Boosted Ensemble Learning based on Randomized NNs for Time Series Forecasting*

Grzegorz Dudek[0000−0002−2285−0327]

Electrical Engineering Faculty,
Czestochowa University of Technology, Czestochowa, Poland
grzegorz.dudek@pcz.pl

Abstract. Time series forecasting is a challenging problem particularly when a time series expresses multiple seasonality, nonlinear trend and varying variance. In this work, to forecast complex time series, we propose ensemble learning which is based on randomized neural networks, and boosted in three ways. These comprise ensemble learning based on residuals, corrected targets and opposed response. The latter two methods are employed to ensure similar forecasting tasks are solved by all ensemble members, which justifies the use of exactly the same base models at all stages of ensembling. Unification of the tasks for all members simplifies ensemble learning and leads to increased forecasting accuracy. This was confirmed in an experimental study involving forecasting time series with triple seasonality, in which we compare our three variants of ensemble boosting. The strong points of the proposed ensembles based on RandNNs are very rapid training and pattern-based time series representation, which extracts relevant information from time series.

Keywords: Boosted ensemble learning · Ensemble forecasting · Multiple seasonality · Randomized NNs · Short-term load forecasting.

1 Introduction

Ensemble methods are considered to be a cornerstone of modern machine learning [1]. They are commonly used for regression and classification problems. Ensembling is also a very effective way of increasing the predictive power of forecasting models. Combining many base models improves the final forecasting accuracy as well as the stability of the response when compared to a single model approach. Success in ensemble learning depends on the proper flexibility of the ensemble members and the trade-off between their performance and diversity [2]. It is also determined by the way learners are generated at the successive stages of ensembling and the method employed to combine them.

The effectiveness of ensembling in forecasting is evidenced by the fact that in the most renowned forecasting competition, M4 [3], of the 17 most accurate models, 12 used ensembling in some form [4]. The winning submission, which

* Supported by grant 020/RID/2018/19 from the Polish Minister of Science and Higher Education titled "Regional Initiative of Excellence", 2019-22.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_26

is a hybrid model combining exponential smoothing and long short-term memory, used three types of ensembling simultaneously [5]: combining results of the stochastic training process, bagging, and combining multiple runs.

To improve the performance of ensemble learning many approaches have been proposed such as stacking [6], bagging [7], boosting [8], negative correlation learning [9], snapshot ensembles [10], and horizontal and vertical ensembles developed for deep learning [11]. Boosting, which this work focuses on, is a general ensemble technique that involves sequentially adding base models to the ensemble where subsequent models correct the performance of prior models. This approach is very effective as evidenced by the high ranking positions of boosted models such as XGBoost [12], i.e. ensemble of decision trees with regularized gradient boosting, in competitions such as those organized by Kaggle. Boosting also has many applications in the forecasting field. Some examples are: [13], where an ensemble of boosted trees is used for bankruptcy prediction; [14], where XGBoost is combined with a Gaussian mixture model for monthly streamflow forecasting; [15], where AdaBoost is applied as a component of a hybrid model for multi-step wind speed forecasting; and [16], where a natural gradient boosting algorithm is applied for solar power probabilistic forecasting. This last work highlights a valuable advantage of ensembling. It can produce probabilistic forecasts, i.e. the distribution of the forecasted variable in the future.

In this study, we propose a boosted version of our ensemble of randomized neural networks (RandNNs) for complex time series forecasting [17]. In [17], we focused on strategies for controlling the diversity of ensemble members. The members were trained independently. Here, we construct an ensemble sequentially. When a new member is added to the ensemble, it learns by taking into account the results of the ensemble members so far. We consider three methods of boosting. The contribution of this study is threefold:

1. We propose new methods of ensemble boosting: ensemble learning based on corrected targets and ensemble learning based on opposed response. They are both employed to ensure similar tasks for all ensemble members. This unification of the tasks justifies the use of identical base models in terms of architecture and hyperparameters at all stages of ensembling.
2. We develop three ensemble learning approaches for forecasting complex time series with multiple seasonality. They are based on RandNNs, pattern-based time series representation and three methods of boosting: based on residuals, corrected targets and opposed response.
3. We empirically compare the performance of the proposed ensemble methods on challenging short-term load forecasting problems with triple seasonality, and conclude that the opposed response-based approach outperforms its competitors in terms of accuracy and sensitivity to hyperparameters.

The rest of the work is organized as follows. In Section II, we present a base model, RandNN. Details of the proposed three methods of boosted ensemble learning are described in Section III. The experimental framework used to evaluate and compare the proposed ensemble methods is described in Section IV. Finally, Section V concludes the work.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_26

2 Base Forecasting Model - RandNN

As a base model, we use a single hidden layer feedforward NN with m logistic sigmoid hidden nodes [18]. In randomized learning, the weights of hidden nodes are selected randomly from a uniform distribution and symmetrical interval $U = [-u, u]$. The biases of these nodes are calculated, according to recent research [19], based on the weights as follows: $b_j = -\mathbf{a}_j^T \mathbf{x}_j^*$, where $\mathbf{a}_j$ is the vector of weights for the j-th hidden node, and $\mathbf{x}_j^*$ is one of the training patterns selected at random (see [19] and [20] for details, justification and other variants). This way of generating hidden nodes places the sigmoids into the input feature space limited to some hypercube, avoiding their saturated parts. Moreover, the sigmoids are distributed according to the data distribution. These improve the aproximation and generalization abilities of the model [19], [20].

