Constructive Aggregation and Its Application to Forecasting with Dynamic Ensembles

Vitor Cerqueira[1,2(✉)], Fabio Pinto[1], Luis Torgo[1,2,3], Carlos Soares[1,2], and Nuno Moniz[1,2]

[1] University of Porto, Porto, Portugal
vitor.cerqueira@fe.up.pt
[2] INESC TEC, Porto, Portugal
[3] Dalhousie University, Halifax, Canada

Abstract. While the predictive advantage of ensemble methods is nowadays widely accepted, the most appropriate way of estimating the weights of each individual model remains an open research question. Meanwhile, several studies report that combining different ensemble approaches leads to improvements in performance, due to a better trade-off between the diversity and the error of the individual models in the ensemble. We contribute to this research line by proposing an aggregation framework for a set of independently created forecasting models, i.e. heterogeneous ensembles. The general idea is to, instead of directly aggregating these models, first rearrange them into different subsets, creating a new set of combined models which is then aggregated into a final decision. We present this idea as constructive aggregation, and apply it to time series forecasting problems. Results from empirical experiments show that applying constructive aggregation to state of the art dynamic aggregation methods provides a consistent advantage. Constructive aggregation is publicly available in a software package. Data related to this paper are available at: https://github.com/vcerqueira/timeseriesdata. Code related to this paper is available at: https://github.com/vcerqueira/tsensembler.

Keywords: Ensemble learning · Forecasting
Constructive induction · Regression · Dynamic expert aggregation

1 Introduction

Supervised learning consists of searching for a model that represents an accurate hypothesis about some unknown function f we want to approximate. In particular, one way ensemble learning methods approach this problem is by constructing a set of M models $H = \{h^1, \ldots, h^M\}$, and aggregating them in some way to create a final decision $\hat{h}$:

$$\hat{h} = \sum_{i=1}^{M} w^i h^i \tag{1}$$

M. Berlingerio et al. (Eds.): ECML PKDD 2018, LNAI 11051, pp. 620–636, 2019.
https://doi.org/10.1007/978-3-030-10925-7_38

where w^i denotes the weight of model h^i, $\forall\ i \in \{1, \ldots, M\}$. The predictive advantage of combining different models over using a single one is nowadays widely accepted [3,20]. However, estimating the weighting factors for each model in an ensemble remains an open research question.

Previous work by Webb et al. [29,30] shows that combining ensemble learning techniques may improve predictive performance through a better trade-off between diversity and individual error of the ensemble members. These approaches typically combine different ensemble methods during the learning process. We hypothesise that similar effects can be obtained from a portfolio of heterogeneous models [5]. This is the main motivation for this work. This approach can be advantageous because models in a portfolio are typically independent and thus easily parallelised.

In this paper we propose an aggregation framework for a set of diverse and independently created models H, following the basic principles of constructive induction [31]. Constructive induction refers to procedures that modify a set of original attributes, where some of the attributes are removed, others are added, and some of the existing ones are aggregated [31]. This leads to a new set of attributes which hopefully provides an overall better description for approximating f relative to the original set. We follow a similar approach, but in our work the attributes refer to predictive models. To the best of our knowledge, there is no closely related approach for aggregating predictive models.

The aggregation framework proposed in this paper works by rearranging a set of diverse models H into different, overlapping subsets (denoted as $\mathcal{C}_H$). The elements in each of these subsets are then aggregated, leading to a new set of models H' (or sub-ensembles [30]) for approximating f. Similarly to constructive induction approaches, the search for $\mathcal{C}_H$ is done by analysing the individual predictive performance of the original models H in observations not used in the learning process (e.g. a validation set). Essentially, the new models $h' \in H'$ correspond to aggregated subsets of consistently top performing models h. We refer to this as **constructive aggregation** (CA). Our working hypothesis is that, similarly to approaches for combining different ensemble methods [30], CA leads to a decrease in the individual error of ensemble members, without overly decreasing the diversity among them.

To illustrate our idea, the workflow of CA is presented in Fig. 1 with $H = \{h^1, h^2, h^3\}$. After analysing the performance of each model in unseen observations, the set of committees $\mathcal{C}_H = \{\{h^1\}, \{h^1, h^2\}, \{h^2, h^3\}\}$ is created; then the models h^i within each committee are aggregated into models H'. Finally, the new set of combined models H' is aggregated into a final approximation $\hat{h}'$. Both of these aggregations are done according to a linear combination (Eq. 1). The construction of the committee set $\mathcal{C}_H$ is carried out by applying the concept of out-performance contiguity (OC) which is also formalised in this paper.

