# Chapter 13
# Ensemble Methods for Time Series Forecasting

**Héctor Allende and Carlos Valle**

## 13.1 Introduction

Time series forecasting have received a great deal attention in many practical data mining of engineering and science. Traditionally, the autoregressive integrated moving average (ARIMA) model has been one of the most widely used linear models in time series prediction. However, the ARIMA model cannot easily capture the nonlinear patterns. Artificial neural networks (ANN) and other novel neural network techniques have been successfully applied in solving nonlinear pattern estimation problems.

Extensive works in literature suggest that substantial enhancement in accuracies can be achieved by combining forecasts from different models. However, forecast combination is a difficult as well as a challenging task due to various reasons, and often simple linear methods are used for this purpose. In this work, we propose a nonlinear weighted ensemble mechanism for combining forecasts from time series models.

Ensemble techniques for time series forecasting have indispensable importance in many practical data mining applications. It is an ongoing dynamic area of research, and over the years various forecasting models have been developed in the literature [1, 2]. These methods consider the individual as well as the correlations in pairs of forecasts for creating the ensembles without reducing its efficiency, simplicity, robustness and flexibility. However, this is not at all an easy task and so far no single model alone can provide best forecasting results for all kinds of time series data [2].

H. Allende (✉) · C. Valle
Departamento de Informática,
Universidad Técnica Federico Santa María, Valparaíso, Chile
e-mail: hallende@inf.utfsm.cl

C. Valle
e-mail: cvalle@inf.utfsm.cl

R. Seising and H. Allende-Cid (eds.), *Claudio Moraga: A Passion for Multi-Valued Logic and Soft Computing*, Studies in Fuzziness and Soft Computing 349, DOI 10.1007/978-3-319-48317-7_13

Combining forecasts from conceptually different methods is a very effective way to improve the overall precisions in forecast. The earliest use in the practice started in 1969 with the important work of Bates on Granger [3]. Since then, numerous forecast combination methods have been developed in the literature [4]. The vital role of model combination in time series forecasting can be credited in the following facts: (i) by adequate ensemble techniques, the forecasting strengths of the participating models aggregate and their weaknesses diminish, thus enhancing the overall forecasting accuracy to a great extent, (ii) it is frequently a large uncertainty about the optimal forecasting model, and in such situations combination strategies are the most appropriate alternatives to use, and (iii) combining multiple forecasting can efficiently reduce errors arising from faulty assumptions bias, or mistakes in the data.

The ordinary average is the most simply used forecast ensemble technique. It is easy to understand implement and interpret. However, this method is frequently criticized because it does not utilize the relative performances of the contribution models and is quite sensitive to the outliers. For this reason, other forms of averaging, such as median, winsorized mean and trimmed mean, have been addressed in the literature [5]. Another method is the weighted linear combinations of individual forecasts in which the weights are determined from the past errors of the contributing models [6].

However, this method completely ignores the possible relationships between two or more participating models and hence is not adequate for combining nonstationary models, which have been addressed by researchers [7].

The focus of this chapter will be ensembles for combining time series forecasting. Design, implementation and application will be the main topics of this chapter. Specifically, what will be considered are the conditions under which ensemble based systems may be more beneficial than their single machine; algorithms for generating individual components of ensemble systems and various procedures through which they can be combined. Various ensemble-based algorithms will be analyzed: Bagging, Adaboost and negative correlation; as well as combination rules and decision templates.

Finally future research directions will be included like incremental learning, machine fusion and the others areas in which ensemble of machines have shown promise.

## 13.2 Time Series Analysis

### *13.2.1 Linear Models*

The statistical approach for forecasting involves the construction of stochastic models to predict the value of an observation $x_t$ using previous observations. This is often accomplished using linear stochastic difference equation models, with random input. By far, the most important class of such models is the linear autoregressive integrate moving average (ARIMA) model. Here we provide a very brief review of the linear ARIMA-models and optimal prediction for these models. A more comprehensive treatment may be found for example in [1]. The seasonal ARIMA $(p, d, q) \times (P, D, Q)^S$ model for such time series is represented by

$$\Phi_P(B^S)\phi_p(B)\nabla_S^D \nabla^d x_t = \Theta_Q(B^S)\theta_q(B)\varepsilon_t \tag{13.1}$$

where $\phi_P(B^S)$ is the nonseasonal autoregressive operator of order p, $\theta_q(B)$ is the nonseasonal moving average operator of order q, $\Phi_P(B^S)$, $\Theta_Q(B^S)$ are the seasonal autoregressive and moving average operator of order P and Q and the terms $x_t$ and $\varepsilon_t$ are the time series and a white noise respectively. Moreover it is assumed that $E[\varepsilon_t | x_{t-1}, x_{t-2}, \ldots] = 0$. This condition is satisfied for example when $\varepsilon_t$ are zero mean, independent and identically distributed and independent of past $x_t s$. It is assumed throughout that $\varepsilon_t$ has finite variance $\sigma^2$. The backshift operator B shifts the index of a time series observation backwards, e.g. $Bx_t = x_{t-1}$ and $B^k x_t = x_{t-k}$. The order of the operator is selected by Akaike's information criterion (AIC) or by Bayes information criterion (BIC) [8] and the parameters $\Phi_1, \ldots, \Phi_P, \phi_1, \ldots, \phi_p$, $\Theta_1, \ldots, \Theta_Q$ and $\theta_1, \ldots, \theta_q$ are selected from the time series data using optimization methods such as maximum likelihood [1] or using robust methods such as recursive generalized maximum likelihood [9]. The ARMA-models is limited by the requirement of stationarity and invertibility of the time series. In other words, the system generating the time series must be invariant and stable. In addition, the residuals must be independent and identically distributed [10].

