). *New introduction to Multiple Time Series analysis*. Springer Science & Business Media.

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PloS one*, *13*.

Makridakis, S., Wheelwright, S. C., & Hyndman, R. J. (2008). *Forecasting methods and applications*. John Wiley & sons.

Mallat, S., & Zhang, Z. (1993). *Matching pursuit with time-frequency dictionaries*. Technical Report Courant Institute of Mathematical Sciences New York United States.

Moudiki, T., Planchet, F., & Cousin, A. (2018). Multiple Time Series Forecasting Using Quasi-Randomized Functional Link Neural Networks. *Risks*, *6*, 22.

Nicholson, W. B., Matteson, D. S., & Bien, J. (2014). Structured regularization for large Vector Autoregression. *Cornell University*, .

Niederreiter, H. (1992). *Random number generation and quasi-Monte Carlo methods*. SIAM.

Pfaff, B. (2008). VAR, SVAR and SVEC Models: Implementation Within R package vars. *Journal of Statistical Software*, *27*. URL: `http://www.jstatsoft.org/v27/i04/`.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian Optimization of Machine Learning algorithms. In *Advances in neural information processing systems* (pp. 2951–2959).

Stock, J. H., & Watson, M. (2009). Forecasting in dynamic factor models subject to structural instability. *The Methodology and Practice of Econometrics. A Festschrift in Honour of David F. Hendry*, *173*, 205.

Vincent, P., & Bengio, Y. (2002). Kernel Matching Pursuit. *Machine Learning*, *48*, 165–187.

Wang, D., & Li, M. (2017a). Robust Stochastic Configuration Networks with kernel density estimation for uncertain data regression. *Information Sciences*, *412*, 210–222.

Wang, D., & Li, M. (2017b). Stochastic Configuration Networks: Fundamentals and algorithms. *IEEE transactions on cybernetics*, *47*, 3466–3479.