M.Sc. Engg. Thesis

A Novel Layer Based Ensemble Architecture for Time Series Forecasting

by

Md. Mustafizur Rahman

Submitted to

Department of Computer Science and Engineering

in partial fulfilment of the requirements for the degree of
Master of Science in Computer Science and Engineering

Department of Computer Science and Engineering

Bangladesh University of Engineering and Technology (BUET)

Dhaka 1000

August 2013

Dedicated to my loving parents

Author's Contact

Md. Mustafizur Rahman

Lecturer

Department of Computer Science & Engineering

Bangladesh University of Engineering & Technology (BUET).

Email: mailto:md.mustafizur.rahman@csebuet.orgmd.mustafizur.rahman@csebuet.org

The thesis titled "A Novel Layer Based Ensemble Architecture for Time Series Forecasting", submitted by Md. Mustafizur Rahman, Roll No. **0411052015P**, Session April 2011, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering and approved as to its style and contents. Examination held on August 24, 2013.

Board of Examiners

1. _____

Dr. Md. Monirul Islam — Chairman
Professor — (Supervisor)
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology, Dhaka.

2. _____

Dr. Abu Sayed Md. Latiful Hoque — Member
Head and Professor — (Ex-Officio)
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology, Dhaka.

3. _____

Dr. Md. Monirul Islam — Member
Assistant Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology, Dhaka.

4. _____

Dr. A. B. M. Alim Al Islam — Member
Assistant Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology, Dhaka.

5. _____

Dr. Mohammad Nurul Huda — Member
Professor — (External)
Department of Computer Science and Engineering
United International University, Dhaka.

Candidate's Declaration

This is hereby declared that the work titled "A Novel Layer Based Ensemble Architecture for Time Series Forecasting" is the outcome of research carried out by me under the supervision of Dr. Md. Monirul Islam, in the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka 1000. It is also declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.

Md. Mustafizur Rahman

Candidate

Acknowledgment

I express my heart-felt gratitude to my supervisor, Dr. Md. Monirul Islam for his constant supervision of this work. He helped me a lot in every aspect of this work and guided me with proper directions whenever I sought one. His patient hearing of my ideas, critical analysis of my observations and detecting flaws (and amending thereby) in my thinking and writing have made this thesis a success.

I would also want to thank the members of my thesis committee for their valuable suggestions. I thank Dr. Md. Monirul Islam, Dr. A. B. M. Alim Al Islam and specially the external member Dr. Mohammad Nurul Huda.

In this regard, I remain ever grateful to my beloved parents, who always exists as sources of inspiration behind every success of mine I have ever made.

Abstract

Time series forecasting (TSF) has been widely used in many application areas such as science, engineering, and finance. Usually the characteristics of phenomenon generating a series are unknown and the information available for forecasting is limited to the past values of the series. It is, therefore, important to use an appropriate number of past values, termed lag, for forecasting. Although ensembles (combining several learning machines) have been widely used for classification problems, there is only a handful work for TSF problems. Existing algorithms for TSF construct ensembles by combining base predictors involving different training parameters or data sets . The idea of ensemble is also employed to find the optimal parameter of predictors used for TSF. The aim of using different parameters or data sets is to maintain diversity among the learning machines in an ensemble. It has been known that the performance of ensembles greatly depends not only on diversity but also on accuracy of the learning machines. However, the issue of accuracy is totally ignored in ensemble approaches used for forecasting.

This thesis proposes a layered ensemble architecture (LEA) for TSF. Our LEA is consisted of two layers. Each of the layers uses a neural network ensemble. However, tasks of ensembles in the two layers are different. While the ensemble of the first layer tries to find an appropriate time window of a given time series, it of the second layer makes prediction using the time window obtained from the lower (first) layer. For maintaining diversity, LEA uses a different training set for each network in the ensemble of the first and second layers. LEA has been tested extensively on the time series data sets of NN3 competition. In terms of prediction accuracy, our experimental results have showed clearly that LEA is better than other ensemble and nonensemble algorithms.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

A time series is a sequence of observations of the same random variable at different times, normally at uniform intervals. Share prices, profits, imports, exports, interest rates, popularity ratings of politicians and amount of pollutants in the environment are some examples of time series. Fig.1.1 gives an idea about a time series where the exchange rate between the British pound and the US dollar are recorded over a specific number of weeks. Here exchange rate is the rate at which one currency will be exchanged for another. Lagged variables, autocorrelation and non-stationarity are the major characteristics that distinguish time series data from other types of data. The difficulties posed by these special features make forecasting time series extremely difficult.

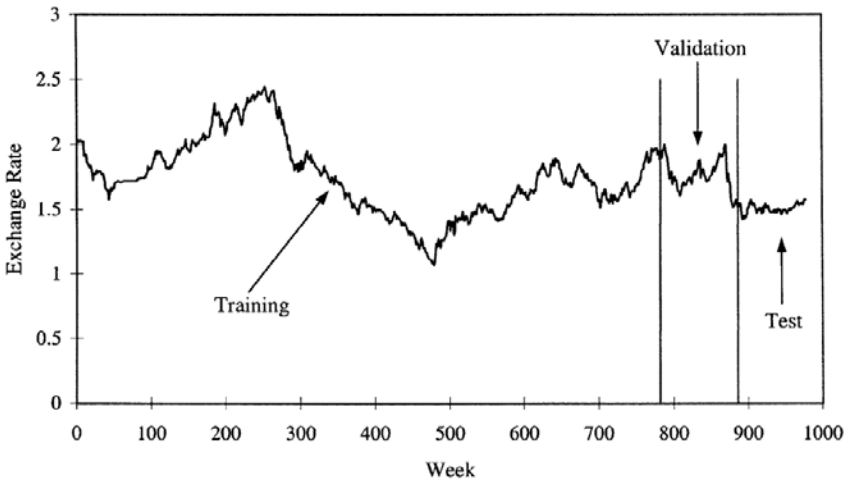

Figure 1.1: British pound vs US dollar exchange rate

Time series forecasting (TSF) is to predict future values based on previously observed values

using a model or technique. TSF has been widely used in many real-world applications such as financial market prediction [5], weather prediction [6], electric load forecasting [7] etc. Generally the characteristics of the phenomenon generating a time series are unknown and information available for forecasting is limited to the past values of the series. The relations between the past and the future values should be deduced in the form of functional relation approximations. It is thus important to use an appropriate number of past values, termed *time window* or *lag*, for forecasting. Using a time window of a fixed size has proven to be limiting in many applications: if the time window is too narrow, important information may be left out while, if the window is too wide, useless inputs may cause distracting noise. Ideally, for a given problem, the size of the time window should be adapted to the context.

Over the last few decades, there have been immense interests for understanding the past and predicting the future. This gives us many forecasting methods; most of them are relying on linear and non-liner statistical models. Linear statistical models are easy to explain and implement, but they may be inappropriate for time series originating from non-linear processes. Although nonlinear statistical models [8], [9] overcome some of the limitations of the linear models, they are still limited in a way to solve real-world problems [10]. This is because the time series data has to be hypothesized with little knowledge of the underlying law that generates the series. Furthermore, the formulation of a non-linear statistical model for the series is a difficult task [11]. Most of the statistical methods involve the construction of a mathematical model that best represents the behavior of the observed system. However, identifying a suitable model requires skilled and experienced forecasters as real-world processes often exhibit non-linear characteristics which are difficult to model. Different forecasters may arrive at different models even if they use the same set of observations. For example, in ocean modeling and weather prediction, individual forecasters can be sensitive to small variations in initial and boundary conditions [12].

Several studies try to employ machine learning techniques to avoid the aforementioned shortcomings. Ensemble of multiple learning machines that brings together several learning machines to provide a single output has received a lot of research interests in the machine learning community. Hansen and Salamon (1990) in their seminal work show that the generalization ability of neural networks (NNs), a machine learning technique, can be significantly improved through ensembling. Furthermore, an ensemble alleviates the difficulty associated with the conventional design strategy of selecting a single network and its parameters. In [13], it is explained that NN models are inherently unstable in perfor-

mance, i.e., small changes in the training set and/or parameter selection can produce large changes in the network architectures. Thus, the sensitivity to small perturbation in model parameters can be exploited to generate an ensemble of forecasters, and the output from all ensemble members can be analyzed and combined to improve the overall performance.

1.1 Our contribution

NN ensembles have been widely used for classification problems, but there is only a handful of work for TSF problems [14], [15]. The main issue of applying ensemble approaches is the consideration of "accuracy" and "diversity" (chapter 2) among individual networks, called base predictors, in an ensemble. Existing ensemble approaches used for TSF try to induce diversity in the base predictors, but they totally ignore the other important component i.e., accuracy of the predictors. Specifically, they use the same and fixed time window for different TSF problems. As mentioned earlier, the time window is the most critical component in forecasting, because it corresponds to the number of past observations used to capture the underlying pattern of a time series. The use of many or small number of past examples is not beneficial for learning and may affect the accuracy of base predictors.

Besides, a common problem with existing the time series forecasting model is the low accuracy of long term forecasts. The estimated value of a variable may be reasonably reliable in the short term, but for longer term forecasts, the estimate is likely to become less accurate. While reliable multi-step ahead time series prediction has many important applications and is often the intended outcome, most of the existing works usually considers one-step time series prediction. The main reason for this is the increased difficulty of the multi-step ahead TSF problems and the fact that the results obtained by simple extensions of techniques developed for one-step prediction are often disappointing.

The main contribution of this thesis is to devise a method to work well on both one-step and multi-step prediction. To achieve this goal, we propose a layer based ensemble architecture (LEA) for TSF. Our LEA is consisted of two layers. We use an ensemble of multi-layer perceptron (MLP) networks for each of the layers. The salient features of LEA are as follows.

- LEA considers both accuracy and diversity in constructing layered ensemble architectures for TSF.
- Due to use of ensemble for TSF, LEA is less sensitive to the system parameters, such as the architecture of NNs, the initial weights of NNs and learning rate for training NNs.

- LEA does not put any constrain on the type of networks to be used for constructing ensembles.
- LEA can be applied to a wide range of time series prediction problem.

1.2 Outline of the thesis

The reminder of this thesis is organized as follows.

In chapter 2 we briefly discuss about the basic components necessary to understand the idea presented in this thesis. We start with the discussion of the architecture and learning algorithm for training the NNs. Then we briefly discuss about how we can map a time series data into the conventional data format to train the NNs. The architecture and the working principle of ensemble of NNs are then introduced. Then we present some arguments to justify why ensemble of NNs should be used to improve the performance. Finally, we provide a general overview of the existing state of the art algorithms using ensemble for TSF.

Chapter 3 presents the major contribution of this thesis. Our LEA algorithm and main components of LEA are thoroughly discussed with their specific purposes. The mechanism of inducing both diversity and accuracy in ensemble of NNs is also presented here. A comparative discussion illustrating how LEA is different from the existing state of the art algorithms concludes this chapter.

In chapter 4, the experimental analysis regarding the performance of LEA is presented. The description of the dataset used in this thesis are presented at first. It is followed by the discussion of the various performance metrics and their formulation. Since, LEA uses basic bagging algorithm, so we provide an extensive simulation and result analysis against basic bagging algorithm. However, we also compare the performance of LEA with boosting. In order to establish the fact that LEA provides better result than basic ensemble algorithm we also perform statistical analysis here.

Finally, in chapter 5 we provide some future aspects of this research and conclude the thesis. We try to provide some directions regarding how the presented LEA algorithm can be modified and extended further.

Chapter 2

Background

Neural networks (NNs) are a class of machine learning algorithms that draw inspiration from biological neural systems. They are generally implemented in computer software with the aim of enabling automatic learning and subsequently autonomous problem solving.

This chapter begins with introduction to NNs followed by an explanation of the architecture and the working principle of ensemble of NNs. To clarify the idea of ensemble further, three most popular ensemble algorithms i.e. bagging [16], boosting [17] and random subspace [18] are presented here along with their pros and cons. Next, we provide a brief comment to justify why ensemble of NNs should be used to improve the performance and how NNs can be applied to time series forecasting. Finally, we conclude this chapter by describing some state of the art ensemble approaches for TSF.

2.1 Neural networks

NNs are computational models implemented in computer systems in an attempt to replicate some of the behavioral and adaptive features of biological neural systems. The biological nervous system consists of assemblies of interconnected cells called neurons. Each neuron has a certain transfer threshold that the sum of the incoming voltages (from the dendrites) need to exceed in order to activate an outgoing electrical signal. When this occurs, we say the neuron is excited, or that it is firing. NNs are by no means exact replicas of their biological counterparts. In the following, we are going to discuss about how these replicas are made.

2.1.1 Artificial neurons

Every NN consists of a set of units (or neurons) and a set of connections between them. Each neuron is basically just a mathematical function ϕ (the transfer function) that takes as parameter the transfer a, which is a weighted sum of all the incoming signals to the neuron. The value of $\phi(a)$ is the outgoing signal of the neuron.

It is important to note that the transfer parameter of a given neuron is a weighted sum of all its incoming signals:

$$a = \sum w_i \times x_i \tag{2.1}$$

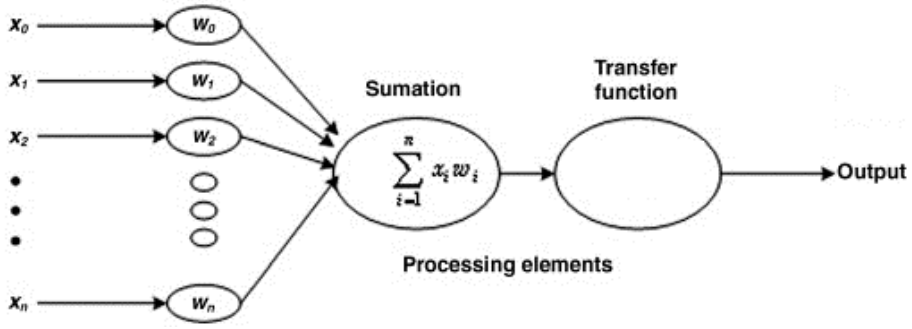

Figure 2.1: Artificial neuron model

In fig. 2.1 w_i is the weight of the incoming connection i, and x_i is the signal value that is sent by the neuron on the other side of that connection. It's clear that the higher the weight of the connection is, the more influence it will have on the neuron. Here we can simulate adaptation by adjusting the values of these weights. This is typically done using a method called *backpropagation*, which we will come back to later.