The hidden node sigmoids are combined linearly by the output nodes: $\varphi_k(\mathbf{x}) = \sum_{j=1}^{m} \beta_{j,k} h_j(\mathbf{x})$, where $h_j(\mathbf{x})$ is the output of the j-th hidden node, and $\beta_{j,k}$ is the weight between j-th hidden and k-th output nodes. The only learnable parameters are the output weights. They are calculated from $\boldsymbol{\beta} = \mathbf{H}^+ \mathbf{Y}$, where $\mathbf{H}$ is a matrix of the hidden layer outputs, $\mathbf{H}^+$ denotes its Moore–Penrose generalized inversion, and $\mathbf{Y}$ is a matrix of target output patterns. $\mathbf{H}$ is a nonlinear feature mapping from n-dimensional input space to m-dimensional projection space (usually $m \gg n$). Note that this projection is random. Due to the fixed parameters of the hidden nodes, the optimization problem in randomized learning (selection of weights $\boldsymbol{\beta}$) becomes convex and can be easily solved by the standard least-squares method.

The forecasting model based on RandNN, which we adapt from [18], has two more components: encoder and decoder. Let us consider time series $\{E_k\}_{k=1}^{K}$ with multiple seasonality. The encoder transforms this series into input and output patterns expressing seasonal sequences of the shortest length. The input patterns, $\mathbf{x}_i = [x_{i,1}, \ldots, x_{i,n}]^T$, represent sequences $\mathbf{e}_i = [E_{i,1}, \ldots, E_{i,n}]^T$, while the output patterns, $\mathbf{y}_i = [y_{i,1}, \ldots, y_{i,n}]^T$, represent forecasted sequences $\mathbf{e}_{i+\tau} = [E_{i+\tau,1}, \ldots, E_{i+\tau,n}]^T$, where n is a period of the seasonal cycle (e.g. 24 hours for daily seasonality), $i = 1, ..., K/n$ is the sequence number, and $\tau \geq 1$ is a forecast horizon. The patterns are defined as follows:

$$\mathbf{x}_i = \frac{\mathbf{e}_i - \overline{e}_i}{\widetilde{e}_i}, \qquad \mathbf{y}_i = \frac{\mathbf{e}_{i+\tau} - \overline{e}_i}{\widetilde{e}_i} \tag{1}$$

where $\overline{e}_i$ is the mean value of sequence $\mathbf{e}_i$, and $\widetilde{e}_i = \sqrt{\sum_{t=1}^{n} (E_{i,t} - \overline{e}_i)^2}$ is a measure of sequence $\mathbf{e}_i$ dispersion.

Input patterns $\mathbf{x}_i$ represent successive seasonal sequences which are centered and normalized. They have a zero mean, and the same variance and unity length. Thus they are unified and differ only in shape. In contrast, patterns $\mathbf{y}_i$ are not globally unified and can express additional seasonality, e.g. when time series include both daily and weekly seasonalities, the former is expressed in the y-pattern shape, while the latter is expressed in y-pattern level and dispersion (compare x- and y-patterns in Fig. 2 and see discussion in [18]). In the case of such seasonalities, we build forecasting models that learn from data representing

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_26

the same days of the week, e.g. for test query pattern $\mathbf{x}$ representing Monday, training set Φ consists of x-patterns representing all historical Mondays from the data and y-pattern representing the Tuesdays following them (assuming $\tau = 1$).

The decoder based on the y-pattern forecasted by the network, $\widehat{\mathbf{y}}$, and coding variables describing the query sequence, $\widetilde{e}$ and $\overline{e}$, using transformed equation (1) for $\mathbf{y}$, calculates the forecasted seasonal sequence:

$$\widehat{\mathbf{e}} = \widehat{\mathbf{y}}\widetilde{e} + \overline{e} \tag{2}$$

Remarks:

1. RandNN was designed for forecasting time series with multiple seasonalities. In the case of one seasonality, y-patterns express only one seasonality, and there is no need to decompose the forecasting problem. In the case of no seasonality, the input pattern length should be selected experimentally, while the y-pattern length is equal to the forecast horizon.
2. The bounds of the interval for random weights, u, correspond to the maximum sigmoid slope. To increase interpretability, let us express the bounds u using the slope angles α_{max} [20]: $u = 4 \tan \alpha_{max}$, and treat α_{max} as a hyperparameter. The second hyperparameter is the number of hidden nodes, m. Both hyperparameters decide about the bias-variance tradeoff of the model and should be tuned to the complexity of the target function.
3. Advantages of RandNN are very fast training and the simplification of the forecasting task due to pattern representation. In [18], it was shown that RandNN can compete with fully-trained NN in terms of forecasting accuracy, but is much faster to train.