The ARMA models require a stationary time series in order to be useful for forecasting. The condition for a series to be weak stationary is that for all t

$$E[x_t] = \mu; \quad V[x_t] = \sigma^2; \quad COV[x_t, x_{t-k}] = \gamma_k. \tag{13.2}$$

Diagnostic checking of the overall ARMA models is done by the residuals. Several tests have been proposed, among them the most popular one seems to be the so-called portmanteau test proposed by [11], and its robust version by [12]. These tests are based on a sum of squared correlations of the estimated residuals suitably scaled.

### *13.2.2 Non-linear Models*

Theory and practice are mostly concerned with linear methods and models, such as ARIMA models and exponential smoothing methods. However, many time series exhibit features which cannot be explained in a linear framework. For example some economic series show different properties when the economy is going into, rather than coming out of, recession. As a result, there has been increasing interest in non-linear models.

Many types of non-linear models have been proposed in the literature. For example, see bilinear models [13], classification and regression trees [14], threshold autoregressive models [15] and Projection Pursuit Regression [16]. The rewards from using non-linear models can occasionally be substantial. However, on the other side, it is generally more difficult to compute forecasts more than one step ahead [17].

Another important class of non-linear models is that of non-linear ARMA models (NARMA) proposed by [18], which are generalizations of the linear ARMA models to the non-linear case. A NARMA model obeys the following equations:

$$x_t = h(x_{t-1}, x_{t-2}, \ldots, x_{t-p}, \varepsilon_{t-1}, \ldots, \varepsilon_{t-q}) + \varepsilon_t, \tag{13.3}$$

where $h$ is an unknown smooth function, and as in Sect. 13.2.1 it is assumed that $E\left[\varepsilon_t | x_{t-1},\ \ x_{t-2}, \ldots\right] = 0$ and that variance of $\varepsilon_t$ is $\sigma^2$. In this case the conditional mean predictor based on the infinite past observation is

$$\hat{x}_t = E[h(x_{t-1}, x_{t-2}, \ldots, x_{t-p}, \varepsilon_{t-1}, \ldots, \varepsilon_{t-q}) | x_{t-1}, x_{t-2}, \ldots]. \tag{13.4}$$

Suppose that the NARMA model is invertible in the sense that there exists a function $\nu$ such as

$$x_t = \nu(x_{t-1}, x_{t-2}, \ldots) + \varepsilon_t. \tag{13.5}$$

Then given the infinite past of observations $x_{t-1}, x_{t-2}, \ldots$, one can compute the $\varepsilon_{t-1}$ in (13.3) exactly:

$$\varepsilon_{t-j} = \kappa(x_{t-j}, x_{t-j-1}, \ldots),\ \ j = 1, 2, \ldots, q. \tag{13.6}$$

In this case the mean estimate is

$$\hat{x}_t = h(x_{t-1}, x_{t-2}, \ldots, x_{t-p}, \varepsilon_{t-1}, \ldots, \varepsilon_{t-q}), \tag{13.7}$$

where the $\varepsilon_{t-j}$ are specified in terms of present and past $x_u$'s. The predictor of (13.7) has a mean square error $\sigma^2$.

Since we have only a finite observation record, we cannot compute (13.6) and (13.7). It seems reasonable to approximate the conditional mean predictor (13.7) by the recursive algorithm

$$\hat{x}_t = h(x_{t-1}, x_{t-2}, \ldots, x_{t-p}, \hat{\varepsilon}_{t-1}, \ldots, \hat{\varepsilon}_{t-q}), \tag{13.8}$$

$$\hat{\varepsilon}_{t-j} = x_{t-j} - \hat{x}_{t-j},\ \ j = 1, 2, \ldots, q, \tag{13.9}$$

with the following initial conditions

$$\hat{x}_0 = \hat{x}_{-1} = \cdots = \hat{x}_{-p+1} = \hat{\varepsilon}_0 = \cdots = \hat{\varepsilon}_{-q+1} = 0. \tag{13.10}$$

For the special case of non-linear autoregressive model (NAR), it is easy to check that (13.3) is given by

$$x_t = h(x_{t-1}, x_{t-2}, \dots, x_{t-p}) + \varepsilon_t. \tag{13.11}$$

In this case, the minimum mean square error (MSE) optimal predictor of $x_t$ given $x_{t-1}, x_{t-2}, \dots, x_{t-p}$ is the conditional mean (for $t \geq p + 1$).

$$\hat{x}_t = E[x_t | x_{t-1}, \dots, x_{t-p}] = h(x_{t-1}, \dots, x_{t-p}). \tag{13.12}$$

This predictor has mean square error $\sigma^2$.

### *13.2.3 Concept Drift*

A challenging scenario for time series is when the underlying law of probability changes. It is known as *concept drift* and it refers to the fact that data is obtained from a non-stationary environment [19]. In the classification setting, we can describe concept drift as any scenario where the posterior probability of a class $c_k$ that a given instance belongs, $P(c_k|x) = P(x|c_k)P(c_k)/P(x)$, changes over time, i.e. $P_{t+1}(c_k|x) \neq P_t(c_k|x)$. $P(x)$ describes probability of an instance $x \in \mathscr{X}$. $P(c_k|x)$ is the likelihood of observing a data point $x$ within a particular class. $P(c_k)$, defines the class prior probabilities, and relates class balance to the overall underlying law of probability. Note that an observation of change in $P(x)$ might be insufficient to characterize a concept drift, because of its independence of the class labels.