The exact nature of the transfer function $\phi(a)$ can be defined in several different ways. One very simple and somewhat common approach is to use a step function which is either 1 or 0 based on whether the transfer a is greater than some constant threshold v, *i.e.* :

$$\phi(a) = \begin{cases} 1, & \text{if } a \geq v \\ 0, & \text{otherwise} \end{cases}$$

While this approach works well enough in many situations, it is clear that more information could

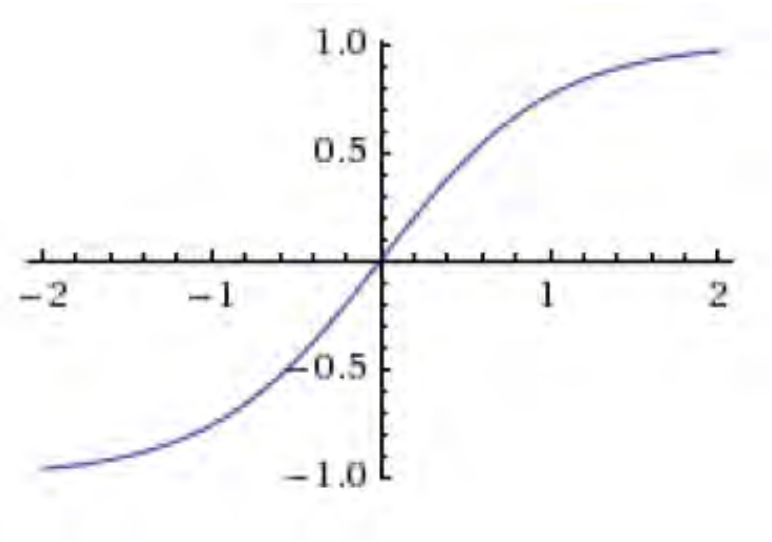

Figure 2.2: The symmetric sigmoid transfer function

be produced by each neuron if its transfer function was continuous instead of just binary. This is because a binary ϕ means the neuron can only take on one of two states, whereas a continuous ϕ can take on any number of different values. One of the most popular choices of continuous transfer functions, which is also the one used by our proposed method, is the symmetric sigmoid function:

$$\phi(a) = \tanh(k \times a) \tag{2.3}$$

Here k is a scaling factor which determines how steep the curve is. The resulting value is bound to the range $\langle -1; +1 \rangle$. Fig. 2.2 shows the shape of this function with $k = 1$.

2.1.2 Layer architecture

Now that we understand the basic mechanism of how these artificial neurons operate individually, we can next consider how a network of them operates. The standard way of designing NNs is to group the neurons into N layers, including one input layer, one output layer, and one or more hidden layer. Such a network is illustrated in fig. 2.3. Notice that in this network, a given neuron in one layer is not necessarily connected to all the neurons in the next. This is what we call a sparse network. A complete network is one in which any given neuron is always connected to every neuron in the next layer.

- **Input layer:** This layer can be thought of as the "sensor organ" of the NN. It is where we set

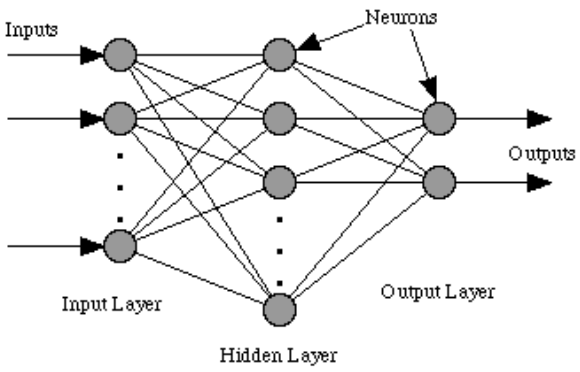

Figure 2.3: Example of an artificial neural network with layers

the parameters of the environment (i.e. the information we want the NN to make a decision about). The neurons in this layer have no incoming connections, since their values are set from an external source. The outgoing connections send these values to the neurons of the next layer in the hierarchy.

- **Hidden layer:** In between the input and output layers, we put a series of one or more "hidden" layers. The reason we call them hidden is that they are invisible to any external processes that interact with the NN. The neurons in these layers have both incoming connections from the preceding layer and outgoing connections to the succeeding layer, and work just as described earlier in this section. The hidden layers can be thought of as the "cognitive brain" of the network.

- **Output layer:** The output layer holds the end result of the computations of the NN. If the input layer holds the parameters of a problem, the information here can be interpreted as the proposed solution. The neurons in this layer have no outgoing connections, because their ϕ-values are read directly by whatever external process is using the network.

To test the network, we simply load the problem information into the input layer neurons, and compute ϕ for every neuron in each of the succeeding layers (layer by layer until we reach the output layer). The resulting values in the output layer will depend on what training we have previously exposed the network to.

2.1.3 Training the NN

Our discussion of NNs so far has explained what they do (i.e. what calculations are made) when they are given some input that is transformed into some output (as the neurons are updated through the network from the input to the output layer). In this section we will see how we can attempt to teach the network to solve specific problems by showing it examples of problems with given correct solutions.

We already mentioned that by varying the weights of the connections in the network we can train the network. This is typically done by implementing a process called "backpropagation of error". Qualitatively, the process can be described roughly as follows:

- Load an example problem from the training data.
- Run the network normally with the problem information.
- Calculate the error between the resulting output of the NN and the actual correct solution.
- Iterate backwards through the layers of the network, and slightly tweak the weights of all the connections in the direction (positive or negative) that minimizes the error of the output.

This process is repeated over a set of training data. The "tweaking" of the weights is done using a formula called the Delta rule, which is based on the principle of gradient descent.

2.2 Ensemble of neural networks

An ensemble of learning machines is using a set of learning machines (i.e. neural network discussed in previous section) to learn partial solutions to a given problem and then integrating these solutions in some manner to construct a final or complete solution to the original problem. Using $\hat{y}_1; \ldots; \hat{y}_m$ to denote m individual learning machines, a common example of ensemble for regression problem is shown in fig. 2.4 where $w_i > 0$ is the weight of the estimator $\hat{y}_i$ in the ensemble and model can be any learning machine (i.e. neural network).

Ensemble methods have been widely used to improve the generalization performance of the single learner. This technique originates from Hansen and Salamon's work [19], which showed that the generalization ability of a neural network can be significantly improved through ensembling a number of neural networks.

Based on the advantages of ensemble methods and increasing complexity of real-world problems, ensemble of learning machines is one of the important problem-solving techniques. Since the last decade, there have been much literature published on ensemble learning algorithms, from Mixtures of Experts [20], bagging [16] to various boosting [17], random subspace [21], random forests [18] and negative correlation learning [22], etc.

Before going into further details about ensemble we need to be familiar with two most common terminologies used in the ensemble literature. These are:

- **Diversity.** Brown et al. [23] gives a good account of why diversity is necessary in neural network ensembles and presents a taxonomy of methods that enforce it in practice. If two neural networks make different errors on the same data points/inputs, they are said to be diverse [23].
- **Accuracy.** Accuracy could be defined as the degree of a network (ensemble member) performing better than random guessing on a new input [23].

2.2.1 Creating an ensemble

Neural network ensemble, which originates from Hansen and Salamon's work [19], is a learning paradigm where a collection of neural networks is trained for the same task. Two interrelated questions need to be answered in designing an ensemble system: i) how will individual classifiers (base classifiers) be generated? and ii) how will they differ from each other? The answers ultimately determine the diversity of the classifiers, and hence affect the performance of the overall system. Therefore, any strategy for generating the ensemble members must seek to improve the ensembles diversity and

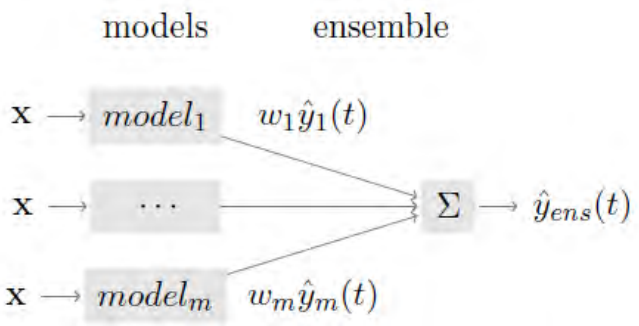

Figure 2.4: A schematic overview of how models can be combined in an ensemble

accuracy. In general, however, ensemble algorithms do not attempt to maximize a specific diversity and accuracy measure. Rather, increased diversity is usually sought somewhat heuristically through various resampling procedures or selection of different training parameters. Bagging (section 2.2.2), boosting (section 2.2.4), and random subspace method (section 2.2.5) are three most popular ensemble algorithms based on the various resampling procedures of the training dataset. In the following section, we are going to give a brief overview about these three algorithms.

2.2.2 Bagging

The bagging technique (bootstrap aggregating) [16] is based on the idea that bootstrap samples of the original training set will present a small change with respect to the original training set, but sufficient difference to produce diverse predictors. Each member of the ensemble is trained using a different training set, and the predictions are combined by averaging or voting. The different datasets are generated by sampling from the original set, choosing N items uniformly at random with replacement. See the basic method in Algorithm 2.1.

Algorithm 2.1 Bagging

Require: Ensemble size L, training set S of size N.

```
for i = 1 to L do
    S_i ← N sampled items from S, with replacement.
    Train predictor h_i using S_i.
end for
for each new pattern do
    if outputs are continuous then
        Average the decisions of h_i, i = 1; ...; L.
    else
        if outputs are class labels then
            Compute the majority voting of h_i, i = 1; ...; L.
        end if
    end if
end for
```

The probability of any object not being selected is $p = (1 - 1/N)^N$. For a large N, a single bootstrap (version of the training set) is expected to contain around 63% of the original set, while 37% are not selected.

There are two interesting points of bagging method:

- Instances of an unstable predictor trained using different bootstraps can show significant differ-

ences,

- Variance reduction.

The bias-variance decomposition is often used to handle discussions on ensemble performances. Bias-variance decomposition analyzes how much selection of any specific training set affects performance of ensemble. In this decomposition, it is assumed that there are infinitely many classifiers built from different training sets. Then bias is defined as expected error of the combined classifiers on new data (e.g. test data) whereas variance is the expected error due to the particular training set used. So, when the classifiers has small bias (errors) but high variance, bagging helps to reduce the variance [24]. According to Breiman [16] it works better with unstable models such as decision trees and neural networks, and can possible not work well with simple or stable models such as k-Nearest Neighbors.

After creating the ensemble we can use any combination method. However, according to [24] simple linear combination using averaging or majority voting (elementary combiners, see section 2.2.6) is optimal.

The number of bagged predictors should be selected by experimenting for each application. In the literature it is common to use from 50 to 100 predictors, but when an applications cannot handle this amount of processing time, this number must be better chosen. A recent study showed that with 10 predictors the error reduction is around 90%, because bagging allow a fast variance reduction as the number of bagged predictors is increased. Curiously, despite the simplicity and many successful applications, there are situations where bagging converges without affecting variance [23], so it seems that this method is still not fully understood.

2.2.3 Boosting

In 1990, Schapire proved that a weak learner, an algorithm that generates classifiers that can merely do better than random guessing, can be turned into a strong learner that generates a classifier that can correctly classify all but an arbitrarily small fraction of the instances [17]. Formal definitions of weak and strong learner, as defined in the PAC learning frame work, can be found in [17], where Schapire also provides an elegant algorithm for boosting the performance of a weak learner to the level of a strong one. Hence called boosting, the algorithm is now considered as one of the most important developments in the recent history of machine learning.

Similar to bagging, boosting also creates an ensemble of classifiers by resampling the data, which are then combined by majority voting. However, similarities end there. In boosting, resampling is strategically geared to provide the most informative training data for each consecutive classifier. In essence, boosting creates three weak classifiers: the first classifier C_1 is trained with a random sub-set of the available training data. The training data subset for the second classifier C_2 is chosen as the most informative subset, given C_1. That is, C_2 is trained on a training data only half of which is correctly classified by C_1, and the other half is misclassified. The third classifier C_3 is trained with instances on which C_1 and C_2 disagree. The three classifiers are combined through a majority vote.

Schapire has shown that the error of this three classifier ensemble is bounded above, and it is less than the error of the best classifier in the ensemble, provided that each classifier has an error rate that is less than 0.5. For a two-class problem, an error rate of 0.5 is the least we can expect from a classifier, as an error of 0.5 amounts to ran-dom guessing. Hence, a stronger classifier is generated from three weaker classifiers. A strong classifier in the strict PAC learning sense can then be created by recursive applications of boosting.

2.2.4 Adaboost

In 1997, Freund and Schapire introduced AdaBoost [25], which has since enjoyed a remarkable attention. AdaBoost is a more general version of the original boosting algorithm and capable of handling multiclass and regression problems, respectively. Adaboost (adaptive boosting) [25], tries to combine weak base predictor in order to produce an accurate "strong" predictor. There are many variations and derived methods of boosting in the literature. It is similar to bagging since the base predictors are built over different training sets. However it is a ensemble learning method, not a general methodology to construct an ensemble.

The method is an iterative process that builds an ensemble of base predictors. The algorithm trains base predictors sequentially, a new model per round. At the end of each round, the misclassified patterns are weighted in order to be considered more important in the next round, so that the subsequent models compensate error made by earlier predictors. The learning algorithm of the predictor used in Adaboost must allow the use of a weight for each training pattern.

Algorithm 2.2 shows the procedure of boosting. Note that bagging uses uniform distribution for S_i, while Adaboost adapts a non-uniform one W_i over the elements of S. It also checks if the current predictor has at least a performance better than random guessing, that is, the error $\epsilon_i < 0.5$. When

this condition is not true, another predictor is trained or the iteration is terminated. After each iteration, the distribution W_i is updated so that the samples wrongly estimated by h_i have the half of the distribution mass. For example, if the error is 0.15, the next predictor will devote 50% of effort in order to estimate correctly the examples from that 15% wrongly estimated samples, while the other objects are less emphasized.

Algorithm 2.2 Adaptive Boosting

Require: Ensemble size L, training set S of size N where $y_i \in \{+1, -1\}$, initialize uniform distribution W_i over S.

1: **for** $i = 1$ to L **do**
2: Train predictor h_i using distribution W_i.
3: Compute $\epsilon_i \leftarrow P_{Wi}(h_i(x) \neq y)$.
4: **if** $\epsilon_i \geq 0.5$ **then**
5: Break.
6: **end if**
7: $\alpha_i \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_i}{\epsilon_i}$
8: $W_{i+1} \leftarrow \frac{W_i \exp(-\alpha_i y_i h_i(x))}{Z_i}$
9: where Z_i is a normalization factor to ensure that W_{i+1} is a valid distribution.
10: **end for**
11: **for** each new pattern x **do**
12: $H(x) \leftarrow \text{sign}(\sum_{i=1}^{L} \alpha_i h_i(x))$
13: **end for**

2.2.5 Random Subspace Method

The Random Subspace Method (RSM) [18] randomly selects an arbitrary number of subspace from the original feature space, and build a predictor on each subspace. This randomization should create predictors that are complementary. The combination can be carried out by simple fixed rules. Experimental evidences showed that RSM works well with feature spaces with large feature sets and redundant features. It avoids the curse of dimensionality.

The algorithm projects each feature vector into a less dimensional subspace, by selecting m random components. Algorithm 2.3 shows the procedure.

RSM is similar to bagging but instead of sampling objects, it performs a kind of feature sampling without replacement since it would be useless to include one feature more than once. Note that the number of features that is used to train each predictor, d_i can be different from one predictor to another, although it is often set to be uniform.

This method seems to work well for large feature sets with redundant features. It was also observed

that it avoids the curse of dimensionality [26]. The key issue is how to define the dimensionality of the subspaces (variable d). For this there is no clear guideline, for each application experiments have to be carried out to understand the effect of d on the performance.

Algorithm 2.3 Random Subspace Method

Require: Ensemble size L, feature set S of size N, where the number of features is D; choose d_i to be the number of features to train each individual predictor, where $d_i < D$, for $i = 1; \ldots; L$.

for $i = 1$ to L **do**
    $S_i \leftarrow d$ randomly chosen features out of D, without replacement.
    Train predictor h_i using S_i.
end for
for each new pattern **do**
    if outputs are continuous **then**
        Average the decisions of h_i, $i = 1; \ldots; L$.
    else
        if outputs are class labels **then**
            Compute the majority voting of h_i, $i = 1; \ldots; L$.
        end if
    end if
end for

2.2.6 Combining the members of ensemble

Another key component of any ensemble system is the strategy employed in combining classifiers or predictors. Combination methods try to find out an optimal way to combine the output of the predictors in the ensemble so that the ensemble exhibits lower error on the test set than any member in the ensemble.