We apply CA to tackle time series forecasting tasks, where the goal is to predict future numeric values of time series. In experiments on 30 time series from several domains, aggregating a number of forecasting models using CA provides a consistent advantage in terms of predictive performance. That is,

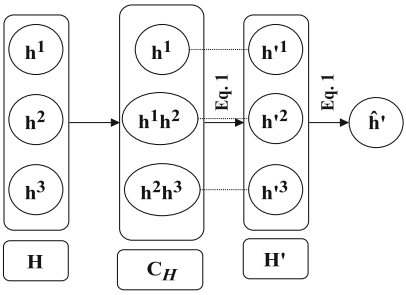


Fig. 1. Workflow of constructive aggregation: the set of available models H is rearranged into different subsets $\mathcal{C}_H$. Each subset is aggregated into H', and become hypotheses for approximating f. The models in H' are aggregated into the final decision $\hat{h}'$.

applying state of the art aggregation methods to H' leads to a better predictive performance relative to applying them to H. Moreover, we provide results that demonstrate that the constructive aggregation process entails a small execution time overhead. In summary, the contributions of this work are the following:

- Constructive Aggregation (CA), a new concept which consists in rearranging a portfolio of heterogeneous models H into different subsets $\mathcal{C}_H$, aggregating them, and using them as hypotheses H';
- Out-performance Contiguity (OC), an approach for building $\mathcal{C}_H$ in time-dependent forecasting problems;
- An extensive set of experiments including: paired comparisons that quantify the percentual difference in error between using CA and aggregating H directly (non-CA); execution time analysis of CA using OC; a sensitivity analysis of the main parameters behind the approach; and a study of CA in terms of bias-variance-covariance trade-off.

In the next section we overview the related work to this paper. In Sect. 3 we present CA, and formalise its application to forecasting problems via OC. In Sect. 4 we provide an extensive set of experiments to validate the proposal. Afterwards, in Sect. 5 we discuss the results and limitations of our work, outlining future directions. Finally, we summarise the paper in Sect. 6. The proposed method is publicly available as an R software package[1].

2 Related Work

In this section we briefly review the related work to this paper. We describe ensemble approaches for forecasting, focusing on dynamic aggregation of several experts designed for these problems. We then outline important approaches for combining different ensemble learning methods, which motivated this work.

[1] **tsensembler**: on CRAN or at https://github.com/vcerqueira/tsensembler.

2.1 Ensembles for Forecasting

An assumption of our work is that no forecasting model is universally applicable. Aiolfi and Timmermann [1] demonstrated this in empirical experiments, showing that different forecasters presented a varying relative performance over time. These findings can be regarded as a manifestation of the No Free Lunch theorem by Wolpert [33] (no learning algorithm is best suited for all tasks).

A successful approach to cope with this issue is to combine the opinion of a number of models, i.e. ensemble methods. While the simple average has been shown to be a robust combination method [28] (`Simple`), ensemble approaches for time series forecasting are typically dynamic: the weight of each model in the ensemble changes over time. This dynamic component is usually designed to cope with concept drift [13].

2.2 Dynamic Combination of Forecasting Experts

Windowing. One of the most common and successful approaches to combine predictive models in time dependent data is to weight them according to their recent performance [24,27] (`WindowLoss`). Essentially, these approaches are based on the assumption that the immediate future is likely to resemble the recent past.

Regret Minimisation. In the seminal work on online learning by Cesa-Bianchi and Lugosi [8], the authors describe several approaches for dynamically aggregating the opinion of a set of experts. These are typically designed for regret minimisation, having interesting theoretical properties. Essentially, regret is the average error suffered with respect to the best we could have obtained. In this paper we focus on the following three approaches: the exponentially weighted average [8, Sect. 2.1] (`EWA`), the polynomially weighted average [8, Sect. 2.1] (`MLpol`), and the fixed share aggregation [8, Sect. 5.2] (`FixedShare`). We will consider the application of these approaches using the gradient trick [8, Sect. 2.5]. We refer to the work by Cesa-Bianchi and Lugosi [8] for a comprehensive read on these approaches. Zinkevich [36] proposed an approach based on online convex programming and gradient descent that also minimises regret (`OGD`).

Combining by Learning. Another approach for dynamically combining forecasting models is to apply multiple regression to their output, similarly to stacked generalisation [32]. For example, Gaillard et al. [12] describe a setup in which Ridge regression is used to aggregate experts by minimising the L2-regularised least-squares (`Ridge`). Recently, Cerqueira et al. [7] proposed a metalearning approach named `ADE` (arbitrated dynamic ensemble), in which the weights of the experts are computed according to predictions of the loss that they will incur.

2.3 Combining Different Ensemble Approaches

Following the advantage shown by ensemble methods [19], further gains have been reported by approaches that combine different ensemble learning methods. Webb [29] developed `Multiboosting`, which combines `AdaBoost` [10] with `Wagging` [2]. In a posterior work [30], Webb and Zheng claim that `Multiboosting` and other similar approaches provide a better trade-off between the error of the individual members of the ensemble and the diversity among them. Yu et al. [35] presented an approach for combining a number of ensembles, dubbed `Cocktail Ensemble`, using the ambiguity decomposition. This kind of approaches have also been shown to be successful in popular machine learning competitions [25].

In this paper we follow a similar motivation for aggregating a number of forecasting models. Notwithstanding, the outlined approaches differ from our work in an important way: these build the different sub-ensembles during the learning process. Conversely, our approach settles on a portfolio of heterogeneous models [5] where the sub-ensembles are formed after the learning process, using a validation data set.

3 Methodology

This paper is focused on the problem of forecasting future values of uni-variate time series. A time series Y is a temporal sequence of values $Y = \{y_1, y_2, \ldots, y_t\}$, where $y_i \in \mathbb{R}$, $\forall\ y_i \in Y$ is the value of Y at time i.