There are two ways to make a model adapt to a concept-drift. The first one consists in adapting a single model so that it can learn example by example and adapt to a changing pattern. Another approach is to use an ensemble of models. Zang et al. [20] conclude that both have the same capacity to handle stream data, including the presence of concept drift. Incremental single algorithms have a better performance in terms of efficiency, but the ensembles adapt better to the concept drift and are more stable.

## 13.3 Elements of the Ensemble Architecture

Ensemble methods are based on the idea of combining a set of individual predictors $\{f_1, f_2, \dots, f_N\}$, building a decision function $F$ by means of an aggregation operator that combines the individual forecasts instead of the popular keep-the-best (KTB) model [21, 22]. Ensemble methods can differ in three different stages: manipulating data, manipulating the base learner, and the function that the ensemble uses to generate the "consensus" output.

### 13.3.1 Data Manipulation

This is the most popular way to introduce diversity in a ensemble. There are different forms to manipulate the data: Implicit diversity can be introduced by using a random sample $S^t$ to train each learner $n$ according to a distribution $D$ which can be uniform as in Bagging [23], or $S^n$ can be created by using a weight distribution over the examples on the training set, as in Wagging [24]. Explicit diversity approaches, like Adaboost [25] and its variants, compute the distribution of instances for each learner according to a metric which depends on the previous ensemble.

Another way to encourage diversity is manipulating either the inputs or the outputs. For example, input smearing [26] enhances the diversity of Bagging by adding Gaussian noise to the inputs. The amount of noise, and therefore the amount of diversity, is controlled by a parameter. Experimental results show that this technique can outperform Bagging in terms of generalization error. Rodriguez et al. [27] encourage diversity by randomly splitting the feature set into $K$ disjoint subsets. Next, Principal Component Analysis (PCA) is applied to each subset retaining all principal components. On the other hand, Breiman proposes output smearing [28] that adds noise to the targets in the same way as input smearing.

The next two sub-categories involve *sub-sampling*, which is a widely-known technique commonly used to cut down on the memory size and the computational cost of the training process. But also this technique contributes to implicitly encourage the diversity in the ensemble, hence, improving performance. For example, Zhang and Berardi [22] propose a neural network ensemble with both systematic and serial partitioning to build the sub-samples. The first partition method divides data into $k$ subsamples, where $k$ is the input lag that the network uses. Each data partition consists of a set of $k$-lag input vectors, selected from all $k$-lag input vectors throughout the original training sample, with non-overlapping examples to feed each ensemble member. According to the authors, this subsampling scheme diminishes the correlation among the ensemble predictors. The second data partitioning technique divides data into $k$ disjoint subsamples based on the chronological time sequence. Thus, each neural network trains with one of the subsamples, obtaining an ensemble of $k$ models. On the other hand, Subagging [29, 30] trains each predictor by using a subsample of size $m$ from the original training set of $M$ examples without replacement. This approach has performed similarly to Bagging for high-dimensional datasets. Crogging [31] uses cross-validation (CV) samples to train each forecasting model. The authors test different cross validation methods to encourage diversity among the neural networks. Crogging averages the learner outputs to obtain the aggregated prediction. Experiments show that this algorithm outperforms Bagging and KBT model.

Finally, unlike horizontal partitioning methods, vertical (Feature Set) partitioning techniques keep all the training examples in each subsample; however, each of them contains a subset of the original set of features. Thus, each classifier learns a different projection of the training set. This diversity can reduce the correlation among the predictors [32]. However, in classification setting this methods might lead to imbalanced class problems [33]. Several researchers have proposed algorithms which belong to this category, for example, Tumer and Oza [34] present an ensemble algorithm which trains each predictor with a subset of features selected depending on the correlations between individual features and class labels. The authors claim that this approach reduces dimensionality and improves generalization performance. Bryll et al. [32] propose attribute bagging (AB) which selects each subset of features by randomly selecting the features without replacement. Then, the number of features is selected by computing the classification performance of different sized random subsets of attributes.

### *13.3.2 Manipulating the Base Learner*

Introducing differences among the learners is a well-known strategy to add diversity, and there are different forms to apply it. First, depending on the optimization method, diversity can be encouraged in the ensemble by starting the search in the hypothesis space from different points. Since back-propagation can lead to local optima, assigning different initial weights to the neural networks that compound the ensemble [35] can implicitly generate diversity. It is generally accepted that this way of introducing diversity obtains poor performance [36]. The second way is to manipulate the inducer's parameters, for example, using neural networks as base learners, and several attempts to change the network architecture have been proposed, as in [37, 38]. Opitz and Shavlik vary the number of the nodes or use different neural networks topology [39]. The first approach is not effective in practice, but the second method effectively encourages the diversity and the performance. The third group of techniques encourages explicit diversity by adding a regularization term in the optimization function of each learner. Negative correlation algorithms [40, 41] are the most studied approaches in this category. These methods will be studied in more detail in Sect. 13.5.3. Finally the fourth class of methods induce implicit diversity by using different base learners. The gist of this idea is to combine the best of each base learner. Even more ambitiously, these techniques hope that the synergistic effects of this combination may improve the performance of the ensemble [33]. Wichard et al. [42] state that a heterogeneous ensemble can reduce the variance because the individual errors are strongly uncorrelated. For instance, Woods et al. [43] combine four types of base learners: decision trees, neural networks, K-nearest neighbors and quadratic Bayes. Given a new instance from the testing-set, the output is obtained with model selection, that is, the algorithm chooses the learner classifier with the highest local accuracy in the feature space. Wang et al. [44] combine decision trees and neural networks in medical classification. Using different base learners implies that the bias of the learner has to match with the characteristics of the application domain [45]. Thus, every ensemble in this category needs an exhaustive tune in stage, in order to pick the appropriate base learners [46].