Several reasons of combining the predictors are summarized by Timmerman [27]. First argument is due to diversification. One model is often suited to one kind of data. Thus, the higher degree of overlap in the information set, the less useful a combination of forecasts is likely to be. In addition, individual forecasts may be very differently affected by structural breaks in time series. Another related reason is that individual forecasting models may be subject to misspecification bias of unknown form. Lastly, the argument for combination of forecasts is that the underlying forecasts may be based on different loss functions. A forecast model with a more symmetric loss function could find a combination of the two forecasts better than the individual ones.

The forecast combination problem generally seeks an aggregator that reduces the information in a potentially high-dimensional vector of forecasts to a lower dimensional summary measure. Poncela

et al. [28] denotes that one point forecast combination is to produce a single combined 1-step-ahead forecast $\hat{Y_t}$ at time t, with information up to time t, from the N initial forecasts; that is

$$\hat{Y_t} = w_t \times Y_{t+1|t} \tag{2.4}$$

where w_t is the weighting vector of size N for the combined forecast, $Y_{t+1|t}$ is N dimensional vector of forecasts at time t. A constant could also be added to the previous combining scheme to correct for a possible bias in the combined forecast. The main aim is to reduce the dimension of the problem from N forecasts to just a single one, $\hat{Y_t}$.

After creating an ensemble, there are numerous methods for combining predictors. The final decision can be obtained by two principal approaches:

- **Integration (or fusion):** all predictors contribute to the final decision, assuming competitive predictors.
- **Selection:** one predictor is used to give the final decision. It assumes that predictors are complementary.

Integration of members

Various integration methods may be applied in practice. In this thesis, we will compare methods based on the averaging, both simple and weighted on predictors performance. Some widely used averaging techniques are briefly described below:

- **Simple average:** In the simple averaging schema, the final forecast is defined as the average of the results produced by all different predictors. The simplest one is the ordinary mean of the partial results. The final prediction $\hat{Y_t}$ from N predictors is defined by:

$$\hat{Y_t} = 1/N \sum_{i=1}^{N} \hat{Y_{t_i}} \tag{2.5}$$

  where $\hat{Y_{t_i}}$ is the prediction made by the individual predictor. This process of averaging may reduce the final error of forecasting if all predictive networks are of comparable accuracy. Otherwise, weighted averaging shall be used.

- **Weighted average:** The accuracy of weighted averaging method can be measured on the basis of particular predictor performance on the data from the past. The most reliable predictor should be considered with the highest weight, and the least accurate one with the least weight. The estimated prediction is calculated as

$$\hat{Y_t} = 1/N \sum_{i=1}^{N} w_i \times \hat{Y_{t_i}} \tag{2.6}$$

where $\hat{Y_{t_i}}$ is the prediction made by the individual predictor and w_i is weight associated with each predictor. One way to determine the values of the weights ($i = 1, 2, \ldots, N$) is to solve the set of linear equations corresponding to the learning data, for example, by using ordinary least squares. Another way is using relative performance(i.e. mean scaled error(MSE)) of each predictor [27], where the weight is specified by:

$$w_i = \frac{1/MSE_i}{\sum_{i=1}^{N} 1/MSE_i} \tag{2.7}$$

In this weighted average, the high performance predictor will be given larger weight and vice versa.

- In the **trimmed average**, individual forecasts are combined by a simple arithmetic mean, excluding the worst performing k% of the models. A trimming of 10% to 30% is usually recommended [29], [30].

- In the **median-based combining**, the combination function is the median of the individual forecasts. Median is sometimes preferred over simple average as it is less sensitive to extreme values [31], [32].

- In the **variance-based method**, the optimal weights are determined through the minimization of the total sum of squared error (SSE) [30], [33].

Selection of members

Franses [34] states that the prediction methods that need to be combined are those which contribute significantly to the increased accuracy of prediction. The selection of prediction models in the ensemble is usually done by calculating the performance of each model toward the hold-out sample. Andrawis

et al. [35] use 9 best models out of 140 models to combine. Previously, Armstrong [36] states that only five or six best models are needed to get better prediction result. Authors in [37] also suggests that selecting few best models are crucial for improving the forecasting result. Some selection mechanisms are described here in brief.

- **Winner takes all:** In this schema, the final output of an ensemble system is the output of a member of that ensemble which makes the lowest error on the hold-out sample. For error measure any performance metric can be used (i.e. *MSE*, *sMAPE*).
- **Diversity based selection:** A set of diverse member of an ensemble is selected to give the final output of the ensemble based on some diversity measuring mechanisms. The author in [38] selects the most diverse members (vary the number of diverse member from 5 to 20) from a random forest ensemble and bagged ensemble for regression problem.
- Kuncheva presents an hybrid approach between the selection of best predictor and the combination of the predictors [39]. It uses paired t-hypothesis test to verify if there is one predictor meaningfully better than the others. If positive, it uses the best predictor, if not it uses a combination approach.

2.3 Reasons for better performance of ensemble

Theorems have shown that there is not a single predictor that can be considered optimal for all problems [40]. There is no clear guideline to choose a set of learning methods and it is rare when one has a complete knowledge about data distribution and also about the details of how the prediction algorithm behaves. Therefore, in practical pattern prediction tasks it is difficult to find a good single predictor.

The choice of a single predictor trained with a limited (size or quality) dataset can make the design even more difficult. In this case, selecting the best current predictor can led to the choice of the worst predictor for future data. Especially when the data used to learn was not sufficiently representative in order to estimate properly new objects, the test set provides just apparent errors $\hat{E}$ that differ from true errors E, in a generalization error: $\hat{E} = E + \delta$. This common situation, where small and not representative data is used as an input to a predictor, can led to difficulties when one must choose from a set of possible methods.

According to Dietterich [41], there are three main motivations to combine predictors, the worst case, the best case and the computational motivation:

- **Statistical (or worst case) motivation:** It is possible to avoid the worst predictor by averaging several predictors. It was confirmed theoretically by Fumera and Roli in [24]. This simple combination was demonstrated to be efficient in many applications. There is no guarantee, however, the combination will perform better than the best predictor.

- **Representational (or best case) motivation:** Under particular situations, fusion of multiple predictors can improve the performance of the best individual predictor. It happens when the optimal predictor for a problem is outside of the considered "predictor space". There are many experimental evidences that it is possible if the predictors in an ensemble makes different errors. This assumption has a theoretical support in some cases when linear combination is performed.

- **Computational motivation:** Some algorithms performs an optimization task in order to learn and suffer from local minima. Algorithms such as the backpropagation for neural networks are initialized randomly in order to avoid locally optimum solutions. In this case it is a difficult task to find the best predictor, and it is often used several (hundreds or even thousands) initializations in order to find a presumable optimal predictor. Combination of such predictors showed to stabilize and improve the best single predictor result [42].

2.4 Time series using NN

The problem of forecasting time series with NN is considered as obtaining the relationship from the value at period "t" (in this system, the resulting NN will have only one output neuron) and the values from previous elements of the time series $(Y_{t-1}, Y_{t-2}, \ldots, Y_{t-k})$ to obtain a function as it is shown in Eq. 2.8.

$$Y_t = f(Y_{t-1}, Y_{t-2}, \ldots, Y_{t-k}) \tag{2.8}$$

To obtain a NN for making a single step forecast, one have to go through the following steps:

- **Normalization of data:** An initial step has to be done with the original values of the time series, i.e. normalizing the data. The original values of the time series are normalized into the

range [-1,1]. Once the NN gives the resulting values, the inverse process is carried out, rescaling them back to the original scale.

- **Number of node in input layer:** Defining the number of node in input layer is the most important things because it corresponds to the number of past lagged observation which will affect the future point. So basically the number of input nodes for TSF problem is equal to the lag of the time series.

- **Number of node in hidden layer:** The number of hidden nodes, empowers neural networks with the nonlinear modeling capability. It has been shown that the in-sample forecasting and the out-of-sample forecasting ability of neural networks are not very sensitive to the number of hidden nodes [43]. Therefore, for the sake of simplicity, number of hidden nodes is kept same as the number of input nodes.

- **Number of node in output layer:** For single-step ahead forecast, the number of node in the output layer is always will be one. But for multi-step ahead forecast, depending upon the method of forecasting, the number of node in this layer may vary.

- **Training set generation:** To obtain a training set, the time series will be transformed into a pattern set depending on the k input nodes of a particular NN, and each pattern will consist of the following:

  - k inputs values that correspond to k previous values of period $t : Y_{t-1}, Y_{t-2}, , Y_{t-k}$.
  - One output value: Y_t (the desired target).

  This patterns set will be used to train and validate each NN of the ensemble. Therefore, patterns set will be split into two subsets, training and validation. The complete patterns set is ordered into the same way the time series is. The first x% (where x is a parameter) from the total patterns set will generate the train patterns subset, and the validation subset will be obtained from the rest of the total patterns set. The test subset will be the future (and unknown) time series values that the user wants to forecast. An example of this process using an NN with 3 input nodes ($k = 3$) can be seen at fig. 2.5.

- **Activation functions:** Regarding to the transfer function, transformation of time series data is needed. Depending upon which data normalization scheme we use, there are several transfer

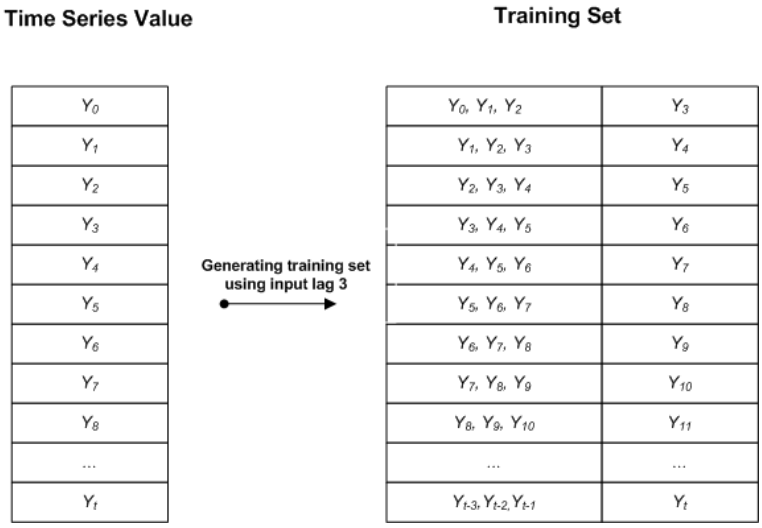

Figure 2.5: Process to generate training set from time series data

function. Like sigmoid transfer function works on the range [0,1] whereas hyperbolic tangent function uses the range [-1,1]. For the hidden layer, the transfer function always should be either sigmoid logistic or hyperbolic tangent [44]. But for output layer normally linear transformation function is used [44].

- **Learning algorithm:** There are several types of training algorithms in the literature used for learning of feed forward artificial neural networks. One of the widely used training algorithm for TSF is Levenberg-Marquardt (LM) [44]. However, back propagation or other gradient descent based learning algorithms can also be used.

2.5 Related work

Very few works have been conducted to deal with the time series forecasting problem using ensemble. Most of these works fall into one of the four different categories: TSF using bagging ensemble approaches, TSF using boosting ensemble approaches, TSF using hybrid ensemble approaches and TSF using miscellaneous ensemble. Here, we provide a brief overview of all these types of approaches.

2.6 Existing state of the art ensemble algorithms for TSF

For classification problems there exists many ensembles algorithms in the literature. However, the use of ensembles for TSF problems is very few. The main issue of applying ensemble approaches for TSF (or classification) is the consideration of accuracy and diversity among base predictors (or predictors) in an ensemble. Combining several predictors to improve the forecasting accuracy has been extensively studied in the traditional forecasting literature. Clemen [45] provides a comprehensive review and annotated bibliography in this area. The aim of this section is to describe computational intelligence based ensemble approaches used for TSF problems.

There are two different ways in inducing diversity: one is varying the parameters involved with the base predictors and one is varying data sets used for constructing/training the base predictors. Boosting [17] and bagging [16] are popular methods for creating data sets with some degree of variations among them. Random subspace method [18] is also like bagging but the variation is applied on the feature set instead of dataset.

2.6.1 TSF using Boosting ensemble

Boosting is used in [46] where feed-forward neural networks are employed as base predictors for forecasting. The algorithm is based on a fundamental observation that often the mean squared error of a predictor is significantly greater than the squared median of the error due to a small number of large errors. Boosting is also coupled with recurrent neural networks in several studies. The authors in [47] first create an ensemble by randomizing the initial weights of a pool of Elman networks, a kind of recurrent networks and then use the modified AdaBoost [48] by directly weighting the cost function of the networks. Assaad *et al.* [49] introduce a new parameter in boosting for creating an ensemble of recurrent networks. The new parameter tunes the boosting influence on available examples.

Instead of using neural networks as base predictors, other machine learning techniques can also be used. Genetic programming (GP) [50]) is one such technique. Paris *et al.* [51] propose GPBoost that uses boosting algorithm with GP as a base predictor. A very similar idea is used in [52], where the correlation coefficients are used to update the weights and the final hypothesis instead of the loss function commonly used by boosting. The correlation coefficient measures the relation between the real and the predicted values of a given time series data.

Boosting algorithm is used in [46] for the prediction of a benchmark time series, but with MLPs

as models.

2.6.2 TSF using Bagging ensemble

Like boosting, the application of bagging is also found in the forecasting literature. The authors in [53] use bagging with neural networks for binary prediction of financial time series. Inoue and Kilian (2005) explore the usefulness of bagging in forecasting economic time-series from the perspective of linear multiple regression models. It has been observed that bagging tends to produce large reductions in the out-of-sample error and provides a useful forecasting tool. Bagging with competitive associative networks (CANs) is applied to time series data of NN3 competition [54]. To remove seasonality, time series data are first preprocessed by employing first-order difference and then bagging is applied to create different training sets for different CANs in an ensemble.

2.6.3 TSF using Random Subspace

The authors in [14] employ 500 classification and regression (CART) as base predictor to constitute the ensemble. This paper presents an ensemble algorithm for time series forecasting that combines the bootstrap sampling and random subspace. To train each base CART 63.2% of data is randomly selected; within the selected data 70% of feature is randomly selected. To generate training data from time series the authors employ lag number as a training parameter. A simple model selection based on minimizing the in sample Symmetric Mean Absolute Error ($sMAPE$) Eq. (3.1) is employed to select the lag parameter. Finally, the model that minimizes the $sMAPE$ is selected to make the final forecast. The proposed ensemble method participated in the NN3 competition [1] and secured the third position among the computational intelligence methods.

2.6.4 TSF using different base predictors ensemble

All the ensemble approaches discussed so far combine base predictors taken from one single class. However, it is possible to construct ensembles by combining base predictors taken from several classes. Wichard and Maciej [55] combine predictors taken from linear and polynomial models, k-nearest neighbor model, neural networks and perceptron radial basis network. Unlike [55], the authors in [40] use radial basis function networks, k-nearest neighbors and self organizing maps for constructing ensembles. Ten different predictors for each of the three classes are first generated by varying the

number of hidden layers for neural networks, the number of neighbors for k-nearest neighbors or the map size of self-organizing maps. Three best predictors one from each class are then combined to generate the ensemble output.