As is standard in these problems, we construct a set of observations (the training cases) which are based on the past K lags of the time series. Each observation is composed of a feature vector $x_i \in \mathbb{X} \subset \mathbb{R}^K$, which denotes the previous K values, and a target value $y_i \in \mathbb{Y} \subset \mathbb{R}$, which represents the value we want to predict. The objective is to approximate an unknown function $f : \mathbb{X} \rightarrow \mathbb{Y}$, i.e. a function that maps the recently observed values of the series (as represented by the lagged feature vector), to the future value of the series.

3.1 Constructive Aggregation

Our approach is based on heterogeneous ensembles [5]: a set of models created in parallel and separately from each other. Diversity in the ensemble, a key ingredient in these methods [3], is introduced by varying the learning algorithm used to train each model $h \in H$.

We propose an aggregation approach for H based on constructive induction [31], and denote this idea as constructive aggregation (CA). CA works by rearranging the models $h \in H$ into different, possibly overlapping, subsets $\mathcal{C}_H$, and aggregating these into a new set of hypotheses H'. Given that forecasting models typically show a varying relative performance over time [1], our idea is that there are different subsets of models that work well in different time intervals. As such, aggregating these subsets into different combined opinions (H') may lead to better representations for approximating f.

Similarly to related approaches in the literature (c.f. Sect. 2.3) this approach may result in a better trade-off between diversity and the individual error of the ensemble members. Particularly, we hypothesise that aggregating H' decreases the individual error without overly decreasing diversity (relative to aggregating H).

CA can be split in three main steps (c.f. Fig. 1): (i) how to build $\mathcal{C}_H$ from H; (ii) how to aggregate the elements of $\mathcal{C}_H$ into a new set of hypotheses H'; and (iii) how to aggregate H' into a final decision $\hat{h}'$. In the next subsections we will address each of these steps in turn.

Algorithm 1. Out-performance contiguity for $\mathcal{C}_H$

input : set of hypotheses H; validation set $\mathcal{D}$;
smoothing window λ; contiguity window α

1 **foreach** *hypothesis* h^i *in* H **do**
2 **foreach** *observation* (x_j, y_j) *in* $\mathcal{D}$ **do**
3 $e^i_j \leftarrow |\hat{y}^i_j - y_j|$ `// absolute error of` h^i `in observation` j
4 $e'^i_j \leftarrow moving\_average(e^i_j, \lambda)$
5 **foreach** *observation* (x_j, y_j) *in* $\mathcal{D}$ **do**
6 $r_j \leftarrow rank(e'_j)$ `// rank of each hypothesis in observation` j
7 $\mathcal{C}_H \leftarrow \{\}$ `// list of committees`
8 **foreach** *size* T *from 1 to* $(|H|\text{-}1)$ **do**
9 $TOP_T \leftarrow$ top T ranked hypothesis across $\mathcal{D}$
10 **foreach** τ *in* TOP_T **do**
11 **if** τ *is top ranked for at least* α *consecutive points* **then**
12 $\mathcal{C}_H \leftarrow \mathcal{C}_H \cup \tau$
13 return $\mathcal{C}_H$

3.2 CA for Forecasting via Out-Performance Contiguity

We propose out-performance contiguity (OC) for building $\mathcal{C}_H$ from H. This approach is geared towards time series where observations are time-dependent.

Let $\mathcal{D}$ denote a set of validation observations. OC works by analysing the predictive performance of each model $h \in H$ in $\mathcal{D}$. More precisely, this procedure can be summarised as follows: a subset with size T of available models H is aggregated and used as an hypothesis for approximating f, if its elements are the top T performers (relative to H) for α contiguous observations. From the example in Fig. 1, the subset $\{h^1, h^2\}$ is aggregated into an hypothesis $h' \in H'$ because h^1 and h^2 outperform h^3 during a contiguous time interval of size α.

This idea is formalised in Algorithm 1. Initially (lines 1–4), for each model we compute its absolute error in observations of validation data ($\mathcal{D}$). To control for outliers, the loss is smoothed using a moving average of window size λ. Afterwards (lines 5–6), we compute the rank of each model $h \in H$ across $\mathcal{D}$, using the smoothed error. Then (line 8), for each possible size T of the subsets

of H (except the size of the full set H), i.e. from 1 to $|H|$-1, we do as follows. All top ranked T models across $\mathcal{D}$ are retrieved (line 9). If the models composing this top is unchanged for α observations, the respective subset becomes an element of $\mathcal{C}_H$ (lines 10–12). In the extreme case in which $\mathcal{C}_H$ is empty, we revert to using the original set of hypotheses H.

Effectively, OC searches for groups of models that perform well in consecutive unseen observations to form $\mathcal{C}_H$. The goal of this search is to make a better exploration of the regions of the input space where these groups consistently perform better relative to other models in the available pool. Moreover, these groups may be relevant in the future due to the potential variance in relative performance shown by forecasters [1], or by other recurring concepts that typically characterise time series [13].

3.3 Aggregation Steps

$\mathcal{C}_H$ into H'. The preceding subsection presents an approach for retrieving the set of subsets $\mathcal{C}_H$ from H. Most of the elements in $\mathcal{C}_H$ are potentially comprised by several models $h \in H$. As such, they need to be aggregated, in order to form a single combined opinion $h' \in H'$.