3 Boosting of Ensemble Learning

3.1 Ensemble Learning Based on Residuals

An ensemble based on residuals, which is a simplified variant of a gradient boosting algorithm [21], is constructed sequentially. In the first step, a base model learns on the training set $\{(x_i, y_i)\}_{i=1}^N$ (we consider a scalar input and output for simplicity) and fits to original data. Let us denote this model $f_1(x)$ and the ensemble model including one member $F_1(x) = f_1(x)$. In the second step, the second base model is added, $f_2(x)$, such that the sum of this model with the previous one, $F_2(x) = F_1(x) + f_2(x)$, is the closest possible to the target y. In the successive steps, further base models are added with the same expectation. In the k-th step, the function fitted by the ensemble is $F_k(x) = F_{k-1}(x) + f_k(x)$. Thus, the error in this step, $MSE_k = \frac{1}{N} \sum_{i=1}^N (y_i - F_k(x_i))^2$, can be written as:

$$MSE_k = \frac{1}{N} \sum_{i=1}^{N} (f_k(x_i) - r_{k-1,i})^2 \tag{3}$$

where $r_{k-1,i} = y_i - F_{k-1}(x_i)$ is a residual between the target and the response of the ensemble of $k-1$ members.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_26

Equation (3) clearly shows that the base model added to the ensemble at the k-th stage fits to the residuals between the target and the ensemble built at stage $k-1$. So, it attempts to correct the errors of its predecessors. The training set for the base model in the k-th stage is $\{(x_i, r_{k-1,i})\}_{i=1}^{N}$. The final ensemble response is the sum of all member responses: $F_K(x) = \sum_{k=1}^{K} f_k(x)$.

Algorithm 1 EnsR: Boosting based on residuals

Input: Base model f (RandNN), Training set $\Phi = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$, Ensemble size K
Output: RandNN ensemble F_K
Procedure:
for $k = 1$ **to** K **do**
  Learn f_k based on Φ
  Calculate ensemble response $F_k(\mathbf{x}_i) = \sum_{l=1}^{k} f_l(\mathbf{x}_i), i = 1, ..., N$
  Determine residuals $\mathbf{r}_{k,i} = \mathbf{y}_i - F_k(\mathbf{x}_i), i = 1, ..., N$
  Modify training set $\Phi = \{(\mathbf{x}_i, \mathbf{r}_{k,i})\}_{i=1}^{N}$
end for

Algorithm 1 and Fig. 1 demonstrate the process of building an ensemble based on residuals (EnsR) for pattern-based forecasting. Learner 1 (RandNN) learns the original target function on Φ. Each subsequent learner learns the residuals between the target patterns $\mathbf{y}$ and the aggregated outputs of its predecessors shown in the lower panel of Fig. 1. Note that in EnsR, each learner can have a different problem to solve, expressing different features. Learner 1 learns the specific patterns of seasonal cycles $\mathbf{y}$, while the next learners learn completely different tasks, i.e. the residuals, which do not have such distinct patterns as $\mathbf{y}$ and have a large stochastic component. This inconsistency between the problems solved by the learners can affect negatively the final result, especially in the common case of using identical base models (the same architecture and hyperparameters) at every stage of ensembling.

3.2 Ensemble Learning Based on Corrected Targets

To unify the problems solved by the learners at successive stages of ensembling, we modify the EnsR framework as follows. During the sequential process, at stage k, the base model $f_k(x)$ is added to the ensemble. Ensemble response at this stage is the average of k learners: $F_k(x) = \frac{1}{k}\sum_{l=1}^{k} f_l(x)$. The loss function can be expressed as:

$$\begin{aligned} MSE_k &= \frac{1}{N}\sum_{i=1}^{N}\left(y - \left(\frac{1}{k}\sum_{l=1}^{k-1} f_l(x) + \frac{f_k(x)}{k}\right)\right)^2 \\ &= \frac{1}{Nk}\sum_{i=1}^{N}\left(f_k(x) - \left(ky - \sum_{l=1}^{k-1} f_l(x)\right)\right)^2 \end{aligned} \tag{4}$$

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_26

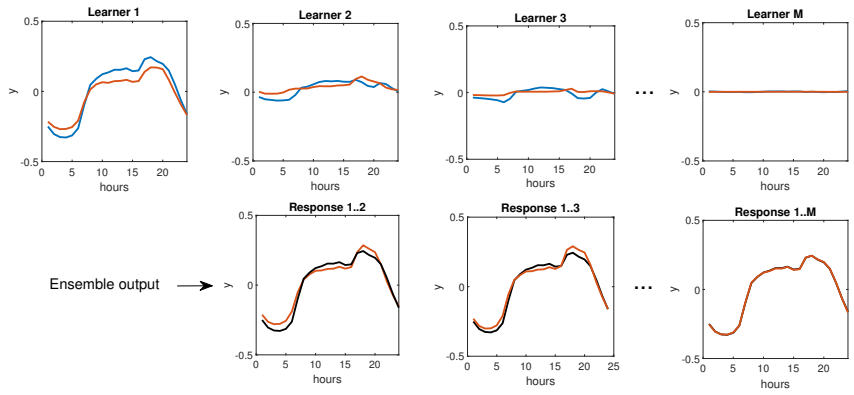

Fig. 1. EnsR: Responses produced by individual learners (upper panel) and the ensemble (lower panel). Target responses of individual learners in blue, target responses of the ensemble in black, real responses in red.