2.6.5 TSF using other ensemble approach

Apart from the above discussed methods, there exists several other ensemble method for TSF. For example, the single layer nonlinear neural network ensemble model proposed in [56] used principle component analysis to determine the ensemble members. Here [56] a triple-phase nonlinear neural network ensemble model is proposed for financial time series forecasting. First of all, many individual neural predictors are generated. Then an appropriate number of neural predictors are selected from the considerable number of candidate predictors. Finally, selected neural predictors are combined into an aggregated neural predictor in a nonlinear way. In 2007, Bo Qian and Khaled Rasheed [57] examined Dow Jones Industrial Index using voting and stacking ensemble methods of back propagation neural networks, k-nearest neighbor and decision trees.

Brown *et al.* [23] paper investigates the performance of two different types of neural network ensembles. One type varies the initial weights for the individual networks an ensemble, while the other type varies the architecture of the individual networks. The parameter of recurrent neural networks is varied in constructing ensembles using such networks. For example, in [58], an ensemble approach using "echo state" neural networks, a special case of recurrent neural networks, with different memory length is proposed. A special gating echo state neural network is used to combine the networks.

Recently, a multi-level ensemble architecture consisting of m neural networks is proposed in [59]. Each network has a different randomly selected architecture with respect to hidden layers. In the first level, there are k groups of networks having on average m/k networks in each group. Each group of networks is trained on a different subset of data obtained by randomly permutating, partitioning and injecting noise to the original data. The best networks from each of the k groups are combined in the second level to form the ensemble output. In [2], ensemble is first used to simultaneously determine the optimal spreads of radial basis function networks and the optimal lag of a given time series. These optimal values are then used to train the networks for forming an ensemble. In both level, only three radial basis function networks are used and the same data set is used for training all networks in the ensemble.

In [2], an ensemble architecture consisting of three radial basis function (RBF) networks is proposed

and applied to the time series data of NN3 competition. The author in [2] suggests to use the spread factor set of RBF networks as the 50th, 75th, and 95th percentiles of the nearest distances of all training samples to the rest of the points. It is, however, not known whether this suggestion is effective for other time series data. To find an appropriate lag of a given time series, the ensemble architecture is trained l_{max} times using the suggested spread factor set. Here l_{max} represents the maximum lag of the series. Finally, using the obtained lag and the suggested spread factor set, the ensemble architecture is trained one time for forecasting.

The lag parameter is generally used in generating a data set for a given time series. In addition to this, the authors in [60] use this parameter for partitioning the data set and propose two schemes. In the first scheme, the data set is systematically partitioned into k subsets of approximately the same size when the lag is k. The same approach is also used in [14]. Unlike the first scheme, the second scheme partitions the data set into k mutually exclusive subsets. The subsets of data produced by these schemes are then used to construct an ensembles consisting of k MLPs.

2.7 Problems of the existing method

A good ensemble is produced when its individual members are both accurate and diverse i.e., the members have low error rates and their errors are uncorrelated [23], [61], [62]. A careful scrutiny of the existing work reveals that all other works but [2] emphasize only on the diversity of the base predictors. Diversity in those works is encouraged by varying the data sets used for training the predictors (e.g. [14], [46], [49], [59]), varying the parameters involved with the predictors (e.g. [23], [58]) and varying the type of base predictors (e.g. [40], [55]). Not only that except [2] all other works do not consider accuracy and diversity in combining predictors for constructing ensembles. Although the method proposed in [2] ensures accuracy of the predictors by using an appropriate lag, the base predictors will be less diverse as they differ only by the spread factor. It has been known that training predictors using different data is more effective for maintaining diversity [63], [64]. Furthermore, the way [2] finds the appropriate lag is computationally expensive. This is because an ensemble has to sequentially train l_{max} times for finding the appropriate lag. So we are motivated to develop a layered based ensemble architecture for time series forecast that will capture not only the methods of accuracy for ensemble generation of traditional methods using lag parameter but also incorporate the powerful diversity mechanism of bagging.

Chapter 3

Proposed Method

Motivated by the problems stated in previous chapter, we have devised a novel layer based ensemble architecture LEA for TSF. In this chapter, we describe the details of LEA algorithm. The objective of LEA is twofold, to improve the accuracy and diversity of the base predictors, and to improve forecasting accuracy. LEA technique employs a layer-wise mechanism by which important lag or time window for a time series is identified first in layer one. Then using this lag information, design an ensemble where the base predictors have higher accuracy and diversity. Finally, LEA uses a powerful combination algorithm to provide a final forecast.

3.1 Layered Ensemble Architecture

In order to reduce the detrimental effect of using a pre-defined lag and to devise an efficient forecasting scheme, a layered ensemble approach, LEA, is adopted in this work. Ensembles' requirement for maintaining diversity and accuracy among the base predictors matches well with the emphasis on using a layered architecture. In its current implementation, LEA uses MLP networks as base predictors.

The major steps of LEA can be described by fig. 3.1, which are explained further as follows.

1. Preprocess data of a given time series for handling seasonality, noise and missing attribute values.

2. Hold out k data points (observations) for testing LEA and use the remaining observations for constructing the forecasting model.

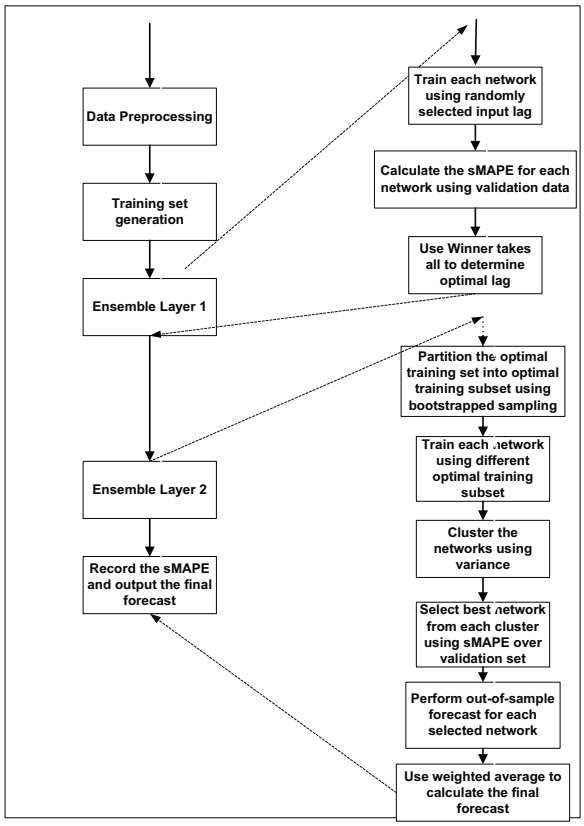

Figure 3.1: Major steps of our algorithm

3. **Ensemble Layer 1**

   (a) Generate an ensemble consisting of N MLP networks. Here N is a user-defined parameter and greater than l_{max}, the maximum lag of the series. For example, l_{max} can be 12 for a monthly time series.

   (b) Assign a random lag, l_i, to the network i of the ensemble. This is done by generating a number uniformly at random within 1 and l_{max}.

   (c) Define the architecture of each network in the ensemble. The network has an input layer, a hidden layer and an output layer. The number of nodes in the input and hidden layers equals to the lag assigned to the network, while the number of nodes in the output layer is one.

   (d) Create N training sets, one for each network, using the lags assigned to all N networks in the ensemble.

   (e) Train each network in the ensemble on the training set generated for it using the Levenberg-

Marquardt (LM) algorithm.

(f) Evaluate each network in the ensemble on a validation set containing n data points. The symmetric mean absolute percent error ($sMAPE$) is used for evaluation. According to [1], it can be expressed as

$$sMAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i - \hat{Y}_i|}{(Y_i + \hat{Y}_i)/2} \times 100 \tag{3.1}$$

where Y_i and $\hat{Y}_i$ are the true and predicted values for the i-th time point, respectively. The above equation provides a value between 0% and 200%. The smaller the $sMAPE$ is, the better the prediction accuracy is. We use $sMAPE$ because it is used in many previous forecasting competitions (e.g. NN3 [1], NN5 [4]) and previous studies (e.g. [65], [66]). However, any other performance measure can be used for evaluating the networks.

(g) Find the ensemble output by applying the *winner-take-all* method on the $sMAPE$s of all the N networks. Since the aim of the first layer is to find appropriate lag, the output of the ensemble is a lag that provides the lowest $sMAPE$ on the validation set.

4. **Ensemble Layer 2**

(a) Generate an ensemble consisting of N MLP networks.

(b) Assign the same lag, which is obtained by the first layer, to each network in the ensemble.

(c) Define the architectures of networks in the same way as described in the step 3c. Since the lag assigned to each network is same, the architectures of all N networks in this layer will be same.

(d) Create a training set, D_{tr}, using the lag.

(e) Train each network, j, in the ensemble on a training subset D_{tr}^{j} using the LM algorithm. Training subsets are created from D_{tr} by re-sampling its k% data so that that there is some differences among the subsets. Bagging [16] or boosting [17] can be used for creating such sets.

(f) Use the model selection and combination algorithm 3.1 to obtain output of the ensemble. The ensemble output here indicates the final forecast.

The above layered ensemble architecture appears to be straight forward, but its essence is the techniques incorporated for maintaining accuracy and diversity among the base predictors of the ensembles. Our LEA also has some other components. Details of them are given in the following sections.

3.1.1 Data preprocessing

Noise removal

In many TSF problems (e.g. time series of NN3 [1]), data points in the time series are heavily influenced by noise. A data point is considered as noise if its value is very much different from other values of the series. Failure to take specific measures against noise may lead to a bad forecasting performance.

There are several ways by which one can remove noise from time series data. For example, a large second order difference value is used as an indication of noise in [67]. In this paper, we use a noise detection mechanism proposed in [2], where a data point is identified as a noise whose absolute value is four times greater than the absolute medians of the three consecutive points before and after that point. That is, Y_i is a noise if its value satisfies the following condition: $Y_i \geq 4 \times max\{|m_a|, |m_b|\}$, where $m_a = median(Y_{i-3}, Y_{i-2}, Y_{i-1})$ and $m_b = median(Y_{i+3}, Y_{i+2}, Y_{i+1})$. When a data point is identified as a noise, its value is simply replaced by the average value of the two points that are immediately before and after it.

Treatment of missing value

Depending upon the nature of time series, it might be necessary to perform further data preprocessing. Treatment of missing value is one of those. Specifically, the time series of NN5 [4] competition requires a preprocessing step called *gaps removal*. There are basically two types of anomalies in the NN5 time series data. One is zero value that indicates no money withdrawal occurred on that particular time point and another is missing observations for which no value is recorded. Gaps removal strategies try to identify those anomalies and remove them. In this work, we adopt the gap removal method proposed in [68]: if Y_i is the gap sample, this method replaces the gap with the median of the set $[Y_{i-365}, Y_{i+365},\ Y_{i-7}$ and $Y_{i+7}]$ among which are available.

Deseasonalization

Treatment of seasonality is one of the major issues in TSF literature, because many time series (e.g. time series of NN3 [1] and NN5 [4]) contain some seasonality. There are basically two methods, namely, direct and deseasonalized, for handling seasonality [35]. In the direct method, the base predictors are trained directly on the raw data, whereas in the latter method, seasonal adjustments are made on the raw data before the predictors are trained on.

For the seasonal series of NN3, we adopt a simple deseasonalization procedure suggested in [69]. To obtain a deseasonalized series it simply subtracts the seasonal average from the series. Since the time series of NN5 possess a variety of periodic patterns, the deseasonalization methodology discussed in [35] is adopted here to remove the strong day of the week seasonality as well as the moderate day of the month seasonality. To make final forecast, we restore back the seasonality to provide the output of the forecasting model.

3.1.2 Training set generation

A training set in the form $\{X_i; Y_i\}_{i\ =\ 1\ to\ N}$ is necessary for obtaining optimal or near optimal weights of an MLP network using the LM algorithm. Here X_i represents inputs to the network and may have several components i.e., $X_i = x_{11}, x_{12}, \ldots, x_{1d}$. For the sake of simplicity and without loss of generality, we assume that the output Y_i has one component. To generate the training set for a TSF problem, we need some extra effort because only data points of a given time series are available. The parameter needed in such generation is the lag, which determines how many previous data points will influence the next point.

An appropriate lag of a time series is not known in advance. As mentioned before, the aim of our ensemble layer 1 is to find the appropriate lag. Lacking of knowledge about such a lag enforces LEA to vary the lag from 1 to l_{max}. And LEA generates a different training set using each of different lags. Let the lag equals to 5 and the data points $d_1, d_2, \ldots, d_k$ are used for generating the training set. The generation process takes the lag as a window and shifts it in generating the training set. That is, $X_1 = d_1 \ldots d_5$ and $Y_1 = d_6$, $X_2 = d_2 \ldots d_6$ and $Y_2 = d_7$ and this process continues until the Y_i reaches at the end of the series i.e., d_k. It is now clear that it is possible to get a different training set by using a different lag.

In the ensemble layer 2, the training sets for base predictors are generated in two steps. At the

first step, using the lag obtained from the ensemble layer 1 and the data points $d_1, d_2, \ldots, d_k$, LEA generates the training set, D_{tr}. In the second step, bootstrapped sampling is applied on D_{tr} for generating N training sets, one for each base predictor in the ensemble layer 2.

3.1.3 Base predictors

Since the nature of typical time series output variables is continuous, individual base predictor models should be chosen in such a way so that it includes universal and flexible regression models capable of handling multiple inputs and multiple outputs. Neural networks are considered to be a universal nonlinear regression model. It has the ability to control its complexity and diversity by varying network architectures and initialization conditions, or cross-training. Given all these advantages, we decided to choose a simple Feedforward Multilayer Perceptron (MLP) as a base predictor model that would be used to test the presented architecture against standard predictors and combiners. The chosen NN is trained using an efficient modified BP (MBP) with adaptive learning rates that scales linearly with the number of parameters to be optimized.

The factors related to neural network model architecture include the number of input variables, the number of hidden layers and hidden nodes, the number of output nodes, the activation functions for hidden and output nodes, and the training algorithm and process. Because a single hidden-layer network has been shown both theoretically and empirically capable of modeling any type of functional relationship, it is used in our experimental study. The activation functions used for all hidden nodes are the logistic function while the identity function is employed in the output layer.

The number of input nodes is perhaps the most critical parameter since it corresponds to the number of past lagged observations used to capture the underlying pattern of the time series. Since there is no theoretical result suggesting the best number of lags for a nonlinear forecasting problem, we will experimentally vary this parameter from 1 to 12 in ensemble layer 1. Another important factor is the number of hidden nodes, which empowers neural networks with the nonlinear modeling capability. It has been shown that the in-sample forecasting and the out-of-sample forecasting ability of neural networks are not very sensitive to the number of hidden nodes [43]. Therefore, for the sake of simplicity, number of hidden nodes is kept same as the number of input nodes in the ensemble layer 1. Since we are varying the lag number, so in the ensemble layer 1 we are obtaining a set of base predictors with different architecture (i.e. different number of input nodes and hidden nodes). In the ensemble layer 2, we are using a fixed lag number obtained from ensemble layer 1, so the

architecture of base predictors in this layer is same except some variation in the initial weight of the base predictors.

3.1.4 Model selection for ensemble

There are two important goals of a good ensemble. One is improving the accuracy of the weak predictors so that the combined system is a strong one and another is combination of well-trained complementary base predictors. Individual base predictors are said to be complimentary if they could offset each other's deficiencies to solve a particular problem. Since achieving the aforementioned two goals is a difficult task and also the number of individual base predictors in an ensemble system is not known in advance, the generation of the member of an ensemble system is usually divided into two phases [70]. A set of individual base predictors are first generated and then using a selection mechanism several accurate and diverse base predictors are chosen for combination [61], [71]. Usually for regression problems, variance is used to measure the diversity [72] and simple averaging or weighted averaging is the most common method for combination [73].