In this work we accomplish this by using a windowing approach. Essentially, the elements $h \in H$ in each subset from $\mathcal{C}_H$ are aggregated according to their recent performance [24,27]. As mentioned in Sect. 2.2, the idea is that recent observations are more similar to the one we intend to predict, and thus they are considered more relevant. More formally, the weight of each model h^i in a committee $c \in \mathcal{C}_H$ is given by its relative loss in the last λ observations:

$$w^i = \frac{scale(-\overline{L}^i_\lambda)}{\sum_{i \in c} scale(-\overline{L}^i_\lambda)} \tag{2}$$

where $\overline{L}^i_\lambda$ denotes the average loss of model i in the last known λ observations, and *scale* represents the min–max scaling function.

H' into $\hat{h}'$. The objective of the CA process, from H to H', is to create a new representation that presents a better approximation to f. Therefore, our working idea is that applying state of the art aggregation methods to this new set of hypothesis H' is better than applying them directly to H.

4 Experimental Evaluation

We carried out several experiments to validate CA via OC for forecasting with dynamic ensembles. These address the following research questions:

Q1: How do state of the art approaches for dynamic combination of forecasters perform when applied using CA relative to non-CA?

Q2: How do state of the art aggregation methods applied with CA perform relative to state of the art methods for forecasting?

Q3: What are the implications of CA in terms of bias, variance, and covariance?
Q4: How sensitive is CA via OC to different values of α and λ?
Q5: How does CA scale in terms of execution time relative to non-CA?
Q6: Is CA simply pruning or avoiding poor models?

To address these questions we used 30 real-world time series from five different domains, briefly described in Table 1. The size of each time series was truncated to 3000 instances in order to speed up execution time.

Table 1. Datasets and respective summary

ID	Time series	Data source	Data characteristics	Size	K	$\lvert\mathcal{C}_H\rvert$
1	Rotunda AEP	Porto Water Consumption from different locations in the city of Porto [7]	Half-hourly values from Nov. 11, 2015 to Jan. 11, 2016	3000	16	41
2	Preciosa Mar			3000	8	32
3	Amial			3000	9	32
4	Global Horizontal Radiation	Solar Radiation Monitoring [7]	Hourly values from Apr. 25, 2016 to Aug. 25, 2016	3000	13	44
5	Direct Normal Radiation			3000	12	55
6	Diffuse Horizontal Radiation			3000	12	33
7	Average Wind Speed			3000	6	39
8	CO.GT	Air quality indicators in an Italian city [21]	Hourly values from Mar. 10, 2004 to Apr. 04 2005	3000	15	41
9	PT08.S1.CO			3000	6	39
10	C6H6.GT			3000	6	34
11	NMHC.GT			3000	10	33
12	PT08.S2.NMHC			3000	6	34
13	PT08.S4.NO2			3000	7	37
14	PT08.S5.O3			3000	6	32
15	NOx.GT			3000	10	35
16	NO2.GT			3000	15	36
17	PT08.S3.NOx			3000	6	32
18	Temperature			3000	8	25
19	RH			3000	6	34
20	Humidity			3000	7	41
21	Electricity Total Load	Hospital Energy Loads [7]	Hourly values from Jan. 1, 2016 to Mar. 25, 2016	3000	14	44
22	Equipment Load			3000	10	35
23	Gas Energy			3000	7	29
24	Gas Heat Energy			3000	8	26
25	Water heater Energy			3000	10	18
26	Foreign exchange rates	Data market [14]	Daily, from Dec. 31, 1979 to Dec. 31, 1998	3000	12	25
27	Rainfall in Melbourne		Daily, from from 1981 to 1990	3000	27	33
28	Mean flow Saugeen river		Daily, from from 1981 to 1990	3000	7	50
29	No. of Births in Quebec		Daily, from Jan. 1, 1977 to Dec. 31, 1990	3000	6	21
30	Mean Wave Height		Hourly, from Nov. 9, 2004 to Sep. 9, 2005	3000	17	39

4.1 Experimental Setup

The methods are evaluated using the root mean squared error (RMSE) on a repeated holdout procedure with 20 testing periods. For each repetition, a random point in time is chosen from the full time window available for each series, and the previous window consisting of 60% of the data set size is used for training the ensemble while the following window of size 10% is used for testing. This approach was evidenced to provide robust estimates relative to other estimation procedures [6]. The validation set $\mathcal{D}$ described in Algorithm 1 consists of the final 30% observations of the training set. Specifically, each model $h \in H$ is initially trained in 70% of the training set, and $\mathcal{C}_H$ is built using the following 30% observations. Then, all models are retrained using the complete training set.

We estimate the size K of the feature vectors $x \in \mathbb{X}$ using the false nearest neighbours approach [17]. The estimated value for each series is shown in Table 1. The parameters of OC, α and λ, are set to 30 and 100, respectively. This means that the elements of each subset in $\mathcal{C}_H$ show a better rank than the elements of other subsets of the same size for a contiguous window of 30 points. Each point denotes the average loss of each model in the last 100 observations. The number (averaged across the 20 testing periods) of subsets comprising $\mathcal{C}_H$ is also described in Table 1 ($|\mathcal{C}_H|$).