As can be seen from this equation, the difference between ky and the sum of the previous learners is the new target to which the k-th learner is fitted. This target expresses pattern y corrected by aggregated residuals of the previous learners: $y + \sum_{l=1}^{k-1}(y - f_l(x))$. If the aggregated residuals are much smaller compared to the y-pattern (we expect this), the new targets at all stages of ensembling have a similar shape to the y-pattern. So, all learners have similar tasks to solve and it is justified for them to have the same architecture and hyperparameters. This unburdens us from the awkward and time-consuming task of selecting the optimal model at each stage of ensembling.

Algorithm 2 summarizes ensemble learning based on corrected targets (EnsCT) for pattern-based forecasting. Fig. 2 shows the targets for learners at successive stages, learners' outputs and the ensemble outputs. It was observed that at the initial stages of ensembling, the targets express the y-pattern shape, but this shape degrades gradually in the later stages - see the target for K-th learner in Fig. 2. Thus, the proposed EnsCT only partially solves the problem of inconsistency between tasks learned at the successive stages of ensembling.

Algorithm 2 EnsCT: Boosting based on corrected targets

Input: Base model f (RandNN), Training set $\Phi = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$, Ensemble size K
Output: RandNN ensemble F_K
Procedure:
for $k = 1$ **to** K **do**
  Learn f_k based on Φ
  Calculate ensemble response $F_k(\mathbf{x}_i) = \frac{1}{k}\sum_{l=1}^{k} f_l(\mathbf{x}_i), i = 1, ..., N$
  Determine corrected targets $\mathbf{y}'_{k,i} = (k+1)\mathbf{y}_i - kF_k(\mathbf{x}_i), i = 1, ..., N$
  Modify training set $\Phi = \{(\mathbf{x}_i, \mathbf{y}'_{k,i})\}_{i=1}^N$
end for

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_26

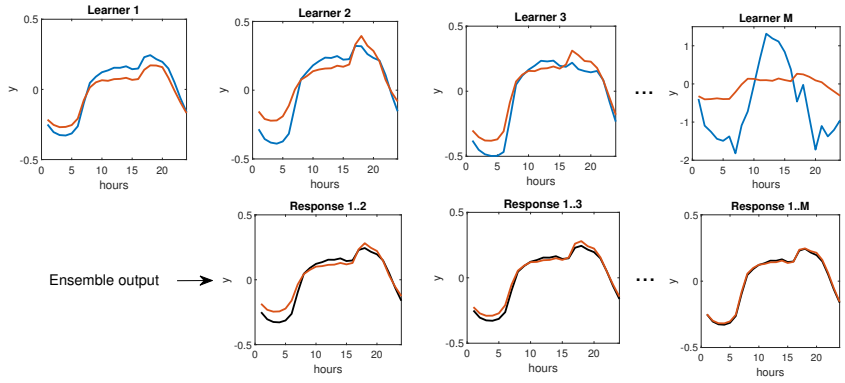

Fig. 2. EnsCT: Responses produced by individual learners (upper panel) and the ensemble (lower panel). Target responses of individual learners in blue, target responses of the ensemble in black, real responses in red.

3.3 Ensemble Learning Based on Opposed Response

To prevent the degradation of the target shapes in the subsequent steps of ensembling, we propose ensemble learning based on opposed response (EnsOR). The first step of the ensemble building procedure is the same as in EnsR and EnsCT. At this stage, the base model $f_1(x)$ learns on original training set Φ. The ensemble response is $F_1(x) = f_1(x)$. The residual is calculated, $r_1 = y - F_1(x)$, and the "opposed" response pattern is determined as follows:

$$\hat{y}'_1 = y + r_1 = 2y - F_1(x) \tag{5}$$

The opposed response pattern expresses the target pattern augmented by the opposed error produced by the ensemble (see Fig. 3). The average of the response pattern $F_1(x)$ and the opposed response pattern $\hat{y}'_1$ gives the target pattern y. In the next step, base model $f_2(x)$ learns on training set $\{(x_i, \hat{y}'_{1,i})\}_{i=1}^{N}$. Thus it learns the opposed response to reduce the ensemble residual. The ensemble response is calculated as the average of learners: $F_2(x) = \frac{f_1(x)+f_2(x)}{2}$. These operations are repeated in the following steps (see Algorithm 3). Namely, in step k, the opposed response pattern determined at stage $k-1$, $\hat{y}'_{k-1} = 2y - F_{k-1}(x)$, becomes the target pattern for learner $f_k(x)$. The ensemble response is the average of k learners and the loss function at stage k takes the form:

$$MSE_k = \frac{1}{N}\sum_{i=1}^{N}\left(f_k(x_i) - \hat{y}'_{k-1,i}\right)^2 \tag{6}$$

where $\hat{y}'_{k-1,i} = y_i + r_{k-1,i} = 2y_i - F_{k-1}(x_i)$ is the opposed response pattern at stage $k-1$.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_26