### *13.3.3 Aggregation Function*

There are a wide variety of methods for combining the individual outputs of the predictors, and the most common in the literature are the following:

1. Linear combination: In regression settings, the joint decision $F$ is commonly obtained by a linear combination of the individual hypotheses [6].

$$F(\mathbf{x}) = \sum_{n=1}^{N} \beta_n f_t(\mathbf{x}). \tag{13.13}$$

   The weight of each model is usually computed with some quality criteria: (i) it can be assigned to be the inverse of the previous forecast error (MSE, MAPE, MAE, etc.) [47]. This is called *error based* approaches. (ii) the optimal weights are obtained by minimizing the total sum of squared error (SSE). This is called variance-based method [47].
2. Simple Average: A particular case of the linear combination is the simple average, all hypothesis weights are equal, this is, $\beta_n = 1/N, \forall n = 1, \ldots, N$ [5, 48].
3. Trimmed mean: This method is an adaptation of the simple average. To predict a single point, it removes the largest and smallest individual predictions before computing the average [5].
4. Windorized Mean: This is another average-based approach. To obtain the prediction of a single instance, it replaces the smallest and largest individual predictions with the predictions closest to them. Next, simple average is computed over the modified set of predictions [48].

## 13.4 Why Ensemble?

Dietterich [49] identified statistical, computational and representational reasons that support these premises. Let us suppose that we have a set of predictors $\{f_1, f_2, \ldots, f_N\}$ with good performance in the training set $S^m$.

1. **Statistical**. We can pick one of them taking the risk that this single classifier might not necessarily be the best choice, since we do not know the exact probability function underlying the data. We can create a set of classifiers with zero training error [50], and therefore, all predictors will become indistinguishable.
2. **Computational**. Some training methods, such as hill-climbing or gradient descent, may fall into local optima and therefore, the initial parameters of each machine can affect the choice of the final hypothesis, obtaining a solution that is far from the optimal model.
3. **Representational**. The hypothesis space of the classifiers used to approximate the underlying function $f_0$ to the data may not contain the optimal prediction. For example, an ensemble of linear functions can approximate any decision function

with pre-determined performance [51]. Some base learners, such as neural networks, with a finite training sample select the first hypothesis that fits the training data. Thus, the effective space of hypotheses $\mathscr{H}'$ by the learner is substantially smaller than the hypotheses space $\mathscr{H}$. In addition, one can note the simplicity of combining weak predictors, in contrast to the use of a single highly complex machine. For this reason, ensemble approaches have been successful in solving real problems, which has made them an increasingly studied discipline.

The remarkable point is that with an appropriate design, the expected performance of the combined predictor can be better than the average performance of individual predictors, even if each of them is weakly good [52].

## 13.5 The Most Popular Ensemble Approaches

As we previously mentioned, different ensemble approaches are distinguished by the manner in which they manipulate the training data, by the way in which they select the individual hypotheses and by the way in how those are aggregated to the final decision. Now we will review the most commonly used ensemble approaches, namely Bagging, AdaBoost and Negative Correlation.

### *13.5.1 Bagging*

Bagging, introduced by Breiman [23] for classification and regression settings, trains each learner using a bootstrap sample from the training set $S^m$ and combines each output by uniform averaging. This technique can be viewed as an algorithm with only an implicit search for diversity, because no information about the predictions of the other models is incorporated to generate each individual predictor. According to Breiman, by averaging a set of predictors trained with a different bootstrap sample, this method might reduce the influence of the outliers [53, 54]. From this point of view, resampling has an effect similar to that of robust M-estimators in statistics [55], where the influence of sample points is bounded using appropriate loss functions, for example, Huber loss or Tukey's bi-square loss. Hence, this approach might considerably reduce the variance of the bagged prediction, especially when the base learner is unstable [23].

However, traditional bootstrap breaks the sequence of data points which may lead to a bad generalization in time series problems. In order to deal with this important issue, Inoue and Kilian adapt the way that Bagging construct its training samples [56]. Instead, the authors use resampling blocks of instances from the original training set with replacement, in order to encourage diversity, and at the same time, to capture the dependence among the examples. This procedure is called *block bootstrap*, and the block size is chosen to capture the dependence in error term [57]. A simulation study reveals that bagging considerably reduces the out-of-sample mean squared error of the predictions. In [58] the training samples keep the temporal order of the points, by introducing a weight for each example that corresponds to the number of times that the point is selected in the bootstrap sample. Thus, if an example has not been selected in the sample, its weight is equal to zero. Finally, the output of the ensemble for a single instance is computed as a weighted average, where the weight of each predictor is the weighted root mean squared error (WRMSE).