4.2 Set of Hypotheses H and Aggregation Approaches

The set of models H forming the ensemble are built using the following **15** learning algorithms: support vector regression with Gaussian, Laplace, and linear kernels (`SVR_g`, `SVR_lp`, `SVR_l`, respectively) [16]; Gaussian processes with Gaussian, Laplace, and linear kernels (`GP_g`, `GP_lp`, `GP_l`, respectively) [16]; `LASSO` regression [11], `Ridge` regression [11]; random forest (`RF`) [34]; multivariate adaptive regression splines (`MARS`) [23]; principal components regression (`PCR`) [22]; partial least squares (`PLS`) [22]; rule-based regression (`RBR`) [18]; and projection pursuit regression with SuperSmoother and splines smoothing methods (`PPR_SS`, and `PPR_Spl`, respectively) [26]. As a preliminary analysis, in Fig. 2 we show the distribution of the rank of each model across the 30 problems. A rank of 1 means that the respective model was the best performing one in a given dataset.

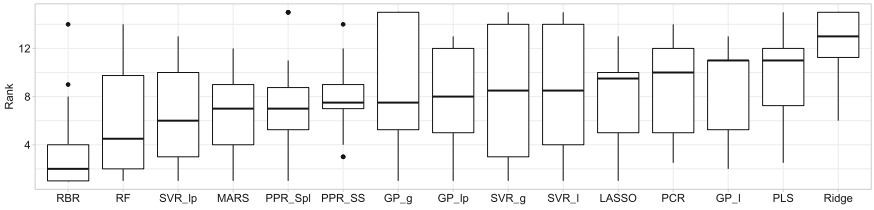


Fig. 2. Boxplot of the rank for each hypothesis in H across the 30 problems

These models are aggregated using the following approaches (c.f. Sect. 2.2 for an overview):

- `WindowLoss`: Weights computed according to the performance of the models in the last λ observations;
- `Simple`: Models are averaged using the arithmetic mean;
- `FixedShare`: The fixed share approach;
- `Ridge`: An approach that uses Ridge regression to aggregate models;
- `ADE`: A method for the arbitrage of forecasting models;
- `OGD`: An approach based on online gradient descent;
- `MLpol`: The polinomially weighted average forecast combination;
- `EWA`: Aggregation approach based on an exponentially weighted average.

When the methods are employed using CA, their name is denoted using the prefix "CA" (e.g. `CA.Simple`). We include the following baselines: `LossTrain`, in which the original hypotheses H are aggregated according to their RMSE in the training data; `CA.SimpleWorst`, a variant of `CA.Simple` in which $\mathcal{C}_H$ is built using the consistently worst performers, as opposed to using the best performers – this is accomplished by searching for the *bottom ranked hypotheses* (see line 9 in Algorithm 1); and `CA.SimpleRandom`, another variant of `CA.Simple` in which $\mathcal{C}_H$ is built randomly – the number of subsets and respective sizes are set according to OC, but these are filled with random models from the available pool. As a forecasting baseline, we include `ARIMA` [15], an online state of the art approach for time series forecasting.

4.3 Results

On Predictive Performance. Our first experiment is used to study the impact of CA on state of the art forecast combination approaches. In other words, we want to understand if, given an aggregation method, it is better to apply it to the set of hypotheses H' built using CA via OC, or directly to the set of original hypotheses H (non-CA).

Figure 3 addresses this question (**Q1**). The boxplots represent the log percentual difference in RMSE between CA and non-CA, for the different aggregation methods and across the 30 time series. Negative values denote better performance when the methods are applied using CA. These results show that, for almost all aggregation methods CA leads to a better predictive performance in the majority of problems. The exception is `MLpol`, which is relatively robust to CA.

These results are corroborated by the critical difference diagram in Fig. 4 [9]. In this, almost all state of the art aggregation methods show a better average rank when applied using CA via OC (again, the exception is `MLpol`). `CA.ADE` shows the best average rank of all methods. Almost all methods significantly outperform `ARIMA`, which shows that these are able predict future values of time series better than a state of the art approach (**Q2**).

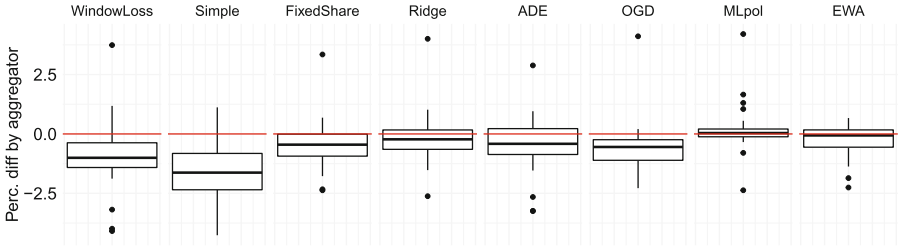


Fig. 3. Log percentual difference in RMSE between CA and non-CA for each aggregation method. Negative values denote a decrease in RMSE (better performance) when using CA.