Since LEA has two different layers of ensembles, it is necessary to use a proper selection and combination method so that the objective of each layer can be achieved. In the ensemble layer 1, we evaluate each network by *sMAPE* over the validation set and the output of the ensemble i.e., the optimal lag is obtained using the *winner-take-all* method. Other combining strategies like averaging or weighted averaging is not suitable for this layer, because the lag should be an integer number and averaging or weighted averaging might gives a fractional number, which is unacceptable for the generation of training data. We might truncate the fractional part, but it may introduce error. Besides, each MLP in ensemble layer 1 comes up with a *sMAPE* over the validation set for the specific lag assigned to it. The optimal lag is the one for which a MLP comes up with a lowest *sMAPE*. Hence, averaging or weighted averaging does not make any sense because it will simply make an average or weighted average over the randomly generated lag. Note that we randomly assign a lag to each network of the ensemble layer 1.

To provide a better forecast, we improvise a model selection and combination method for the ensemble layer 2, which is based on clustering over the variance of MLP networks. The calculation of variance is quite straight forward. For each MLP, we first vary its training set by adding random noise and calculate how much error it makes. Then we calculate the variance based on the errors. To ensure accuracy and diversity of the selected networks, we cluster all the generated networks using

their variance. The networks in the same cluster indicate that they are much more alike, while they in different clusters indicate diversity among them. Finally, we select the best network based on *sMAPE* from each cluster and combine all the selected networks by weighted average. The weight of a network is inversely proportional to its corresponding *sMAPE* over the validation set. The pseudo code of the whole process is given in Algorithm 3.1.

Algorithm 3.1 Model selection and combination algorithm in ensemble layer 2

Require: Ensemble size N, cluster size c.

1: **for** $i = 1$ to N **do**
2: calculate the variance of each MLPs N_i.
3: **end for**
4: Cluster the N MLPs into c classes ($c < N$) by their variance value.
5: Select one MLP from each cluster which has lowest *sMAPE* over validation set in that cluster.
6: For each selected MLP calculate combination weight using their *sMAPE*.
7: For each selected MLP perform out-of-sample prediction.
8: Provide the final out-of-sample forecast using weighted average.

3.1.5 Accuracy and Diversity

The idea of combining multiple predictors is based on the observation that achieving optimal performance in combination is not necessarily consistent with obtaining the best performance for an individual (base) predictor. The idea is that it may be easier to optimize the design of a combination of relatively simple predictors than to optimize the design of a single complex predictor. So, an ensemble are often more accurate than any of its individual members and a necessary and sufficient condition for this is that individual member of the ensemble be **accurate** and **diverse**. Accuracy of a predictor means how accurately a base predictor predicts value (i.e. within a threshold error rate) whereas two individual predictors are diverse when their out of sample errors are uncorrelated (the errors are independent random variables).

Although it is known that diversity among base predictors is a necessary condition for improvement in ensemble performance, there is no general guideline about how to ensure diversity among the base predictors. In our algorithm LEA, ensuring accuracy and diversity among the member of the ensemble is very crucial since we are generating ensemble in two completely different layer and ensemble generation in the layer 2 is completely dependent upon the result of ensemble in layer 1.

According to TSF literature, accuracy of the base predictor largely depends upon the time window or lag parameter of time series. Since for a particular time series, we do not have any knowledge of

optimal lag, the motivation of ensemble layer 1 is to discover the optimal lag through a set of diverse base predictors. In this layer, we generate a set of diverse base predictors by varying architecture of base predictors and different training set. Training set are generated using different time window or lag parameter (i.e. 1 to 12). Since the number of input nodes and hidden nodes in the input and hidden layer is equal to the lag parameter, so architecture of the base predictors also get varied. To ensure that each of the time window or input lag from 1 to 12 are covered by the ensemble, we set the number of neural network N which is greater than l_{max}. By using this setting, we can expect at least one of the base predictor will come up with lowest error on the validation set, thus ensuring the accuracy. Since each neural network is trained using a randomly selected training set the mentioned strategy fulfils two of the important aspect of ensemble design. We are getting more than one expert in certain region of our search space which implies accuracy and at the same time getting diverse expert on different area of the search space which implies diversity.

Using the best lag from ensemble layer 1, we have to build up the ensemble in the layer 2, in more sensible and constructive way. A careful scrutiny in the design of ensemble layer 1, reveals that we are obtaining the best lag for a particular base predictor (a fixed number of input nodes and hidden nodes equal to best time window). To answer the question of ensuring accuracy, it will be sensible to replicate the same architecture using optimal lag in all the members of ensemble layer 2. As a result, the architecture of the all base predictors in this layer is same with only slight variation in the initial weight. Diversity in this layer is obtained using random data sampling. For this purpose, training set D_{tr} is generated using best lag. Then bootstrapped sampling is applied on D_{tr} to obtain several training subset D_{tr}^{j} where $j = 1, 2, \ldots, N$. These training subsets D_{tr}^{j} are used to train the neural networks of ensemble layer 2.

Our data partitioning schemes serves a most important aspect of time series prediction problem: maintains the correlation between the value at Y_t and the values from the previous elements of the time series $Y_{t-lag}, \ldots, Y_{t-2}, Y_{t-1}$. Like classification problem, we cannot use the randomly partitioned data to train the neural network for TSF. However, we will find that most of the literature like the authors in [59] employed this naïve strategy to enforce diversity. And this is the single most reason for the failure of accurate prediction for many state of the art algorithms in the literature of ensemble. We are applying our proposed data partitioning scheme on D_{tr} that is already generated considering the correlation (i. e., best time window or lag parameter) which eventually reduces harmful correlation effect among predictions.

3.2 Difference with existing work

In this section, we try to investigate the novelty of our LEA by comparing it with very recent state-of-the art ensemble based algorithm for TSF [2]. We establish here the difference from the following perspectives.

- **Diversification among the member of ensemble:** In [2] the author proposes an ensemble of radial basis function (RBF) network for TSF. The ensemble constitutes of three RBFs. All three members of the ensemble in both first and second layer receive the same data. In the first layer, members of the ensemble use the all possible input lag $[1, 2, \ldots, 12]$ and in second layer those member use the best lag as an input.

  However, in LEA different members of ensemble use the different lag as an input, which implies different data. In ensemble layer 2, all member of the ensemble use the best input lag for generating the training set D_{tr} and a random re-sampling is applied on D_{tr} to obtain D_{tr}^{j} to train each member j where $j = 1, 2, \ldots, N$.

  Since, in [2] all the members of the ensemble in layer 1, are trained on each possible lag $[1, 2, \ldots, 12]$, this method is not developing expert on different area of the solution space rather making all the member as equal expert. This is not the property of ensemble where most of the member is actually a weak learner. But in LEA, different members of the ensemble in layer 1, get trained on different lagged training set so we are getting experts from different solution space.

  Similarly, in ensemble layer 2, [2] is doing exactly the same thing as he has done in the layer 1, except this time using the best lag. There is no data variation here also. The author is using the structure like an ensemble but applying no property of ensemble here.

- **Pattern retraining for better forecast:** In [2] the author simply uses the best lag for all the three RBFs not so much different than layer 1. As a result, the pattern that might be appear in the test set is not getting any kind of extra treatment here.

  But in LEA, the data variation is obtained by applying random re-sampling on the training set, and this random re-sampling will increase the probability of the appearance of the patterns which might appear in the testing set and as a result will get more emphasized in the training set. As a result, we can expect more accurate forecast from LEA.

- **Computational effectiveness:** For all possible input lag $[1, 2, \ldots, 12]$, the three member of ensemble in [2] are trained. That means for 12 lags there are 12 training set and each member is trained 12 times. If the number of lag is a large number like 100, then in [2] each member will be required to train using all possible 100 lags which is computationally very expensive but in LEA we are training NNs using random lag, which is far more computationally effective.

- **Dependency on the nature of data:** In [2] the only parameter of the architecture is spread factor set and it is again dependent on the distribution of the dataset. For different dataset the spread factor set is different and the author is not sure whether the spread factor set will really achieve a good learner. The author is using three members with three different spread factor set to sort out the problem of finding a good member. To calculate the spread factor set, the author empirically take three percentile (25th, 50th, 75th) for each time series of NN3 competition [1]. That means for other types of time series data this strategy might not work well.

  But, in LEA we do not have any specific requirement of the distribution of the dataset. The only input of LEA is lag number and the ensemble size.

- **Combining the output:** The author [2] uses simple average to calculate the final forecast. For the computation of the best lag the author employs *winner-takes-all* over the validation set. In LEA, we also do the same for determining the best lag. However, for making final forecast we propose a new combination algorithm 3.1 considering the accuracy and diversity issues of ensemble.

  Since all the member of the ensemble have the same data and the architecture, the author in [2] is applying average as a combination rule. But in LEA, we exploit all possible search space of the time series forecasting and a simple averaging will drastically reduces the accuracy of our system because of random re-sampling some members of the ensemble will produce a very large error and some members will produce less error. So picking up the members of the ensemble for making final forecast is very crucial in this stage. Thats why we envision a new combination algorithm 3.1 which is based on clustering the variance of the NNs.

Chapter 4

Experimental Studies

In this chapter, we evaluate the effectiveness of our proposed LEA and compare its performance with different ensemble and non-ensemble methods existing in literature. We have selected two competitive ensemble algorithms bagging [16] and boosting [17]. Since, LEA uses basic bagging algorithm, so we provide an extensive simulation and result analysis against basic bagging algorithm. We evaluate the performance on some real world data sets collected from NN3 [1] and NN5 [4] forecasting competition. We also establish the statistical significance of LEA using Wilcoxon signed rank test for all three performance metrics used in this thesis.

4.1 Simulation framework

In this section, we describe the characteristics of the dataset used in this thesis. Next, we discuss about the performance metrics used to evaluate the performance of LEA. A brief discussion about the experimental setup will conclude this section.

4.1.1 NN3 data set

We first evaluate and compare the performance of LEA using NN3 [1] time series competition dataset. The competition provides two datasets: A and B. The dataset A contains 111 monthly time series drawn from a homogeneous population of empirical business time series, while the dataset B contains only 11 series taken from A. These datasets can be obtained from http://www.neural-forecasting-competition.com/NN3/datasets.htm. We use the data set A for our simulation and comparison.

Organizers of the NN3 competition categorize the time series into long and short based on the

Table 4.1: Characteristics of NN3 [1] Forecasting Competition Data

	short	long	Sum
non-seasonal	25	32	57
seasonal	25	29	54
Sum	50	61	111

length of series. The short series contains less than 50 data points, but the long one contains more than 50 data points. According to the characteristics of data, the time series can be further categorized into seasonal and non seasonal. In a seasonal time series, there exists regularly spaced peaks and troughs that have a consistent direction and approximately have the same magnitude at every period. These are, however, not present in a non-seasonal one. The number of series in each category is presented in table 4.1.

4.1.2 Performance measure

The global performance of a forecasting model is usually evaluated by some accuracy measure such as *sMAPE* [66], median root absolute error (*MdRAE*) [74] and mean absolute scaled error (*MASE*) [74]. *sMAPE* is used in NN3 [1], NN5 [4] and NN GC1 [75] forecasting competitions. One advantage of *sMAPE* is its scale independence property, which is suitable for comparing different methods across various series. Although *sMAPE* has been originally proposed in [76], most of the authors adopt the variant proposed in [66], which does not lead to negative values. We also use this variant in this work.

A particular performance measure may favor a forecasting method if the measure implicitly or explicitly satisfies the assumption or condition of the method. Hence, in order to make exhaustive evaluation, we use *MdRAE* and *MASE* in addition to *sMAPE*. The measures, *MdRAE* and *MASE*, can be expressed as

$$MdRAE = median(|r_i|), \quad r_i = \frac{Y_i - \hat{Y}_i}{Y_i + \hat{Y}_i^*} \tag{4.1}$$

$$MASE = \frac{\sum_{i=1}^{n} |e_i|}{\frac{n}{n-1}\sum_{i=2}^{n} |Y_i - Y_{i-1}|}, \quad e_i = Y_i - \hat{Y}_i \tag{4.2}$$

where $\hat{Y}_t^*$ is the forecast made by a reference method i.e., random walk [76] applied on the series data for a given forecast horizon h.

The error measure *MdRAE* has been considered as the most reliable error measures for a large number of applications e.g. [76], [77]. However, in the case of equal consecutive observations, this error measure returns infinite, a serious deficiency of *MdRAE* [74]. The authors in [76] suggest that *MdRAE* is appropriate only for a very small set of series. The most important advantage of *MdRAE* is its better protection against noise [76].

The error measure *MASE* scales the error based on the in-sample mean absolute error from the naïve method and are independent of the scale of the data. The authors in [74], [78], [79] recommend that *MASE* to become the standard measure for forecast accuracy. This due to the fact that *MASE* is always defined and finite, unlike other measures in certain occasions.

4.1.3 Experimental setup

An MLP network containing one hidden layer with the tan sigmoid activation function is used as the base predictor for the ensemble layers 1 and 2. The learning rate and momentum term of the LM algorithm are set to 0.1 and 0.4, respectively. We stop a learning process after 1000 epochs or when the root mean square error of the predictor reaches to 0.00001. The ensemble in layer 1 and 2 is consisted of 50 MLP networks. It is important to note that we use the aforementioned parameter settings for all time series and they are not meant to be optimal.

According to [1], we withhold the last 18 data points of every time series for testing and use the remaining data points for building forecasting models. More specifically, from the remaining data points, we use 80% data for training and 20% data for validation. We normalize each data point of the series to zero mean and unitary variance. The normalization parameters are computed from the training and validation data, and then applied to training, validation and testing data.

The layer based ensemble architecture for TSF proposed in this paper is implemented using MATLAB (R2012a, The Mathworks, Inc., Natick, MA, USA). MLP networks are implemented using the Neural Network Toolbox of MATLAB. The source codes of LEA is available from the author.

4.2 Results

We first compare our LEA with basic bagging and boosting to show the effect of layering on the performance of these basic ensemble models. We then compare LEA with several other ensemble, non-ensemble and benchmark statistical methods.

4.2.1 Comparison with Bagging

As mentioned in section 3.1, LEA re-samples $k\%$ data points to create different training sets for different base predictors of the ensemble layer 2. To make a fair comparison with bagging, LEA uses here bagging as a method for re-sampling. We call this version of LEA as layered bagging. The value of k used for LEA is set to 9%.

Tables 4.2-4.5 and fig. 4.1-4.6 show the results of basic bagging and layered bagging. It is to be noted that for the calculation of mean, minimum (min), maximum (max) and standard deviation (STD.) we have used simply the accuracy value of 111 time series of NN3 competition. For example, mean *sMAPE* is the average of the *sMAPE* across all the 111 time series. Similarly we have calculated the others. The following observations can be made from these tables and figures.

- It can be seen that the lag obtained by the ensemble layer 1 of LEA is different for different time series (Table 4.2). These results indicate that it is very important to determine the lag of a given time series automatically. Note that the basic bagging algorithm uses the same lag for all time series. According to the suggestion of NN3 [1], the lag is set to 12 for basic bagging.

- In terms of average *sMAPE*, *MASE* and *MdRAE*, layered bagging is found better than basic bagging irrespective of the nature of time series i.e., seasonal, non-seasonal, long or short. For example, for the seasonal time series, the average *MdRAE* achieved by basic bagging is 0.61, whereas it achieved by layered bagging is around 0.54 (Table 4.3). The better average performance indicates the finer approximation capability of layered bagging for different time series.