### *13.5.2 AdaBoost*

In 1997, Freund and Schapire introduced Adaboost [25], a sequentially coupled approach that fits an additive model through a forward stagewise strategy [8], by training a single predictor to minimize the residual of the previous ensemble in each round. This algorithm has a theoretical background based on the "PAC" learning model [59]. The gist of this model is to create a method that combines a group of weak learners to produce a strong learning algorithm. This approach provides good generalization ability but it is computationally expensive since the learners are coupled in a sequential manner. The main idea of AdaBoost is to maintain a sampling distribution over the training set. When the algorithm begins, the distribution is uniform, that is, all weights are set to $w_m^1 = \frac{1}{N}, m = 1, \ldots, M$. Here, $w_m^n$ is the weight of the $m$th example at round $n$. At each round of the algorithm, the weights of the incorrectly classified examples are increased, in such a way that in the following step the weak learner is forced to focus on the misclassified examples of the training set. Thus, the algorithm concentrates on the "hard" examples, instead of on the correctly classified examples. In normal scenarios, this approach usually outperforms Bagging. However, under noisy data it is the robust performance of Bagging that is commonly superior, rather than Adaboost which considerably decreases its generalization capacity [60].

In time series, Shresta and Solomatine introduce AdaBoost.RT [61], where the absolute relative error (ARE) for the learner $n$ is computed by considering the error over a threshold $\phi$. Also, the updating rule reduces the weights of the instances with errors lower than $\phi$, and subsequently, the weights are then normalized to make them a distribution. Finally, $F$ is obtained as the weighted average of the individual learners, with weights equal to $-\log(\beta_n)$. It is worth noting that both algorithms have no proof of convergence [61, 62]. Canestrelli uses Drucker's AdaBoost.R2 [62] for tide levels forecasting, which quantifies the loss for each training sample using one of the three possible functions:

1. Absolute percentage error (APE) $w_m^n = \frac{|f_n(\mathbf{x}_m)-y_m|}{\max_{m=1,\ldots,M}|f_n(\mathbf{x}_m)-y_m|}$.
2. Quadratic $w_m^n = (f_n(\mathbf{x}_m) - y_m)^2 \max_{m=1,\ldots,M} f_n(\mathbf{x}_m) - y_m)^2$.
3. Exponential: $1 - \exp \frac{-|f_n(\mathbf{x}_m)-y_m|}{\max_( m=1,\ldots,M)|f_n(\mathbf{x}_m)-y_m|}$.

Then, the confidence of the predictor is computed as a function of the weighted average error according to the weight distribution. And instances with higher errors increase its weights in order to have more chances to appear in the next training sample. Finally, the output of the ensemble is computed as the weighted median. Assaad et al. adapt this method for recurrent neural networks as the base learner [63]. They make two changes: first, in order to keep the temporal dependence of the training data, they weight the example errors in order to update the RNN weights instead of resampling. Second, an additional parameter is introduced to increase (even more) the weight of the wrong modeled examples. On the other hand, de Souza et al. [64] propose the BCC algorithm, an adaptation of Adaboost.R2 technique that uses the Genetic Programming (GP) as the base learner. It uses the exponential loss function to compute the error of each example. And, in order to update the weights, it includes a multiplicative factor which depends on the correlation between the learner predictions and the true targets. Finally, Goh et al. [65] introduce Modified Adaboost, which combines Ellman Networks using a weighted linear function. Besides, it computes the error of each instance by considering two loss functions widely used for comparing drug dissolution profiles.

According to the gradient descent perspective of Adaboost [8, 66], its goal is to add the i-th learner in the ensemble which minimizes

$$(\beta_n, f_n) = \operatorname{argmin}_{(\beta, f)} \sum_{m=1}^{M} \ell(y_m, F(\mathbf{x}_m)), \tag{13.14}$$

where $F(\mathbf{x}_m) = F_{n-1}(\mathbf{x}_m) + \beta_n f_n(\mathbf{x}_m)$. At each round, the algorithm selects the hypothesis $f_n$ which minimizes Eq. 13.14 over the sample distribution $w^n$. Then it performs a line search along the direction. From this point of view, Buhlmann and Yu [67] introduce a gradient boosting method with the quadratic loss function which is adapted for time series forecasting for load forecasting [68], finance [69], economics [70] and production [71].

## 13.5.3 *Negative Correlation Methods*

A very important feature of an ensemble is the diversity of their members, because a group of learners composed exclusively by exact replicas of the same hypothesis is clearly useless. A way to quantify this property is the so called *Bias-Variance-Covariance* decomposition [72][1] for the quadratic loss of an ensemble $F$ obtained as a convex combination of a set of $N$ predictors and the optimal prediction $f_0(\mathbf{x})$

[1]If the hypotheses are considered fixed, the expectations are taken based on the distribution of the inputs $\mathbf{x}$. If they are considered free, expectations are also taken with respect to the distribution of the sample(s) used to estimate them.

$$E[f_0(\mathbf{x}) - F(\mathbf{x})^2] = \text{bias}(F(\mathbf{x}))^2 + \sum_{t=1}^{T} w_t^2 \text{var}(f_n(\mathbf{x})) + E\left[\sum_i \sum_{i \neq j} w_i w_j \text{cov}(f_i, f_j)\right], \tag{13.15}$$

where $\text{bias}(F(\mathbf{x})) = E[F(\mathbf{x}) - f_0(\mathbf{x})] = \sum_{t=1}^{T} w_t E[f_n(\mathbf{x}) - f_0(\mathbf{x})]$, $\text{var}(f_n(\mathbf{x})) = E\left[(f_n(\mathbf{x}) - E[f_n(\mathbf{x})])^2\right]$, and $\text{cov}(f_i, f_j) = E\left[(f_i(\mathbf{x}) - E[f_i(\mathbf{x})])(f_j(\mathbf{x}) - E[f_j(\mathbf{x})])\right]$.