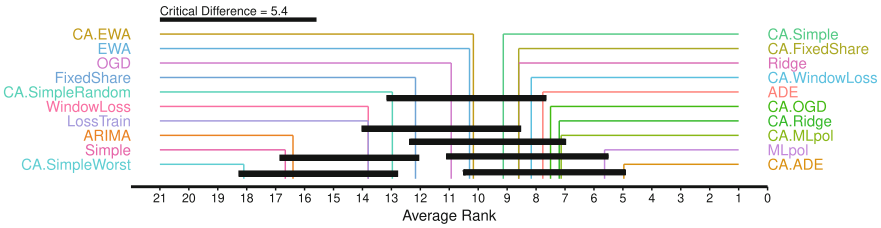


Fig. 4. Critical difference diagram for the post-hoc Nemenyi test [9]

Error Decomposition. From a regression perspective, and according to Brown et al. [4] we decomposed the squared error into bias, variance, and covariance terms as follows: $\overline{bias}^2 + \overline{var}\frac{1}{M} + (1 + \frac{1}{M})\overline{covar} + \sigma^2$, where $\overline{bias}^2$, $\overline{var}$, and $\overline{covar}$ represent the average bias, variance and covariance of the ensemble members, respectively. σ^2 is a constant irreducible term representing the variance of the noise. While a single estimator can be analysed using a bias-variance trade-off, the quadratic error generalisation of a regression ensemble depends on the bias, variance, and covariance of the individual models. The covariance term quantifies the diversity of the ensemble. Increasing values of average covariance denote less diversity. We refer to the work by Brown et al. [4] for a comprehensive read.

To analyse the impact of CA in this decomposition, we measured the percentual difference in each component relative to non-CA, across the 30 problems. Although the decomposition is valid for non-uniformly weighted ensembles [4], in the interest of brevity we focus on the simple average aggregation, i.e. difference between `CA.Simple` and `Simple`. This study is reported in Fig. 5. Negative values represent a percentual decrease in the respective term when applying CA. In the left part of the figure we present the decomposition following the proposed approach. For comparison, in the right side we present the same decomposition using the baseline `CA.SimpleWorst`.

According to the figure, `CA.Simple` shows an average decrease in the bias term relative to `Simple`. This outcome is reasonable since CA focus on searching consistently top performing subsets of models, i.e. regions where some multiple

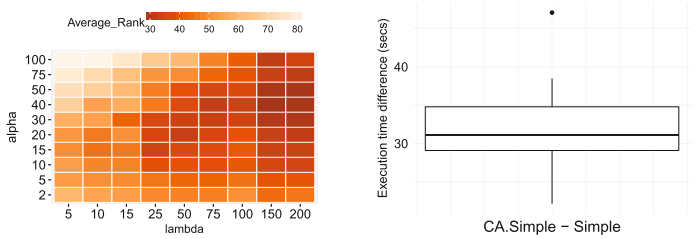
Fig. 5. Log percentual difference in $\overline{bias}^2$, $\overline{var}$, $\overline{covar}$, and RMSE between `CA.Simple` and `Simple` (left), and between `CA.SimpleWorst` and `Simple` (right)

individual models consistently show better rank than other equal-sized groups of models. There is also a considerable decrease in the average variance term. This is expected since most of the models in H' are combinations of models from H, which, when averaged, decrease variance. Oppositely, this leads to an average increase in the covariance term. This is also expected because each model from H can be part of multiple subsets that form the set of hypotheses H', leading to an increase of the ensembles' redundancy.

`CA.SimpleWorst` also leads to an average decrease in variance. Although less noticeable relative to `CA.Simple`, it also leads to a decrease in diversity (increase in covariance). The interesting part of this comparison is that for `CA.SimpleWorst` the bias term increases considerably, leading to a worse performance relative to `Simple`. This outcome suggests, as we hypothesised, that `CA.Simple` improves the performance through an improvement in the average bias of the members of the ensemble, even though it sacrifices some diversity to this effect (**Q3**).

Parameter Sensitivity and Execution Time. To address question **Q4** we analysed the sensitivity of parameters α and λ (c.f. Algorithm 1). This analysis is presented in Fig. 6a. This graphic shows an heatmap with the average rank of each combination of (α, λ) across the 30 problems. For simplicity we focused on the `Simple` aggregation method, and the set of values of each parameter were chosen arbitrarily. Overall, when α and λ are not set with too low values (from the searched grid) the average rank is relatively stable. In practice, the most appropriate set of values strongly depend on the data and the portfolio of models H.

Regarding question **Q5**, we studied the execution time of CA. Again, in the interest of conciseness we focused on the `Simple` aggregation method. To carry out this analysis we compute the time spent to train and aggregate the ensemble. Then, we measure the time difference between CA and non-CA for each time series. The results are reported in Fig. 6b, demonstrating that, on average, the



(a) (b)

Fig. 6. (**a**) Heatmap illustrating the average rank CA for varying α and λ parameters with the `Simple` method. (**b**) distribution of difference in execution time (seconds) when using `Simple` aggregation with and without CA.

`Simple` method using CA takes around 30 s more than when not using CA. This overhead is caused by the creation of $\mathcal{C}_H$ via OC. Notwithstanding, we note that the difference in execution time also depends on the aggregation approach, i.e. how it scales with the number of predictors.

On Pruning. As we already mentioned, CA via OC builds the set of committees $\mathcal{C}_H$ focusing on consistently top performers. In this context, it might be argued that the improvements in performance are due to avoiding poor predictors, and it could be reached by simply pruning them from the aggregation rule.