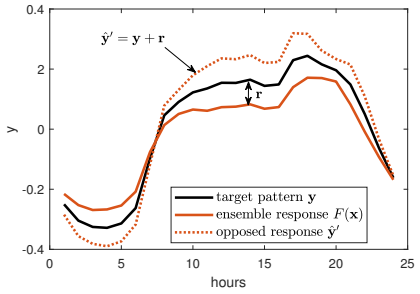

Fig. 3. Construction of the opposed response pattern.

Algorithm 3 EnsOR: Boosting based on opposed response

Input: Base model f (RandNN), Training set $\Phi = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$, Ensemble size K
Output: RandNN ensemble F_K
Procedure:
for $k = 1$ **to** K **do**
  Learn f_k based on Φ
  Calculate ensemble response $F_k(\mathbf{x}_i) = \frac{1}{k}\sum_{l=1}^{k} f_l(\mathbf{x}_i), i = 1, ..., N$
  Determine opposed response $\hat{\mathbf{y}}'_{k,i} = 2\mathbf{y}_i - F_k(\mathbf{x}_i), i = 1, ..., N$
  Modify training set $\Phi = \{(\mathbf{x}_i, \hat{\mathbf{y}}'_{k,i})\}_{i=1}^{N}$
end for

Fig. 4 shows learners' and ensemble responses in the following steps of ensembling. Dashed lines express the opposed responses which become targets for the based models in the next steps (blue lines). Note that the shape of the initial target pattern $\mathbf{y}$ is maintained until the last stage. Thus, all learners learn on similar data, and using identical base models at each stage of ensembling means no objections can be raised, unlike in the case of EnsR.

4 Experimental Study

In this section, we verify our proposed methods of ensemble boosting on four time series forecasting problems. These comprise short-term electrical load forecasting problems for four European countries: Poland (PL), Great Britain (GB), France (FR) and Germany (DE) (data was collected from www.entsoe.eu). The hourly load time series express three seasonalities: yearly, weekly and daily. The data period is four years, from 2012 to 2015. Atypical days such as public holidays were excluded from the data (between 10 and 20 days a year). The forecasting problem is to predict the load profile (24 hourly values) for each day of 2015 based on historical data. The forecast horizon is one day, $\tau = 1$. The number of ensemble members was $K = 50$. As a performance metric we use mean absolute percentage error (MAPE). All algorithms were implemented in Matlab 2021b and run on a ten-core CPU (Intel i7-6950x, 3.0 GHz, 48 GB RAM).

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_26

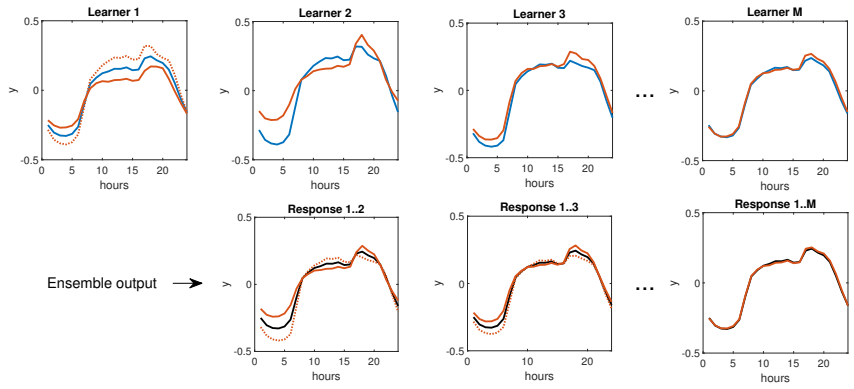

Fig. 4. EnsOR: Responses produced by individual learners (upper panel) and the ensemble (lower panel). Target responses of individual learners in blue, target responses of the ensemble in black, real responses in red, and opposed responses in dashed red.

In the first experiment, we evaluate ensemble sensitivity to the base model hyperparameters: number of hidden nodes m and size of the interval for hidden weights α_{max}. Fig. 5 shows the impact of hyperparameters on the test MAPE. Note that EnsR is the most sensitive to hyperparameters, while EnsOR is the least sensitive. To quantitatively compare the sensitivities of the ensemble variants, as a rough measure of sensitivity to a given hyperparameter, we define standard deviation of the test MAPE for the ensemble with optimal values of other hyperparameters:

$$S_m = Std(MAPE(\mathbf{y}, F(\mathbf{x}, \alpha^*_{max}, m))) \tag{7}$$

$$S_{\alpha_{max}} = Std(MAPE(\mathbf{y}, F(\mathbf{x}, \alpha_{max}, m^*))) \tag{8}$$

where the optimal hyperparameter values are marked with asterisks.