This decomposition suggests that selecting a set of hypotheses with negative covariance $\text{cov}(f_i, f_j)$ without increasing the individual biases and variances would be a very convenient scenario, since this would minimize the expected quadratic loss [8, 41, 72, 73]. Several methods based on the negative correlation among the learners emerge from this result. In the seminal work of Rosen [74], for example, the hypotheses are generated sequentially using a base learner which, at each step $t$, is provided with a different loss function that directly penalizes correlations with previous errors,

$$\ell_R(f_n(\mathbf{x}), y) = (y - f_i(\mathbf{x}))^2 + \lambda \sum_{j=1}^{t-1} d(t, j)(f_n(\mathbf{x}) - y)(f_j(\mathbf{x}) - y), \tag{13.16}$$

where $\lambda$ is a scaling function when $t = j - 1$ and otherwise 0. Note that the learner always minimizes the error within the original set of examples. However, at each step, the learner minimizes a different loss function $\ell_R$ and therefore, the target hypothesis changes from round to round depending on the previous learning step.

In the Negative Correlation algorithm (NC) [41], the $N$ hypotheses $f_1, \ldots, f_N$ are trained simultaneously. The loss function to be minimized by each individual hypothesis $f_n$ is given by equation

$$\ell_{NC}(f_n(\mathbf{x}), y) = (y - f_n(\mathbf{x}))^2 + \lambda \sum_{j \neq n}^{n-1} (f_n(\mathbf{x}) - F(\mathbf{x}))(f_j(\mathbf{x}) - F(\mathbf{x})). \tag{13.17}$$

Note that both $\ell_R$ and $\ell_{NC}$ promote the individual accuracy by penalizing the differences $y - f_n(\mathbf{x})$. However, the correlations among the individual predictions $f_n(\mathbf{x})$ are no longer measured with respect to the target prediction $y$ but with respect to the composite prediction $F(\mathbf{x})$. This is taking into account that equation (13.17) can be expressed as

$$\ell_{NC}(f_n(\mathbf{x}), y) = (y - f_n(\mathbf{x}))^2 + \lambda(f_n(\mathbf{x}) - F(\mathbf{x})) \sum_{j \neq n}^{n-1} (f_j(\mathbf{x}) - F(\mathbf{x})). \tag{13.18}$$

In each training epoch, $F(\mathbf{x})$ is computed as the uniform average of the predictions made for the whole ensemble in the previous timestep.

Note that $\sum_{j \neq n} (f_j(\mathbf{x}) - F) = -(f_n - F)$, since $F(\mathbf{x}) = \frac{1}{N}\left[f_n + \sum_{j \neq n} f_j(\mathbf{x})\right]$. Thus, $\ell_{NC}$ can be re-written as

$$\ell_{NC}(f_n(\mathbf{x}), y) = (y - f_n(\mathbf{x}))^2 - \lambda(f_n(\mathbf{x}) - F(\mathbf{x}))^2. \tag{13.19}$$

Rodan and Tiño [75] propose an ensemble of Echo State Networks [76] with diverse reservoirs, where the negative correlation term defined in equation (13.18) is used to compute their collective read-out.

Another point of interest is that the Negative Correlation algorithm can also be motivated by the ambiguity decomposition [77]

$$(y - F(\mathbf{x}))^2 = \sum_{n=1}^{N} w_n(y - f_n(\mathbf{x}))^2 - \sum_{n=1}^{N} w_n(f_n(\mathbf{x}) - F(\mathbf{x}))^2. \tag{13.20}$$

It states that the quadratic loss of the ensemble is the weight average of the individual errors, minus the weighted average of the individual deviations with respect to the ensemble output. The above-mentioned decomposition suggests that the ensemble error can be reduced by increasing the second term which is called the *ambiguity* term [73]. However, uncontrolled deviations can lead to increasing the first term in (13.20), hence, increasing the ensemble error. Note that the loss given in equation (13.18) is essentially the above mentioned decomposition corresponding to learner $t$.

**Acknowledgments** We thank to Professor Claudio Moraga for their continued support in the design and consolidation of the doctoral program in computer engineering of the Universidad Técnica Federico Santa María.
This work was supported in part by Research Project DGIP-UTFSM (Chile) 116.24.2 and in part by Basal Project FB 0821.

## References

1. George EP Box, Gwilym M Jenkins, and Gregory C Reinsel. *Time series analysis: forecasting and control*. Prentice Hall Englewood cliffs nj, third edition edition, 1994.
2. Jan G De Gooijer and Rob J Hyndman. 25 years of time series forecasting. *International journal of forecasting*, 22 (3):443–473, 2006.
3. John M Bates and Clive WJ Granger. The combination of forecasts. *Journal of the Operational Research Society*, 20(4):451–468, 1969.
4. G Peter Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.
5. Lilian M de Menezes, Derek W. Bunn, and James W Taylor. Review of guidelines for the use of combined forecasts. *European Journal of Operational Research*, 120(1):190 – 204, 2000.
6. Ratnadip Adhikari and R. K. Agrawal. Combining multiple time series models through a robust weighted mechanism. In *1st International Conference on Recent Advances in Information Technology, RAIT 2012, Dhanbad, India, March 15-17, 2012*, pages 455–460. IEEE, 2012.
7. Hui Zou and Yuhong Yang. Combining time series models for forecasting. *International Journal of Forecasting*, 20(1):69–84, 2004.

8. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.
9. Hector Allende and Siegfried Heiler. Recursive generalized m-estimates for autoregressive moving average models. *Journal of Time Series Analysis*, 13(1):1–18, 1992.
10. Bruce L Bowerman and Richard T O'Connell. Forecasting and time series: An applied approach. 3rd. 1993.
11. Greta M Ljung and George EP Box. On a measure of lack of fit in time series models. *Biometrika*, 65(2):297–303, 1978.
12. H Allende and J Galbiati. Robust test in time series model. *J. Interamerican Statist. Inst*, 1(48):35–79, 1996.
13. T Subba Rao. On the theory of bilinear time series models. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 244–255, 1981.
14. Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.
15. *Vector Autoregressive Models for Multivariate Time Series*, pages 385–429. Springer New York, New York, NY, 2006.
16. Jerome H Friedman. Multivariate adaptive regression splines. *The annals of statistics*, pages 1–67, 1991.
17. Jin-Lung Lin and Clive WJ Granger. Forecasting from non-linear models in practice. *Journal of Forecasting*, 13(1):1–9, 1994.
18. Jerome T Connor, R Douglas Martin, and Les E Atlas. Recurrent neural networks and robust time series prediction. *IEEE transactions on neural networks*, 5(2):240–254, 1994.
19. Indre Zliobaite. Learning under concept drift: an overview. *CoRR*, abs/1010.4784, 2010.
20. Wenyu Zang, Peng Zhang, Chuan Zhou, and Li Guo. Comparative study between incremental and ensemble learning on data streams: Case study. *Journal Of Big Data*, 1(1), 2014.
21. Graham Elliott, Clive Granger, and Allan Timmermann, editors. *Handbook of Economic Forecasting*, volume 1. Elsevier, 1 edition, 2006.
22. P. G. Zhang and L. V. Berardi. Time series forecasting with neural network ensembles: an application for exchange rate prediction. *Journal of the Operational Research Society*, 52(6):652–664, 2001.
23. L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
24. E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–139, 1999.
25. Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997.
26. Eibe Frank and Bernhard Pfahringer. *Improving on Bagging with Input Smearing*, pages 97–106. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.
27. Juan J. Rodriguez, Ludmila I. Kuncheva, and Carlos J. Alonso. Rotation forest: A new classifier ensemble method. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(10):1619–1630, October 2006.
28. Leo Breiman. Randomizing outputs to increase prediction accuracy. *Machine Learning*, 40(3):229–242, 2000.
29. P. Bühlmann and Bin Yu. Analyzing bagging. *Annals of Statistics*, 30:927–961, 2002.
30. J.H. Friedman and P. Hall. On bagging and nonlinear estimation. *Journal of Statistical Planning and Inference*, 137, 2000.
31. D. K. Barrow and S. F. Crone. Crogging (cross-validation aggregation) for forecasting x2014; a novel algorithm of neural network ensembles on time series subsamples. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–8, Aug 2013.
32. R. K. Bryll, R. Gutierrez-Osuna, and F. K. H. Quek. Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets. *Pattern Recognition*, 36(6):1291–1302, 2003.
33. L. Rokach. Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography. *Computational Statistics & Data Analysis*, 53(12):4046–4072, 2009.