To test this hypothesis we compared `CA.Simple` with an approach that quantifies the weight of each model according to the average rank in the last λ observations (denoted as `AvgRank`). We focus on the rank because it is the metric we use to build $\mathcal{C}_H$. Moreover, we apply `AvgRank` with a decreasing number of models, where the predictors are dynamically suspended (assigned weight 0) according to the `AvgRank`. For example, when using 10 out of the available 15 models (`AvgRank.10`) and for a given time step we do as follows. We compute the average rank of each of the 15 models in the last λ observations. Then, we drop the worst 5 models, weighing the remaining ones (w.r.t. `AvgRank`).

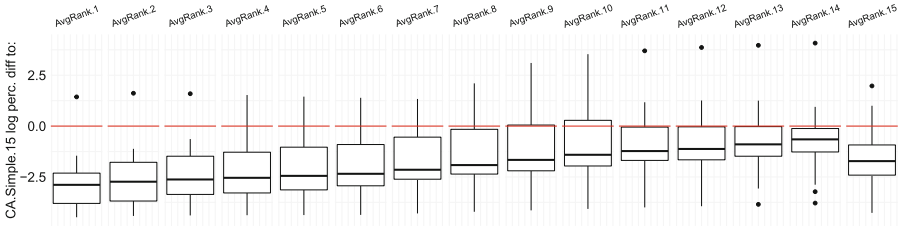


Fig. 7. Log percentual difference in RMSE of `AvgRank` with a decreasing number of predictors (denoted as sufix) relative to `CA.Simple`. Negative values denote lower RMSE by `CA.Simple`.

The results of this analysis are presented in Fig. 7. The suffix in the name of each approach denotes the number of members in each ensemble. The boxplots represent the log percentual difference in RMSE of each variant of `AvgRank` relative to `CA.Simple`, across the 30 time series. Results show that `CA.Simple` presents a better performance, which is increasing as `AvgRank` is applied with a decreasing number of models. This outcome suggests that CA is not simply pruning poor predictors (**Q6**) from the aggregation rule.

5 Discussion

5.1 On the Trade-Off Between Individual Error and Diversity

This paper follows evidence from previous work by Webb et al. [29,30], which showed that combining different ensemble approaches leads to a better trade-off between individual error of the ensemble members and diversity.

The original motivation of Webb with `Multiboosting` [29] was to increase diversity while maintaining a reasonable individual error. Notwithstanding, Webb and Zheng [30] later report different approaches where the inverse happens: cases where both diversity and individual error decrease, also leading to a lower ensemble error. The results from our experiments follow the second case. However interesting, these incite further investigation. Particularly, research into the mechanisms behind the success of this improved trade-off.

5.2 Other Limitations and Future Work

We presented OC for retrieving the set of committees $\mathcal{C}_H$ with a small execution time overhead. Despite the results showing a systematic improvement in predictive performance, our intuition is that this approach can be further improved. For example, as presented (c.f. Algorithm 1), OC searches for subsets of H of all sizes (except the full size of H), which can lead to an unnecessary redundancy. This can be particularly important if the portfolio of models H is larger than in our experimental design. We can potentially overcome this problem by introducing a *depth* parameter, which controls how large the subsets should be.

Contrary to other approaches, CA combines different sub-ensembles after the learning process, starting from a portfolio of heterogeneous models. This can be advantageous in terms of flexibility: sub-ensembles can be updated, new models can be added to the portfolio H, or obsolete ones removed.

This paper is focused on forecasting problems. In particular, the proposed algorithm OC capitalises on the time dependency among observations. Notwithstanding, our intuition is that the basic idea behind CA can be generalised to i.i.d. domains, e.g. standard regression tasks. We will also investigate this issue in future work.

6 Summary

Constructive aggregation (CA) rearranges a set of independently created hypotheses H into different subsets $\mathcal{C}_H$. This is achieved using out-performance contiguity (OC), which searches for groups of models that outperform other groups of same size contiguously during some time interval. These subsets are then aggregated into different combined hypotheses H'.

Applying state of the art aggregation approaches for forecasting to H' is demonstrated to provide better performance relative to applying them to H, on average. Moreover, this is accomplished with mild execution time overhead.

The results also suggest that the improvement in performance is mainly due to a decrease in individual error of the members of the ensemble. Although diversity also decreases, CA leads to a better trade-off between these two factors. The proposed method is publicly available in a software package.

Acknowledgements. This work is financed by Project "Coral - Sustainable Ocean Exploitation: Tools and Sensors/NORTE-01-0145-FEDER-000036", which is financed by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF).