From Fig. 5, we can see that the accuracy of EnsR and EnsCT deteriorates quickly with the number of hidden nodes and interval U size, when these hyperparameters exceed their optimal values. This deterioration is related to the gradual loss of generalization for higher values of m and α_{max}. Deterioration for EnsOR is much slower, which means that this approach is more resistant to overtraining.

Table 2 shows optimal hyperparameters, test and training errors and compares the sensitivities of ensembling variants. The lowest values are in bold. Table 2 clearly shows that EnsOR performs best. This method gave the lowest errors for each dataset and had the lowest sensitivity to both hyperparameters. By contrast, EnsR performed worst in terms of error and sensitivity. Note that the optimal hyperparameter values are smaller for EnsR and EnsCT than for EnsOR. This means that the base models in these boosting variants need to be less flexible (weaker) to prevent overfitting. Nevertheless, EnsR and EnsCT do not perform as well as EnsOR.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_26

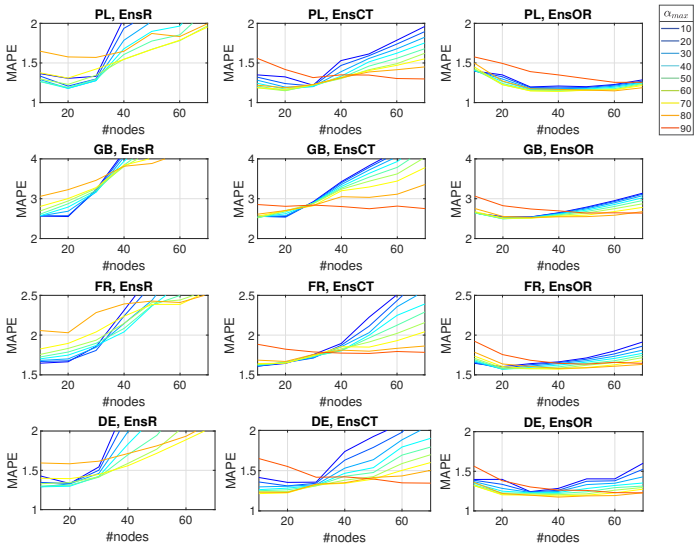

Fig. 5. Ensemble sensitivity to RandNN hyperparameters.

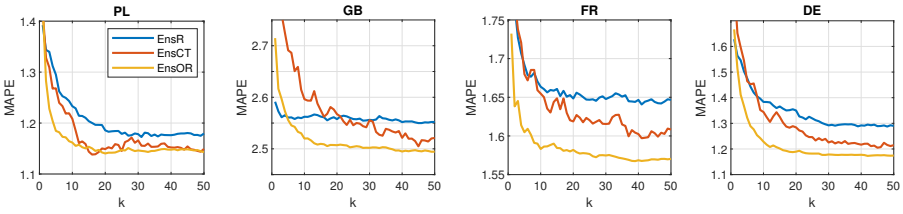

Fig. 6. Ensemble error at successive stages.

Fig. 6 demonstrates test MAPE in successive steps of ensembling. We can observe from this figure that for EnsOR the error converges faster with k than for other ensemble variants. The convergence curve is also smoother for EnsOR than for its competitors. For them, in many cases, adding a new member to the ensemble causes a temporary increase in error. Moreover, it was observed for EnsR that if the hyperparameters are too high (greater than optimal), which means a flexible base model, the error starts to successively increase with subsequent ensembling stages. This phenomenon is to a lesser degree observed for EnsCT, but not observed for EnsOR. Fig. 7 demonstrates this problem for PL data and $m = 100$, $\alpha_{max} = 70$.

To improve further ensemble learning based on opposed response, we introduce weights for training patterns which express their similarity to the query pattern. The more similar training pattern $\mathbf{x}_i$ to query pattern $\mathbf{x}$, the higher the weight. The similarity measure is a scalar product, $\mathbf{x}^T\mathbf{x}_i$. The opposed response takes the form:

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_26

Table 1. Optimal RandNN hyperparameters, errors and ensemble sensitivity.

Data	Ensemble	m^*	α^*_{max}	$MAPE_{tst}$	$MAPE_{trn}$	S_m	$S_{\alpha_{max}}$
PL	EnsR	20	40	1.18	0.76	0.641	0.436
	EnsCT	20	60	1.15	**0.72**	0.092	0.174
	EnsOR	40	70	**1.14**	**0.72**	**0.064**	**0.110**
GB	EnsR	20	10	2.55	1.55	0.509	1.531
	EnsCT	10	50	2.52	**1.51**	**0.104**	0.628
	EnsOR	20	50	**2.49**	**1.51**	**0.104**	**0.163**
FR	EnsR	10	10	1.65	1.27	0.466	0.853
	EnsCT	10	20	1.61	1.24	0.088	0.436
	EnsOR	20	40	**1.57**	**1.20**	**0.058**	**0.069**
DE	EnsR	10	40	1.29	1.37	0.774	0.572
	EnsCT	10	70	1.21	1.26	0.138	0.141
	EnsOR	40	80	**1.17**	**1.05**	**0.036**	**0.063**