34. K. Tumer and N. C. Oza. Input decimated ensembles. *Pattern Analysis and Applications*, 6(1):65–77, 2003.
35. J. F. Kolen, J. B. Pollack, J. F. Kolen, and J. B. Pollack. Back propagation is sensitive to initial conditions. In *Complex Systems*, pages 860–867. Morgan Kaufmann, 1990.
36. G. Brown, J. L. Wyatt, R. Harris, and Xin Yao. Diversity creation methods: a survey and categorisation. *Information Fusion*, 6(1):5–20, 2005.
37. D. Partridge and W. B. Yates. Engineering multiversion neural-net systems. *Neural Computation*, 8:869–893, 1995.
38. W. Yates and D. Partridge. Use of methodological diversity to improve neural network generalization. *Neural Computing and Applications*, 4(2):114–128, 1996.
39. D. W. Opitz and J. W. Shavlik. Generating accurate and diverse members of a neural-network ensemble. In *Advances in Neural Information Processing Systems*, pages 535–541. MIT Press, 1996.
40. R. Ñanculef, C. Valle, H. Allende, and C. Moraga. Training regression ensembles by equential target correction and resampling. *Inf. Sci.*, 195:154–174, July 2012.
41. Yong Liu and Xin Yao. Ensemble learning via negative correlation. *Neural Networks*, 12:1399–1404, 1999.
42. Ogorzalek M. Wichard JD, Christian M. Building ensembles with heterogeneous models. In *7th Course on the International School on Neural Nets IIASS*, 2002.
43. K. S. Woods, W. P. Kegelmeyer, and K. W. Bowyer. Combination of multiple classifiers using local accuracy estimates. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(4):405–410, 1997.
44. Wenjia Wang, P. Jones, and D. Partridge. Diversity between neural networks and decision trees for building multiple classifier systems. In Josef Kittler and Fabio Roli, editors, *Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science*, pages 240–249. Springer, 2000.
45. P. Brazdil, J. Gama, and B. Henery. Characterizing the applicability of classification algorithms using meta-level learning. In F. Bergadano and L. De Raedt, editors, *ECML*, volume 784 of *Lecture Notes in Computer Science*, pages 83–102. Springer, 1994.
46. Niall Rooney, David Patterson, Sarab Anand, and Alexey Tsymbal. *Dynamic Integration of Regression Models*, pages 164–173. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.
47. Christiane Lemke and Bogdan Gabrys. Meta-learning for time series forecasting and forecast combination. *Neurocomputing*, 73(10-12):2006–2016, 2010.
48. Victor Richmond R. Jose and Robert L. Winkler. Simple robust averages of forecasts: Some empirical results. *International Journal of Forecasting*, 24(1):163 – 169, 2008.
49. T. G. Dietterich. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, MCS '00, pages 1–15, London, UK, UK, 2000. Springer-Verlag.
50. D. M. Hawkins. The Problem of Overfitting. *Journal of Chemical Information and Computer Sciences*, 44(1):1–12, 2004.
51. L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley and Sons, Inc., 2004.
52. L.K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.
53. Y. Grandvalet. Bagging down-weights leverage points. In *IJCNN (4)*, pages 505–510, 2000.
54. Y. Grandvalet. Bagging equalizes influence. *Machine Learning*, 55(3):251–270, 2004.
55. P. J. Huber. *Robust Statistics*. Wiley Series in Probability and Statistics. Wiley-Interscience, 1981.
56. Atsushi Inoue and Lutz Kilian. Bagging time series models. *CEPR Discussion Paper No. 4333*, 2004.
57. Peter Hall and Joel L. Horowitz. Bootstrap Critical Values for Tests Based on Generalized-Method-of-Moments Estimators. *Econometrica*, 64(4):891–916, 1996.
58. Nikola Simidjievski, Ljupčo Todorovski, and Sašo Džeroski. Predicting long-term population dynamics with bagging and boosting of process-based models. *Expert Syst. Appl.*, 42(22):8484–8496, December 2015.

59. L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.
60. P. Melville, N. Shah, L. Mihalkova, and R. J. Mooney. Experiments on ensembles with missing and noisy data. In *In: Proceedings of the Workshop on Multi Classifier Systems*, pages 293–302. Springer Verlag, 2004.
61. D. L. Shrestha and D. P. Solomatine. Experiments with adaboost.rt, an improved boosting scheme for regression. *Neural Comput.*, 18(7):1678–1710, July 2006.
62. H. Drucker. Improving regressors using boosting techniques. In D. H. Fisher, editor, *ICML*, pages 107–115. Morgan Kaufmann, 1997.
63. Mohammad Assaad, Romuald Boné, and Hubert Cardot. A new boosting algorithm for improved time-series forecasting with recurrent neural networks. *Inf. Fusion*, 9(1):41–55, January 2008.
64. Luzia Vidal de Souza, Aurora Pozo, Joel Mauricio Correa da Rosa, and Anselmo Chaves Neto. Applying correlation to enhance boosting technique using genetic programming as base learner. *Applied Intelligence*, 33(3):291–301, 2010.
65. Wei Yee Goh, Chee Peng Lim, and Kok Khiang Peh. Predicting drug dissolution profiles with an ensemble of boosted neural networks: a time series approach. *IEEE Transactions on Neural Networks*, 14(2):459–463, 2003.
66. Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.
67. Buhlmann P. and Yu B. Boosting with the l2 loss: Regression and classification. *Journal of the American Statistical Association*, 98:324–339, 2003.
68. Souhaib Ben Taieb and Rob Hyndman. A gradient boosting approach to the kaggle load forecasting competition. *International Journal of Forecasting*, 30(2):382–394, 2014.
69. Francesco Audrino and Peter Bühlmann. Splines for financial volatility. University of st. gallen department of economics working paper series 2007, Department of Economics, University of St. Gallen, 2007.
70. Klaus Wohlrabe and Teresa Buchen. Assessing the macroeconomic forecasting performance of boosting: Evidence for the united states, the euro area and germany. *Journal of Forecasting*, 33(4):231–242, 2014.
71. Nikolay Robinzonov, Gerhard Tutz, and Torsten Hothorn. Boosting techniques for nonlinear time series models. *AStA Advances in Statistical Analysis*, 96(1):99–122, 2012.
72. N. Ueda and R. Nakano. Generalization error of ensemble estimators. In *Proceedings of IEEE International Conference on Neural Networks.*, pages 90–95, Washington, USA, June 1996.
73. G. Brown. *Diversity in Neural Network Ensembles*. PhD thesis, The University of Birmingham, 2004.
74. B. Rosen. Ensemble learning using decorrelated neural networks. *Connection Science*, 8:373–384, 1996.
75. Ali Rodan and Peter Tiño. Negatively correlated echo state networks. In *ESANN 2011, 19th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 27-29, 2011, Proceedings*, 2011.
76. Mustafa C. Ozturk, Dongming Xu, and José C. Príncipe. Analysis and design of echo state networks. *Neural Comput.*, 19(1):111–138, January 2007.
77. A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. In *Advances in Neural Information Processing Systems*, pages 231–238. MIT Press, 1995.