- Our layered bagging is always found better than basic bagging when we compare these two methods based on minimum *sMAPE*, *MASE* or *MdRAE*. However, in terms of maximum *sMAPE*, *MASE* and *MdRAE*, basic bagging is found better than layered bagging for two cases out of 12 cases (four different time series and three different performance metrics) (Tables 4.3-4.4).

- The performance of layered bagging is found more consistent compared to bagging across different time series. This can be observed by looking the standard deviation (std) of these two methods (Tables 4.3-4.4). In terms of std, layered bagging is found better for nine cases, while basic bagging is found better for three cases.

- To get an idea about the performance on different time series, we count the number of times layered bagging is better or worse compared to bagging. Using any error measurement, if layered bagging exhibits better performance than bagging for a specific time series, we call it as win for layered bagging; otherwise it is a loss. Once again, layered bagging is proved superior than basic bagging with respect to three different performance metrics (Table 4.5).

- Fig. 4.1 and 4.2 illustrate the comparison of forecast between the layered bagging and basic bagging for all four (seasonal, non-seasonal, short and long) types of time series. For each type of time series, we select one easy time series and one complex time series. We make this kind of classification (easy and complex) using the information provided in [80]. The magenta and black line represent the forecast made by layered bagging and basic bagging respectively, whereas the real value is shown using green line. The vertical blue line separate the training data from the testing data. It can be observed from the fig. 4.1 and 4.2 for easy time series, layered bagging is following almost the same pattern with real time series. For example, for the easy long time series number 91, the magenta line (layered bagging) follows exactly the same pattern of the green line (real time series value). Basic bagging also tries to follow the pattern but has a lower accuracy. For complex time series, it seems like basic bagging is behaving more like a random walk, whereas layered bagging is trying to catch the pattern of real time series. A short glance at the time series number 19 and 102 reveals the fact.

- Finally, in order to get an overview of per series wise performance of layered bagging over basic bagging we also plot per series wise star plot using all three performance metrics as shown in fig. 4.3-4.5. It can be observed from these fig. 4.3-4.5 that basic bagging is surrounding the layered bagging in all most all case irrespective of any performance metric which again indicates that basic bagging has higher forecasting error than layered bagging.

- It can be observed from fig. 4.6 that layered bagging is better than basic bagging for all forecast horizons except one. For example, for the non-seasonal time series, the average *sMAPE* of basic bagging is better than layered bagging for the forecasting horizon 4. However, layered bagging is found better than basic bagging for all other cases. It is also observed that the average *sMAPE* of layered bagging and basic bagging increases in many instances with the increase of forecast horizon. Similar results are found for *MASE* and *MdRAE* as shown in fig. 4.7 and 4.8.

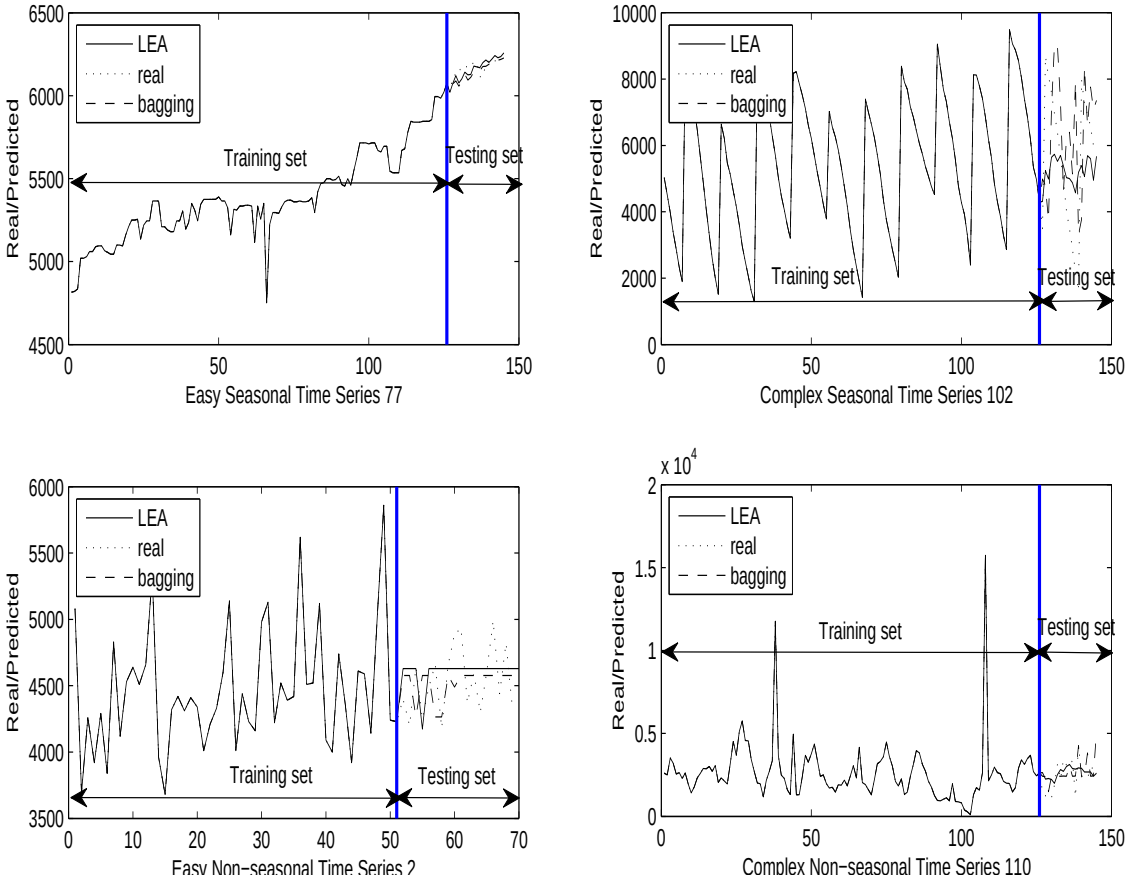

Figure 4.1: Comparison of forecast for seasonal and non-seasonal time series between basic bagging and layered bagging

We use the Wilcoxon signed rank test to assess whether the performance difference between layered bagging and basic bagging is statistically significant. Compared to the *t-test*, the ranked test makes fewer and less stringent assumptions on the sample distributions and thus is more powerful in detecting the existence of significant differences. In our test, the matched pair samples are the per-series *sMAPE*, *MASE* and *MdRAE* of the seasonal, non-seasonal, long and short time series with forecasting horizon 18.

Table 4.6 shows the summary of the Wilcoxon signed rank test [81] based on the *sMAPE*, *MASE* and *MdRAE* of the four different types time series. Here, R^+ corresponds to the sum of ranks for layered bagging and R^- for the basic bagging algorithm. These notations are used throughout the paper. The results show that the null hypothesis has been rejected in favor of layered bagging with

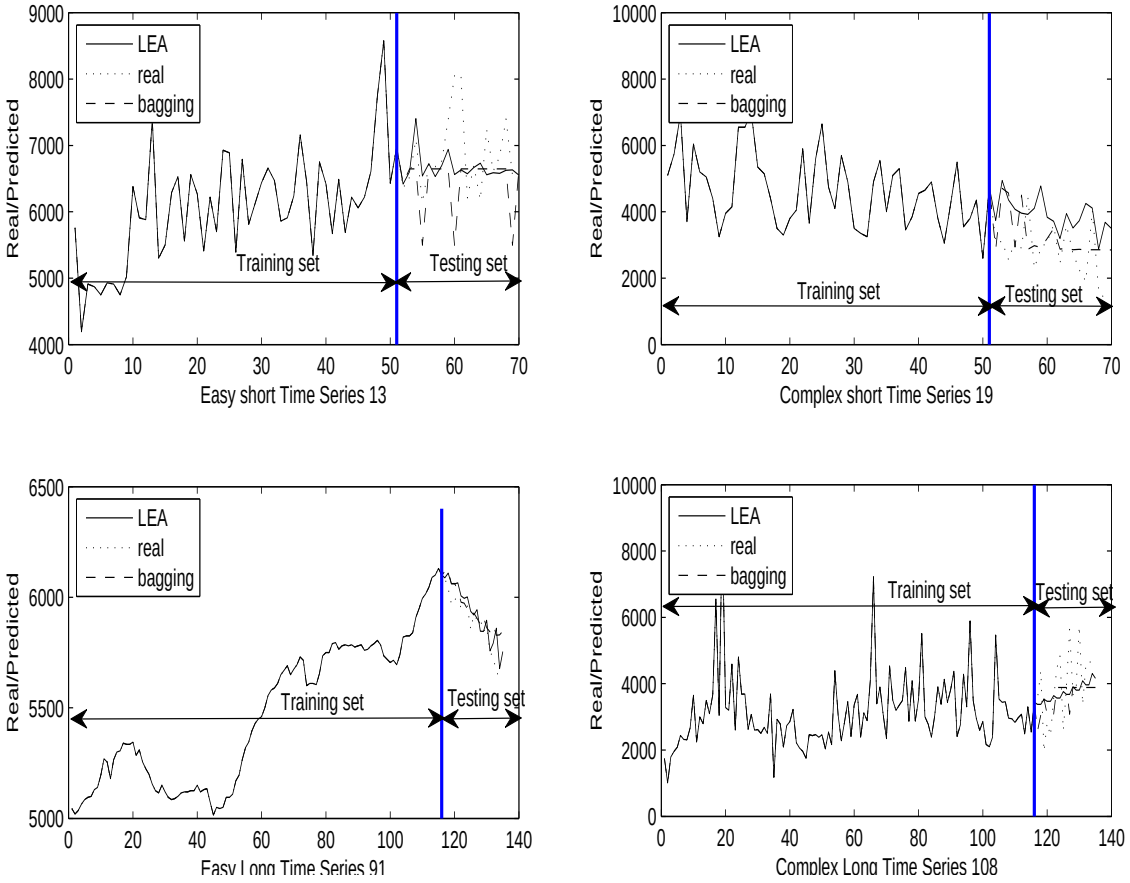

Figure 4.2: Comparison of forecast for short and long time series between basic bagging and layered bagging

a significance level 0.05 for seasonal, non-seasonal, short and long time series. In fact the associated p-values indicate that even a significance level 0.001 would have resulted in the rejection of null hypothesis in favor of layered bagging.

4.2.2 Analysis

In order to understand the reasons behind the better performance of LEA, we analyze the ensembles produced by layered bagging and basic bagging. We employ bias-variance-covariance decomposition, double fault and disagreement for analysis.

Table 4.2: Best lag obtained from ensemble layer 1 for different time series data of NN3 [1] competition

		seasonal	non seasonal	short	long
LEA	mean	7.426	6.8837	6.8	7.557
	minimum	1	1	1	1
	maximum	12	12	12	12
	STD.	3.068	3.5133	3.307	3.175

Table 4.3: Comparison between basic bagging and layered bagging in terms of *sMAPE*, *MASE* and *MdRAE* for seasonal and non-seasonal time series data of NN3 [1] competition. Here the best result is highlighted using boldface text.

		Basic Bagging			Layered Bagging		
		sMAPE	MASE	MdRAE	sMAPE	MASE	MdRAE
seasonal	mean	14.220	1.380	0.618	**13.139**	**1.230**	**0.540**
	min	0.875	0.484	0.206	**0.798**	**0.437**	**0.183**
	max	60.584	8.324	1.291	**60.535**	**4.837**	**1.085**
	std	**11.784**	1.174	0.246	11.903	**0.898**	**0.234**
non-seasonal	mean	17.963	0.940	0.694	**16.448**	**0.679**	**0.565**
	min	5.819	0.499	0.289	**4.157**	**0.293**	**0.202**
	max	**76.622**	2.516	2.227	80.809	**1.707**	**0.967**
	std	13.703	0.501	0.340	**13.498**	**0.270**	**0.207**

Table 4.4: Comparison between basic bagging and layered bagging in terms of *sMAPE*, *MASE* and *MdRAE* for short and long time series data of NN3 [1] competition. Here the best result is highlighted using boldface text.

		Basic Bagging			Layered Bagging		
		sMAPE	MASE	MdRAE	sMAPE	MASE	MdRAE
short	mean	14.333	0.960	0.597	**13.464**	**0.748**	**0.496**
	min	5.819	0.499	0.206	**4.157**	**0.293**	**0.183**
	max	**40.682**	2.523	2.227	44.271	**2.355**	**0.967**
	std	**8.226**	0.535	0.348	8.339	**0.353**	**0.210**
long	mean	17.266	1.415	0.689	**15.205**	**1.236**	**0.595**
	min	0.875	0.484	0.273	**0.798**	**0.437**	**0.202**
	max	**76.622**	8.324	1.291	80.809	**4.837**	**1.085**
	std	15.320	1.213	**0.220**	**15.244**	**0.934**	0.225

Table 4.5: Comparison between basic bagging and layered bagging in terms of Win-loss count for the time series data of NN3 [1] competition. Here the best result is highlighted using boldface text.

Performance Metrics	Number of Wins	
	Basic Bagging	Layered Bagging
sMAPE	26	**85**
MASE	32	**79**
MdRAE	25	**86**

Table 4.6: Wilcoxon Signed Rank Test summary between layered bagging and basic bagging for the time series data of NN3 [1] competition.

	Performance Metrics	R^+	R^-	p-value	hypothesis Significance level=0.05
Seasonal	sMAPE	1899	447	9.16E-06	Rejected for layered bagging
	MASE	1802	544	0.00012	Rejected for layered bagging
	MdRAE	1867	479	2.2E-05	Rejected for layered bagging
Non-seasonal	sMAPE	819	127	2.94E-05	Rejected for layered bagging
	MASE	749	197	0.0086	Rejected for layered bagging
	MdRAE	787	159	0.00016	Rejected for layered bagging
Short	sMAPE	1068	207	3.24E-05	Rejected for layered bagging
	MASE	930	345	0.00048	Rejected for layered bagging
	MdRAE	1085	190	1.56E-05	Rejected for layered bagging
Long	sMAPE	1564	327	8.89E-06	Rejected for layered bagging
	MASE	1560	331	1.02E-05	Rejected for layered bagging
	MdRAE	1512	379	4.72E-05	Rejected for layered bagging

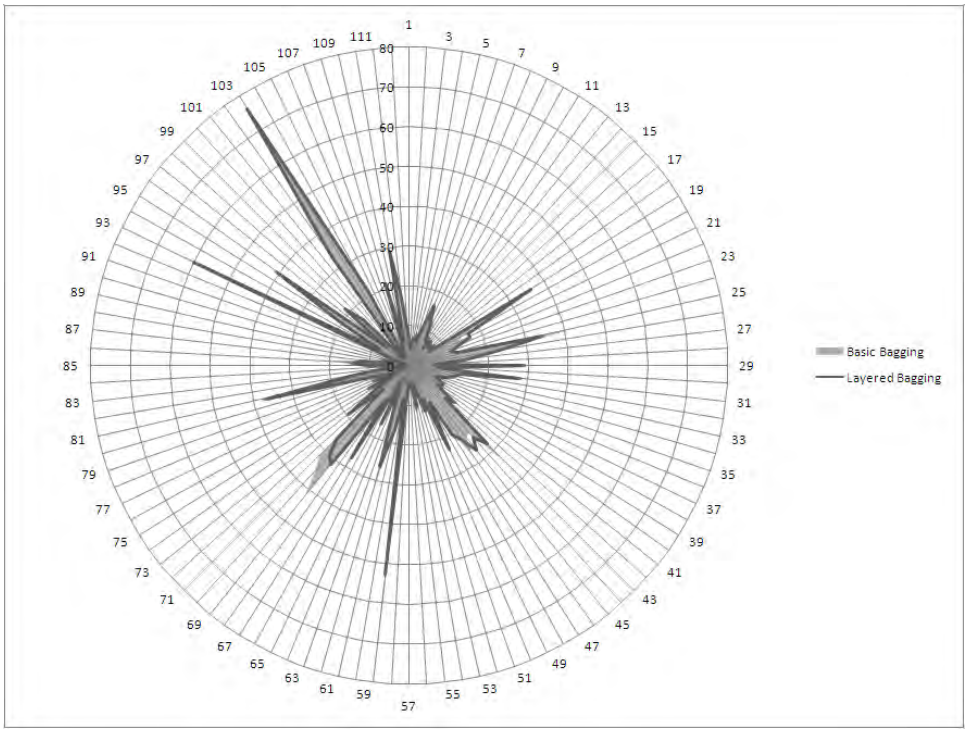

Figure 4.3: Comparison between basic bagging and layered bagging in terms of *sMAPE* on time series data of NN3 [1] competition using star plot.