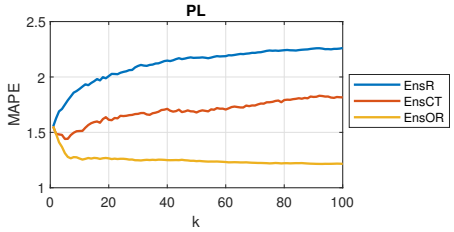

Fig. 7. Ensemble error at successive stages for too flexible base models.

$$\hat{\mathbf{y}}'_{k,i} = \mathbf{y}_i + w_i \mathbf{r}_{k,i}, i = 1, ..., N \tag{9}$$

where $\mathbf{r}_{k,i} = \mathbf{y}_i - F_k(\mathbf{x}_i)$ is a residual vector for the i-th training pattern and $w_i \in [0, 1]$ is the weight of this pattern.

In the base variant of EnsOR, the weights for all patterns are equal to one. In typical ensemble learning, the weights are zero (the base models at each stage of ensembling learn on the original training set Φ; such an approach we considered in [17]). By introducing weights, we try to balance these two approaches.

As a weighting function, we consider four variants. In the simplest one, EnsOR1, we assume that the weighting function g is just the scalar product:

$$g(\mathbf{x}, \mathbf{x}_i) = \mathbf{x}^T \mathbf{x}_i \tag{10}$$

To avoid negative values of (10) we can also use $g(\mathbf{x}, \mathbf{x}_i) = \frac{1}{2}(\mathbf{x}^T \mathbf{x}_i + 1)$ or replace negative values with zeros.

In the second variant, EnsOR2, we sort the training patterns according to similarity to the query pattern, from the most to the least similar. Let $r = 1, ..., N$

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_26

be the rank of the training patterns in the similarity ranking. The weighting function expresses the linear dependence of the weight on the rank:

$$g(\mathbf{x}, \mathbf{x}_i) = 1 + \frac{1 - r_i}{N} \tag{11}$$

In the third variant, EnsOR3, the weighting function expresses the non-linear dependence of the weight on the rank:

$$g(\mathbf{x}, \mathbf{x}_i) = \left(1 + \frac{1 - r_i}{N}\right)^d \tag{12}$$

where $d > 1$ (we assume $d = 4$).

In the fourth variant, EnsOR4, the most similar training patterns to the query pattern have unity weights, while the others have zero weights:

$$g(\mathbf{x}, \mathbf{x}_i) = \begin{cases} 1 & \text{if } \mathbf{x}_i \in \Xi_\kappa(\mathbf{x}) \\ 0 & \text{if } \mathbf{x}_i \notin \Xi_\kappa(\mathbf{x}) \end{cases} \tag{13}$$

where $\Xi_\kappa(\mathbf{x})$ denotes a set of κ nearest neighbors of query pattern $\mathbf{x}$ in Φ.

An example of weights assigned to the training patterns by the above weighting functions are shown in Fig. 8.

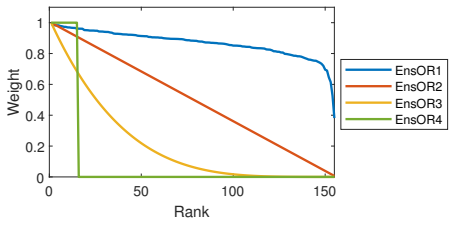

Fig. 8. Examples of weights assigned to training pattern in different weighing variants.

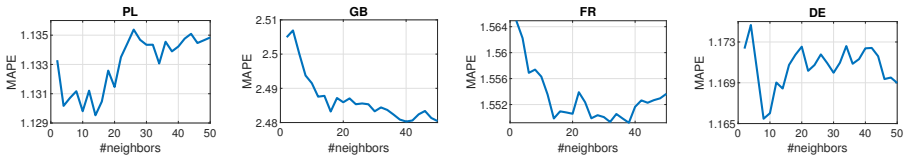

Fig. 9. Ensemble error for EnsOR4 depending on κ.

Fig. 9 allows us to evaluate the impact of the number of nearest neighbours κ in EnsOR4 on the test MAPE. The optimal value of κ depends on the data set: for PL $\kappa = 14$, for GB $\kappa = 40$, for FR $\kappa = 38$, and for DE $\kappa = 8$.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_26

Table 2. Results for EnsOR with weighted patterns.

Data	EsnOR1	EnsOR2	EnsOR3	EnsOR4	RandNN [17]	Ens1 [17]
PL	1.1419	1.1381	1.1329	**1.1295**	1.3206	1.1417
GB	2.4935	2.4804	2.4822	**2.4803**	2.6126	2.5148
FR	1.5668	1.5570	1.5524	**1.5492**	1.6711	1.5690
DE	1.1743	1.1711	1.1683	**1.1655**	1.3809	1.1811

Table 2 compares the proposed methods of pattern weighting. In this table results for individual RandNN and Ens1 are also shown. Ens1 is a classical (not boosted) ensemble of RandNN. Ens1 constructs the final forecast as an average of the responses of RandNN, which learn simultaneously on the same training set Φ [17]. As can be seen from this table, EnsOR4 outperformed its competitors as well as classical ensembling Ens1. The much higher error for individual RandNN justifies fully ensembling. It is worth mentioning that comparing Ens1 with other forecasting models, including statistical models (ARIMA, exponential smoothing, Prophet) and machine learning models (MLP, SVM, ANFIS, LSTM, GRNN, nonparametric models), reported in [17] clearly shows that Ens1 outperforms all its competitors in terms of accuracy.