References

1. Aiolfi, M., Timmermann, A.: Persistence in forecasting performance and conditional combination strategies. J. Econ. **135**(1), 31–53 (2006)
2. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: bagging, boosting, and variants. Mach. Learn. **36**(1–2), 105–139 (1999)
3. Brown, G.: Ensemble learning. In: Sammut, C., Webb, G.I. (eds.) Encyclopedia of Machine Learning, pp. 312–320. Springer, Boston (2010). https://doi.org/10.1007/978-0-387-30164-8_252
4. Brown, G., Wyatt, J.L., Tiňo, P.: Managing diversity in regression ensembles. J. Mach. Learn. Res. **6**, 1621–1650 (2005)
5. Caruana, R., Niculescu-Mizil, A., Crew, G., Ksikes, A.: Ensemble selection from libraries of models. In: Proceedings of the Twenty-First International Conference on Machine Learning, p. 18. ACM (2004)
6. Cerqueira, V., Torgo, L., Smailović, J., Mozetič, I.: A comparative study of performance estimation methods for time series forecasting, pp. 529–538. IEEE (2017)
7. Cerqueira, V., Torgo, L., Pinto, F., Soares, C.: Arbitrated ensemble for time series forecasting. In: Ceci, M., Hollmén, J., Todorovski, L., Vens, C., Džeroski, S. (eds.) ECML PKDD 2017. LNCS (LNAI), vol. 10535, pp. 478–494. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71246-8_29
8. Cesa-Bianchi, N., Lugosi, G.: Prediction, Learning, and Games. Cambridge University Press, New York (2006)
9. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)
10. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. **55**(1), 119–139 (1997)

11. Friedman, J., Hastie, T., Tibshirani, R.: Regularization paths for generalized linear models via coordinate descent. J. Stat. Softw. **33**(1), 1–22 (2010)
12. Gaillard, P., Goude, Y.: Forecasting electricity consumption by aggregating experts; how to design a good set of experts. In: Antoniadis, A., Poggi, J.-M., Brossat, X. (eds.) Modeling and Stochastic Learning for Forecasting in High Dimensions. LNS, vol. 217, pp. 95–115. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18732-7_6
13. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Comput. Surv. (CSUR) **46**(4), 44 (2014)
14. Hyndman, R.: Time series data library. http://data.is/TSDLdemo. Accessed 11 Dec 2017
15. Hyndman, R.J., et al.: forecast: Forecasting functions for time series and linear models, R package version 5.6 (2014)
16. Karatzoglou, A., Smola, A., Hornik, K., Zeileis, A.: kernlab - an S4 package for kernel methods in R. J. Stat. Softw. **11**(9), 1–20 (2004)
17. Kennel, M.B., Brown, R., Abarbanel, H.D.: Determining embedding dimension for phase-space reconstruction using a geometrical construction. Phys. Rev. A **45**(6), 3403 (1992)
18. Kuhn, M., Weston, S., Keefer, C., Coulter N.: C code for Cubist by Ross Quinlan. In: Cubist: Rule-and Instance-Based Regression Modeling, R package version 0.0.18 (2014)
19. Kuncheva, L.I.: A theoretical study on six classifier fusion strategies. IEEE Trans. Pattern Anal. Mach. Intell. **24**(2), 281–286 (2002)
20. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley, New York (2004)
21. Lichman, M.: UCI machine learning repository (2013). https://archive.ics.uci.edu/ml
22. Mevik, B.H., Wehrens, R., Liland, K.H.: pls: Partial Least Squares and Principal Component Regression, r package version 2.6-0 (2016)
23. Milborrow, S.: earth: Multivariate Adaptive Regression Spline Models. Derived from mda:mars by Trevor Hastie and Rob Tibshirani (2012)
24. Newbold, P., Granger, C.W.: Experience with forecasting univariate time series and the combination of forecasts. J. R. Stat. Society. Ser. (Gen.) **137**, 131–165 (1974)
25. Pfahringer, B.: Winning the KDD99 classification cup: bagged boosting. ACM SIGKDD Explor. Newsl. **1**(2), 65–66 (2000)
26. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Austria, Vienna (2013)
27. van Rijn, J.N., Holmes, G., Pfahringer, B., Vanschoren, J.: The online performance estimation framework: heterogeneous ensemble learning for data streams. Mach. Learn. **107**, 1–28 (2018)
28. Timmermann, A.: Forecast combinations. In: Handbook of Economic Forecasting, vol. 1, pp. 135–196 (2006)
29. Webb, G.I.: Multiboosting: a technique for combining boosting and wagging. Mach. Learn. **40**(2), 159–196 (2000)
30. Webb, G.I., Zheng, Z.: Multistrategy ensemble learning: reducing error by combining ensemble learning techniques. IEEE Trans. Knowl. Data Eng. **16**(8), 980–991 (2004)
31. Wnek, J., Michalski, R.S.: Hypothesis-driven constructive induction in AQ17-HCI: a method and experiments. Mach. Learn. **14**(2), 139–168 (1994)

32. Wolpert, D.H.: Stacked generalization. Neural Netw. **5**(2), 241–259 (1992)
33. Wolpert, D.H.: The lack of a priori distinctions between learning algorithms. Neural Comput. **8**(7), 1341–1390 (1996)
34. Wright, M.N.: ranger: A Fast Implementation of Random Forests, R package (2015)
35. Yu, Y., Zhou, Z.H., Ting, K.M.: Cocktail ensemble for regression. In: 7th IEEE International Conference on Data Mining, ICDM 2007, pp. 721–726. IEEE (2007)
36. Zinkevich, M.: Online convex programming and generalized infinitesimal gradient ascent. In: Proceedings of the 20th International Conference on Machine Learning, ICML 2003, pp. 928–936 (2003)