Bias-Variance-Covariance estimation

Mean square error (E_{mse}) of an ensemble can be decomposed into bias (E_{bias}), variance (E_{var}) and co-variance (E_{cov}). For regression problems, this decomposition has been widely used (e.g. [82], [83]) for analyzing the performance of ensembles and can be expressed as

$$E_{mse} = E_{bias} + E_{var} + E_{cov} \tag{4.3}$$

The above equation indicates that to achieve good performance, the bias, variance and covariance of the ensemble should be small.

Let the average output of the ensemble and network i on the nth pattern in the testing set $n = 1, \ldots, N$ are denoted, respectively, by $\overline{\hat{Y}}(n)$ and $\overline{\hat{Y}_i}(n)$, which are given by Eqs. (4.4) and (4.5)

$$\overline{\hat{Y}}(n) = \frac{1}{K} \sum_{k=1}^{K} \hat{Y}^{(k)}(n) \tag{4.4}$$

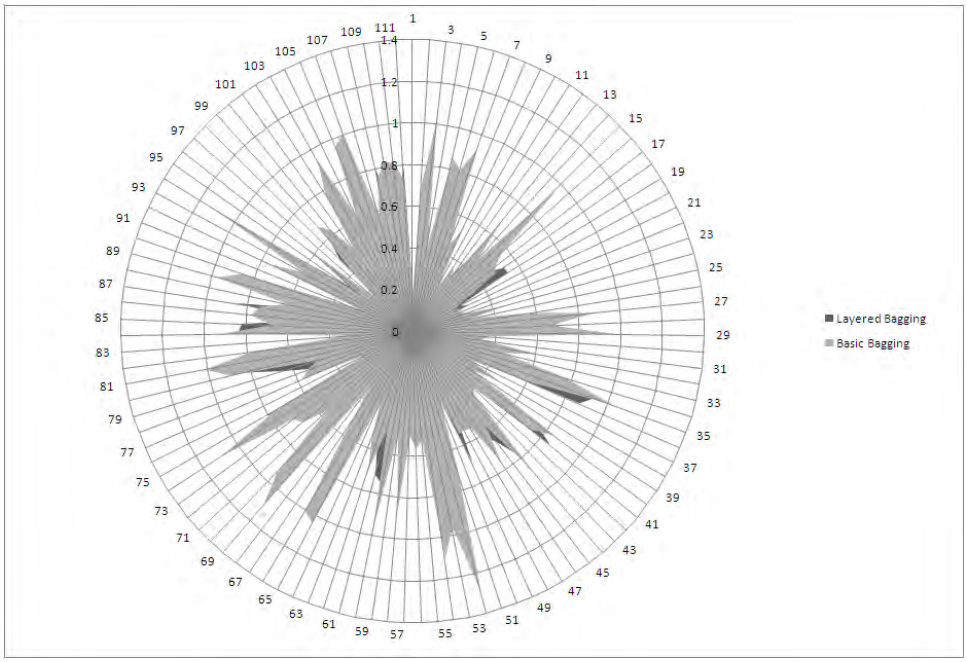

Figure 4.4: Comparison between basic bagging and layered bagging in terms of *MdRAE* on time series data of NN3 [1] competition using star plot.

$$\overline{\hat{Y}_i}(n) = \frac{1}{K}\sum_{k=1}^{K}\hat{Y}_i^{(k)}(n) \tag{4.5}$$

where $\hat{Y}^{(k)}(n)$ and $\hat{Y}_i^{(k)}(n)$ are the output of the ensemble and the network i on the nth pattern from the kth simulation. The bias, variance and covariance of the ensemble are defined by Eqs. (4.6), (4.7) and (4.8), respectively.

$$E_{bias} \equiv \frac{1}{N}\sum_{n=1}^{N}(\overline{\hat{Y}}(n) - Y(n))^2 \tag{4.6}$$

where $Y(n)$ is the true value on nth time point.

$$E_{var} \equiv \sum_{i=1}^{M}\frac{1}{N}\sum_{n=1}^{N}\frac{1}{K}\sum_{k=1}^{K}\frac{1}{M^2}(\hat{Y}_i^{(k)}(n)) - \overline{\hat{Y}_i}(n))^2 \tag{4.7}$$

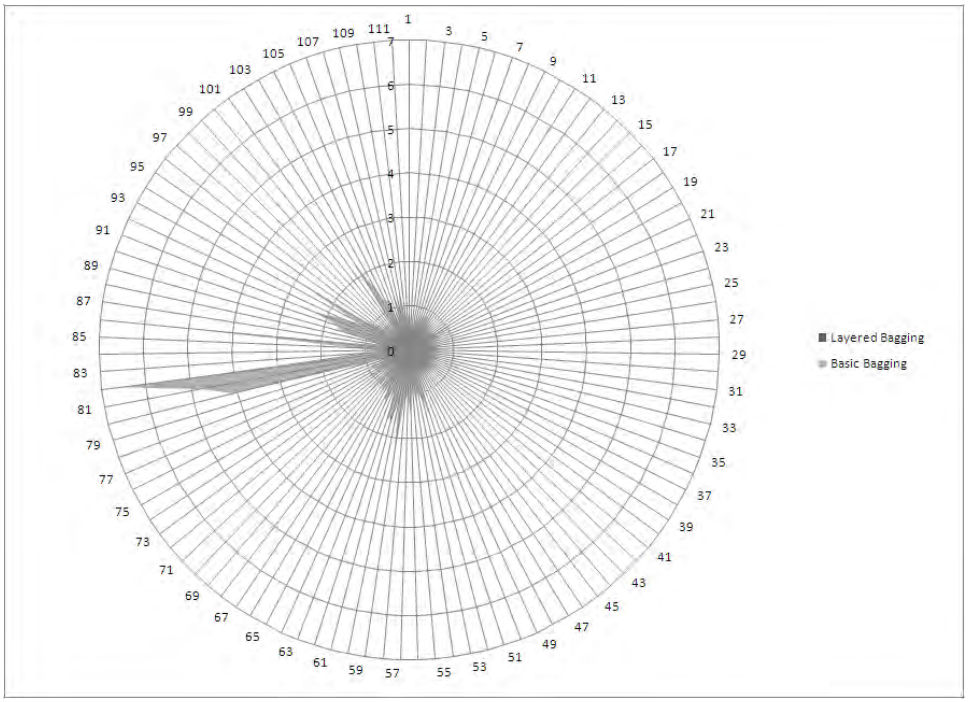

Figure 4.5: Comparison between basic bagging and layered bagging in terms of *MASE* on time series data of NN3 [1] competition using star plot.

$$E_{cov} \equiv \sum_{i=1}^{M} \sum_{j=1, j\neq i}^{M} \frac{1}{N} \sum_{n=1}^{N} \frac{1}{K} \sum_{k=1}^{K} \frac{1}{M^2} (\hat{Y}_i^{(k)}(n)) - \overline{\hat{Y}_i}(n))^2 (\hat{Y}_j^{(k)}(n)) - \overline{\hat{Y}_j}(n))^2 \tag{4.8}$$

E_{mse} can also be defined by Eq. (4.9).

$$E_{mse} \equiv \frac{1}{N} \sum_{n=1}^{N} \frac{1}{K} \sum_{k=1}^{K} (\hat{Y}^{(k)}(n) - Y(n))^2 \tag{4.9}$$

To obtain bias, variance and co-variance of an ensemble architecture, we follow the experimental methodology suggested in [83]. According to [83], several (say, 25) simulations of each ensemble architecture has to be conducted. The only difference in different simulations is the training sets used for training base predictors. Since NN3 competition [1] contains a large number of time series, we select only one series from each of four different types of time series, namely series number 71, 73, 2 and 110 for seasonal, non-seasonal. short and long series, respectively. However, similar results can

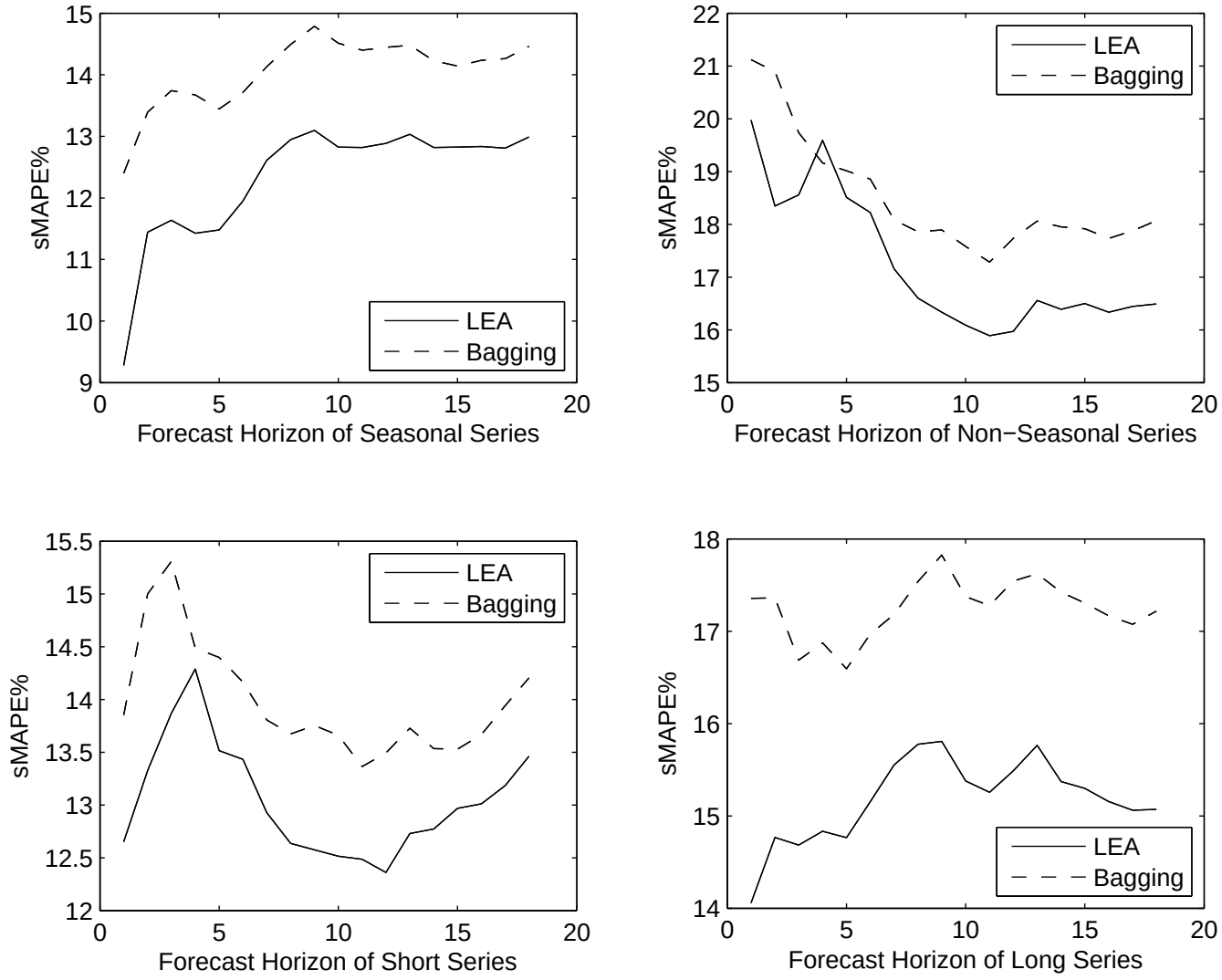

Figure 4.6: Comparison between basic bagging and layered bagging in terms of forecast horizon wise *sMAPE* for seasonal, non-seasonal, short and long time series data of NN3 [1] competition.

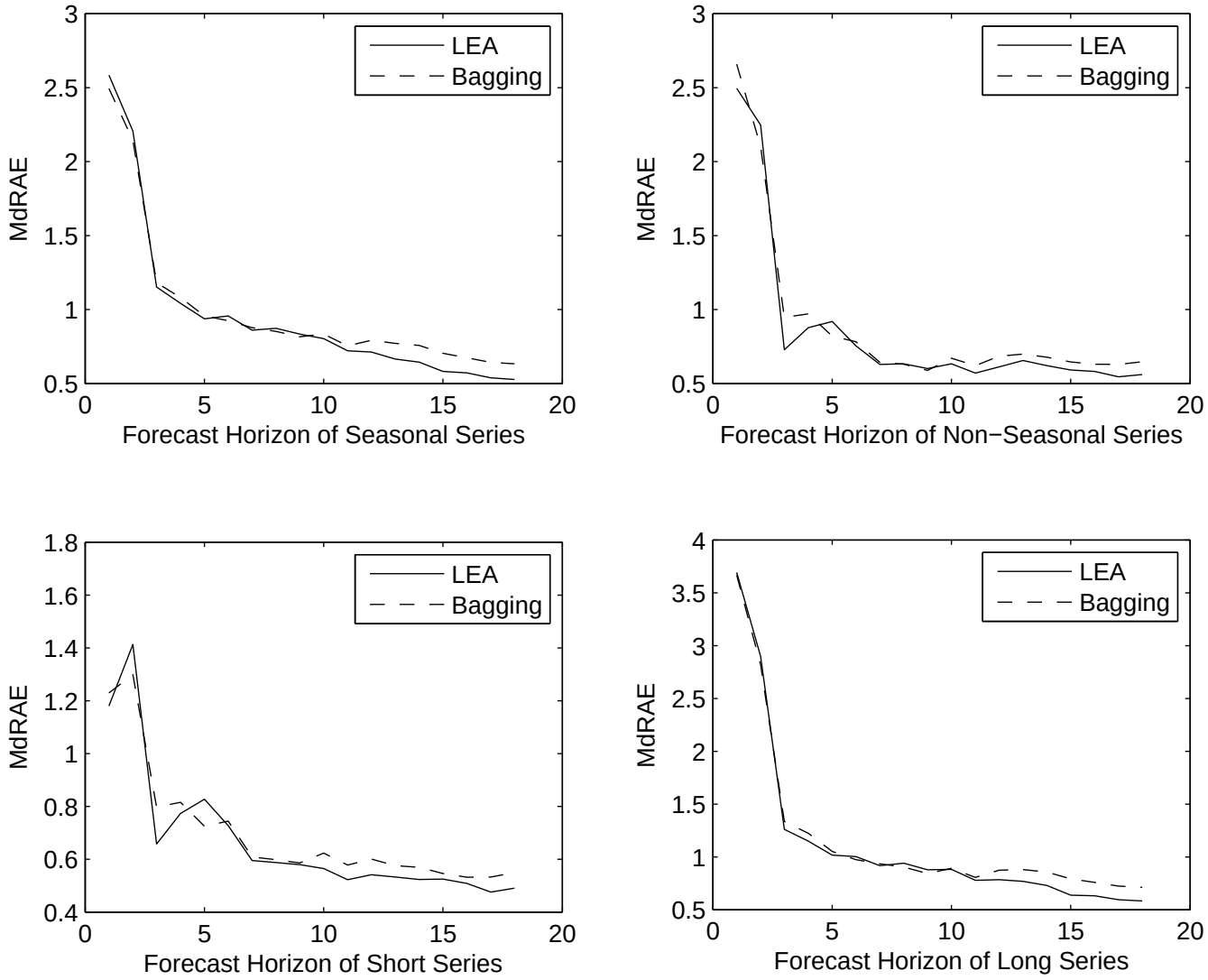

Figure 4.7: Comparison between basic bagging and layered bagging in terms of forecast horizon wise *MdRAE* for seasonal, non-seasonal, short and long time series data of NN3 [1] competition.