5 Conclusion

Forecasting complex time series with multiple seasonalities is a challenging problem, but one we solve using ensemble of randomized NNs. We propose three methods of boosting the RandNN ensemble. We showed that in ensemble learning based on residuals, the base models have different tasks to solve at successive stages of ensembling. Thus, the base model which is optimal at a given stage may not be optimal at other stages. To avoid having to select an optimal model at each stage of ensembling, which is unreasonable and too time-consuming, we propose to unify the tasks solved at all stages. Doing so allows us to use the same base model (the same architecture and hyperparameters), RandNN in our case, which is optimal for all stages. To unify the tasks, we propose ensemble learning based on corrected targets and ensemble learning based on opposed response. The latter proved to be more resistant to task degradation at subsequent stages.

The experimental studies performed on four forecasting problems expressing triple seasonalities confirmed that the opposed response-based approach outperforms its competitors in terms of forecasting accuracy as well as sensitivity to both the base model hyperparameters and ensemble size. Further improvement of the winning solution was achieved by weighting the training patterns according to their similarity to the query pattern.

In further research, we plan to develop the opposed response-based approach for other types of learners, e.g. decision trees, and other types of problems (regression, classification).

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_26

References

1. Reeve, H.W.J., Brown, G.: Diversity and degrees of freedom in regression ensembles. NEUROCOMPUTING **298**, 55–68 (2018)
2. Brown, G., Wyatt, J.L., Tino, P.: Managing diversity in regression ensembles. J MACH LEARN RES **6**, 1621–1650 (2005)
3. Makridakis, S., Spiliotis, E., Assimakopoulos, V.: The M4 competition: Results, findings, conclusion and way forward. INT J FORECASTING **34**(4), 802–808 (2018)
4. Atiya, A.F.: Why does forecast combination work so well? INT J FORECASTING **36**(1), 197–200 (2020)
5. Smyl, S.: A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. INT J FORECASTING **36**(1), 75–85 (2020)
6. Wolpert, D.H.: Stacked generalization. NEURAL NETWORKS **5**(2), 241–259 (1992)
7. Breiman, L.: Bagging predictors. MACH LEARN **24**(2), 123–140 (1996)
8. Drucker, H.: Boosting using neural nets. In: A. Sharkey (ed.), Combining artificial neural nets: Ensemble and modular learning. Springer (1999)
9. Chen, H., Yao, X.: Regularized negative correlation learning for neural network ensembles. IEEE T NEUR NET LEAR **20**(12), 1962–1979 (2009)
10. Huang, G. et al.: Snapshot ensembles: Train 1, get M for free. arXiv:1704.00109 (2017)
11. Xie, J., Xu, B., Zhang, C.: Horizontal and vertical ensemble with deep representation for classification. arXiv:1306.2759 (2013)
12. Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system. In: Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 785–794 (2016)
13. Zieba, M., Tomczak, S.K., Tomczak, J.M.: Ensemble boosted trees with synthetic features generation in application to bankruptcy prediction. EXPERT SYST APPL **58**(1), 93–101 (2016)
14. Ni, L. et al.: Streamflow forecasting using extreme gradient boosting model coupled with Gaussian mixture model. J HYDROL **586**, 124901 (2020)
15. Li, Y. et al.: Smart wind speed forecasting approach using various boosting algorithms, big multi-step forecasting strategy. RENEW ENERG **135**, 540–553 (2019)
16. Mitrentsis, G., Lens, H.: An interpretable probabilistic model for short-term solar power forecasting using natural gradient boosting. APPL ENERG **309**, 118473 (2022)
17. Dudek, G., Pełka, P.: Ensembles of randomized neural networks for pattern-based time series forecasting. In: ICONIP 2021, LNCS, vol 13110, pp. 418–430. Springer, Cham (2021)
18. Dudek, G.: Randomized neural networks for forecasting time series with multiple seasonality. In: IWANN 2021, LNCS, vol. 12862, pp. 196–207. Springer, Cham (2021)
19. Dudek, G.: Generating random weights and biases in feedforward neural networks with random hidden nodes. INFORM SCIENCES **481**, 33–56 (2019)
20. Dudek, G.: Generating random parameters in feedforward neural networks with random hidden nodes: Drawbacks of the standard method and how to improve it. In: ICONIP 2020, CCIS, vol. 1333, pp. 598–606. Springer, Cham (2020)
21. Mason, L., Baxter, J., Bartlett, P.L., Frean, M.: Boosting algorithms as gradient descent. In: Solla S.A. et al. (eds.) Advances in Neural Information Processing Systems 12, pp. 512–18. MIT Press (1999)

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_26