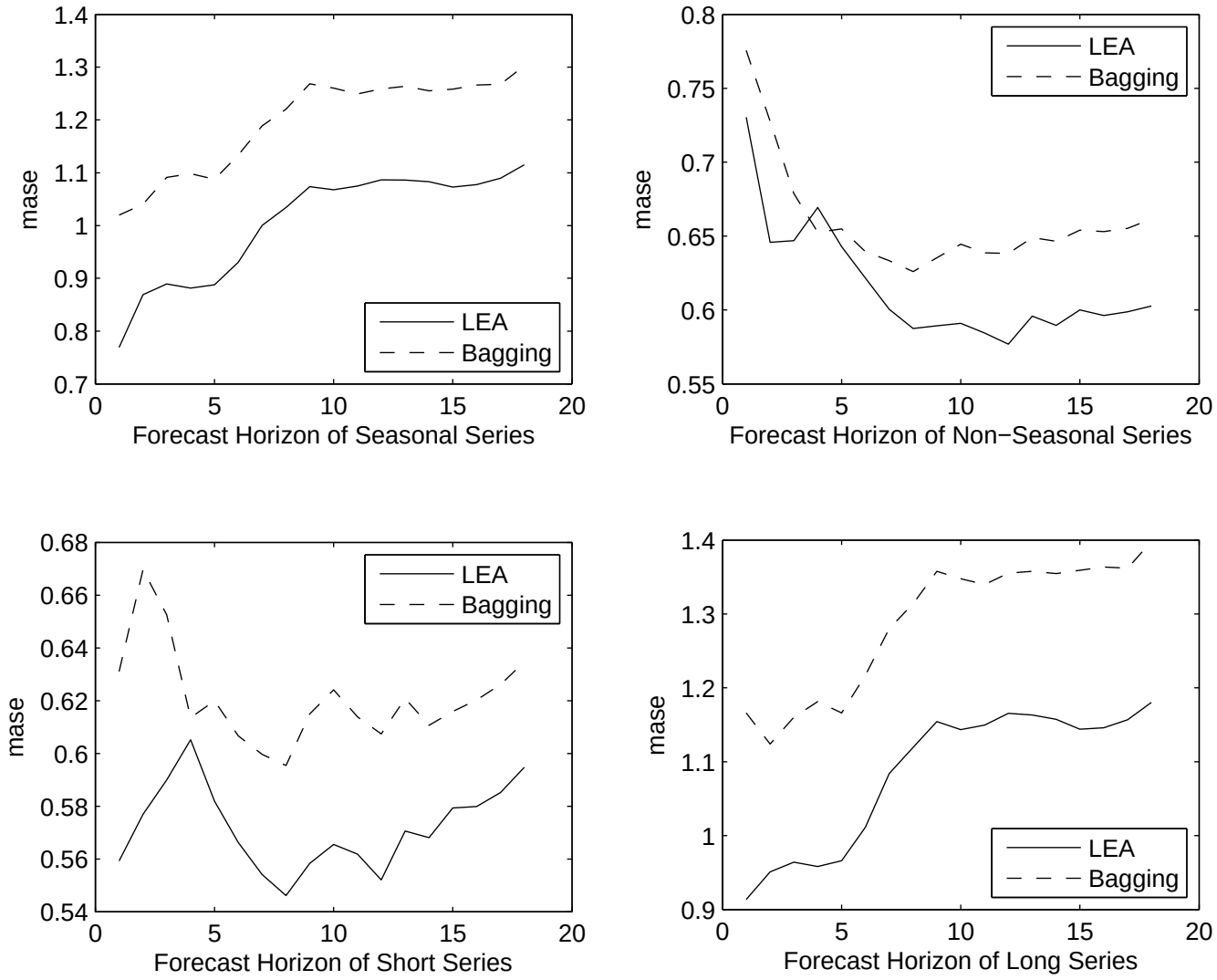

Figure 4.8: Comparison between basic bagging and layered bagging in terms of forecast horizon wise *MASE* for seasonal, non-seasonal, short and long time series data of NN3 [1] competition.

Table 4.7: Comparison between layered bagging and basic bagging in terms of average Bias, Variance and Co-variance decomposition for the time series data of NN3 [1] competition. Here the best result is highlighted using boldface text.

		E_{bias}	E_{var}	E_{cov}	E_{mse}
Layered Bagging	Seasonal	**0.0705**	**4.21E-04**	**0.0101**	**0.0810**
	Non-seasonal	**0.0513**	**4.40E-04**	**0.0055**	**0.0572**
	Short	**0.0538**	**8.55E-04**	**0.0317**	**0.0863**
	Long	**0.0419**	7.87E-04	**0.0276**	**0.0702**
Basic Bagging	Seasonal	0.1026	4.63E-04	0.0046	0.1076
	Non-seasonal	0.0874	3.00E-03	0.0376	0.1280
	Short	0.0735	1.20E-03	0.0476	0.1223
	Long	0.1157	**6.78E-04**	0.0112	0.1275

be obtained for other series.

Table 4.7 summarizes the results of the bias-variance-covariance decomposition of layered bagging and basic bagging. It can be observed from the table 4.7 that layered bagging provides less bias than basic bagging. Once again, the effectiveness of achieving the accurate lag from layer 1 of layered bagging is evident here. Apart from this, layered bagging also produces less variance and covariance in most of the cases than basic bagging. The positive effect of less bias, variance and co-variance is the less mean squared error, as shown in the last column of the table 4.7. Like *sMAPE*, *MdRAE* and *MASE*, layered bagging also defeats here basic bagging.

Disagreement and double fault

In the preceding section, we analyze ensembles produced by layered bagging and basic bagging algorithms based on bias, variance and co-variance decomposition. We here like to analyze these two algorithms based on disagreement and double fault. Although many diversity measures have been proposed in the context of classification problems, very little research has been done to measure diversity in case of ensembles applied on regression problems.

Disagreement is the ratio between the number of observations on which one predictor is correct and the other predictor is incorrect to the total number of observation. Double fault is defined as the portion of the cases that have been misclassified by both predictors. The disagreement ($D_{da\{m,n\}}$) and double fault ($D_{df\{m,n\}}$) between two predictors m and n can be expressed as

$$D_{da\{m,n\}} = \frac{N^{01} + N^{10}}{N^{11} + N^{01} + N^{10} + N^{00}} \tag{4.10}$$

$$D_{df\{m,n\}} = \frac{N^{00}}{N^{11} + N^{01} + N^{10} + N^{00}} \tag{4.11}$$

Let N be the number of instances, 1 denotes correct classification and 0 denotes incorrect classification. In Eqs. (4.10) and (4.11), N^{ij} denotes the number of examples that the first predictor m puts label i on a particular example, while the second predictor n puts label j on the same example. To use disagreement and double fault measures for TSF problems, we use the extension suggested in [38]. For each instance x, we calculate the standard deviation, σ, of the estimated target variable by all predictors in an ensemble. If the true value of the target is α, then a prediction β is considered to be correct if that $\beta \leq \alpha + \sigma$ and $\beta \geq \alpha - \sigma$ i.e., the prediction has to fall within a margin of one standard deviation of the value of the target variable. Otherwise, the prediction is considered as incorrect. We count the number of correct and incorrect predictions for each predictor over the data set and use the formula above to measure disagreement and double fault. From Eqs. (4.10) and (4.11), it is evident that a larger disagreement value indicates better diversity. In contrast, a larger double-fault value indicates worse diversity.

Table 4.8 summarizes the average result of disagreement and double fault for basic bagging and layered bagging for 111 time series. In our experiment there are 50 ensemble, so there in total 1326 pairwise diversity values for each of the time series. For each time series we simply take the average across all the pairs to calculate the final value of diversity. It can be observed that in terms of double fault, layered bagging is generating more diverse ensemble than basic bagging irrespective of the nature of the time series. Since layered bagging is trying to enforce accuracy among the members of an ensemble, it is obvious that the number of instances for which a pair of MLPs makes mistake will be less by layered bagging. This is the main reason for obtaining better double fault results by our method for all four different types of time series. In terms of disagreement measure, both basic bagging and layered bagging are better for two cases.

4.2.3 Comparison with Boosting

At each step of boosting, training data are reweighed in such a way that incorrectly classified examples get larger weights in a new training set. Hence, unlike bagging, we do not need to specify any data

Table 4.8: Comparison between basic bagging and layered bagging in terms of disagreement and double fault for seasonal, non-seasonal, short and long time series data of NN3 [1] competition. Here the best result is highlighted using boldface text.

	Basic Bagging		Layered Bagging	
	disagree	double fault	disagree	double fault
Seasonal	0.255	0.393	**0.276**	**0.362**
Non-seasonal	**0.352**	0.342	0.324	**0.316**
Short	**0.370**	0.390	0.325	**0.357**
Long	0.228	0.359	**0.270**	**0.334**

partition percentage here. To incorporate layering, we simply implement two layers of boosting and named this version as layered boosting, where one layer determine the best lag and the other layer uses it for forecasting.

Tables 4.9-4.12 and fig. 4.9-4.11 show the results of basic boosting and layered boosting. Following observations can be made from these tables and figures.

- Layered boosting outperforms basic boosting in terms of average, minimum and maximum *sMAPE*, *MASE* and *MdRAE*. This is true for four different types of time series we consider here (Tables 4.9 and 4.10). Wilcoxon signed rank test conducted between layered boosting and basic boosting indicates that the superiority of layered boosting over basic boosting is statistically significant (Table 4.12).

- Once again we see that a layered approach provides more stable result than a non-layered one. The standard deviation (std) from tables 4.9-4.10 of layered boosting establishes the fact.

- It can be observed from fig. 4.9 that layered boosting is better than basic boosting for all forecast horizons. Similar results are found for *MASE* and *MdRAE* as shown in fig. 4.10 and 4.11.

- We also calculate the win-loss count over *sMAPE*, *MASE* and *MdRAE* for the 111 time series. A glance at the table 4.11 again reveals that layered boosting is providing better performance than basic boosting irrespective of any performance measure.

Boosting algorithm focuses more on the difficult examples in the training set. Therefore, the number of unique training example decreases. It has been found that the performance of boosting is affected by the training set size and boosting is only be useful for large training sets [84]. Considering

Table 4.9: Comparison between basic boosting and layered boosting in terms of *sMAPE*, *MASE*, *MdRAE* for seasonal and non-seasonal time series data of NN3 [1] competition. Here the best result is highlighted using boldface text.

		Basic Boosting			Layered Boosting		
		sMAPE	MASE	MdRAE	sMAPE	MASE	MdRAE
seasonal	mean	22.980	2.131	1.169	**16.474**	**1.593**	**0.820**
	min	1.742	0.825	0.183	**1.482**	**0.579**	**0.153**
	max	76.102	13.046	4.543	**68.116**	**9.884**	**2.285**
	std	17.579	1.817	0.759	**13.671**	**1.511**	**0.528**
non seasonal	mean	29.718	1.165	1.337	**20.958**	**0.823**	**0.883**
	min	9.743	0.630	**0.052**	**6.068**	**0.361**	0.062
	max	115.230	2.576	3.514	**84.521**	**1.843**	**2.423**
	std	22.479	0.321	0.748	**14.800**	**0.239**	**0.464**

Table 4.10: Comparison between basic boosting and layered boosting in terms of *sMAPE*, *MASE*, *MdRAE* for short and long time series data of NN3 [1] competition. Here the best result is highlighted using boldface text.

		Basic Boosting			Layered Boosting		
		sMAPE	MASE	MdRAE	sMAPE	MASE	MdRAE
short	mean	25.225	1.204	1.322	**17.591**	**0.835**	**0.902**
	min	9.743	0.630	0.187	**6.068**	**0.361**	**0.182**
	max	95.814	2.493	3.514	**50.140**	**1.567**	**2.423**
	std	16.243	0.346	0.724	**9.697**	**0.202**	**0.478**
long	mean	25.890	2.209	1.163	**18.719**	**1.672**	**0.797**
	min	1.742	0.825	**0.052**	**1.482**	**0.579**	0.062
	max	115.230	13.046	4.543	**84.521**	**9.884**	**2.285**
	std	22.429	1.899	0.779	**17.131**	**1.580**	**0.522**

these facts and the average length of the time series of NN3 competition, it is expected that boosting algorithm will exhibit an inferior performance than bagging. In terms of average *sMAPE* over 111 time series, layered boosting and layered bagging achieve error rates of 18.21% and 14.42%, respectively, which establishes the facts more clearly. The same is true if we consider average *MASE* and *MdRAE*.

4.2.4 Comparison with Naïve forecast

In TSF literature (e.g. [27], [36]), naïve forecast is used as a benchmark model against which more sophisticated models can be compared. For time series data, the naïve forecast equals the previous period's actual value. In this section, we establish the evidence that LEA provides statistically

Table 4.11: Comparison between basic boosting and layered boosting in terms of Win-loss count for the time series data of NN3 [1] competition. Here the best result is highlighted using boldface text.

Performance Metrics	Number of Wins	
	Basic Boosting	Layered Boosting
sMAPE	5	**106**
MASE	8	**103**
MdRAE	14	**97**

Table 4.12: Wilcoxon Signed Rank Test summary between layered boosting and basic boosting for the time series data of NN3 [1] competition.

	Performance Metrics	R^+	R^-	p-value	hypothesis Significance level=0.05
Seasonal	sMAPE	2346	0	3.53E-12	Rejected for layered boosting
	MASE	2336	10	4.68E-12	Rejected for layered boosting
	MdRAE	2258	88	1.29E-10	Rejected for layered boosting
Non-seasonal	sMAPE	945	1	1.65E-08	Rejected for layered boosting
	MASE	936	10	2.73E-08	Rejected for layered boosting
	MdRAE	877	69	1.04E-07	Rejected for layered boosting
Short	sMAPE	1274	1	1.11E-09	Rejected for layered boosting
	MASE	1272	3	1.18E-09	Rejected for layered boosting
	MdRAE	1248	27	4.80E-09	Rejected for layered boosting
Long	sMAPE	1891	0	5.14E-11	Rejected for layered boosting
	MASE	1872	19	9.21E-11	Rejected for layered boosting
	MdRAE	1735	156	3.18E-09	Rejected for layered boosting

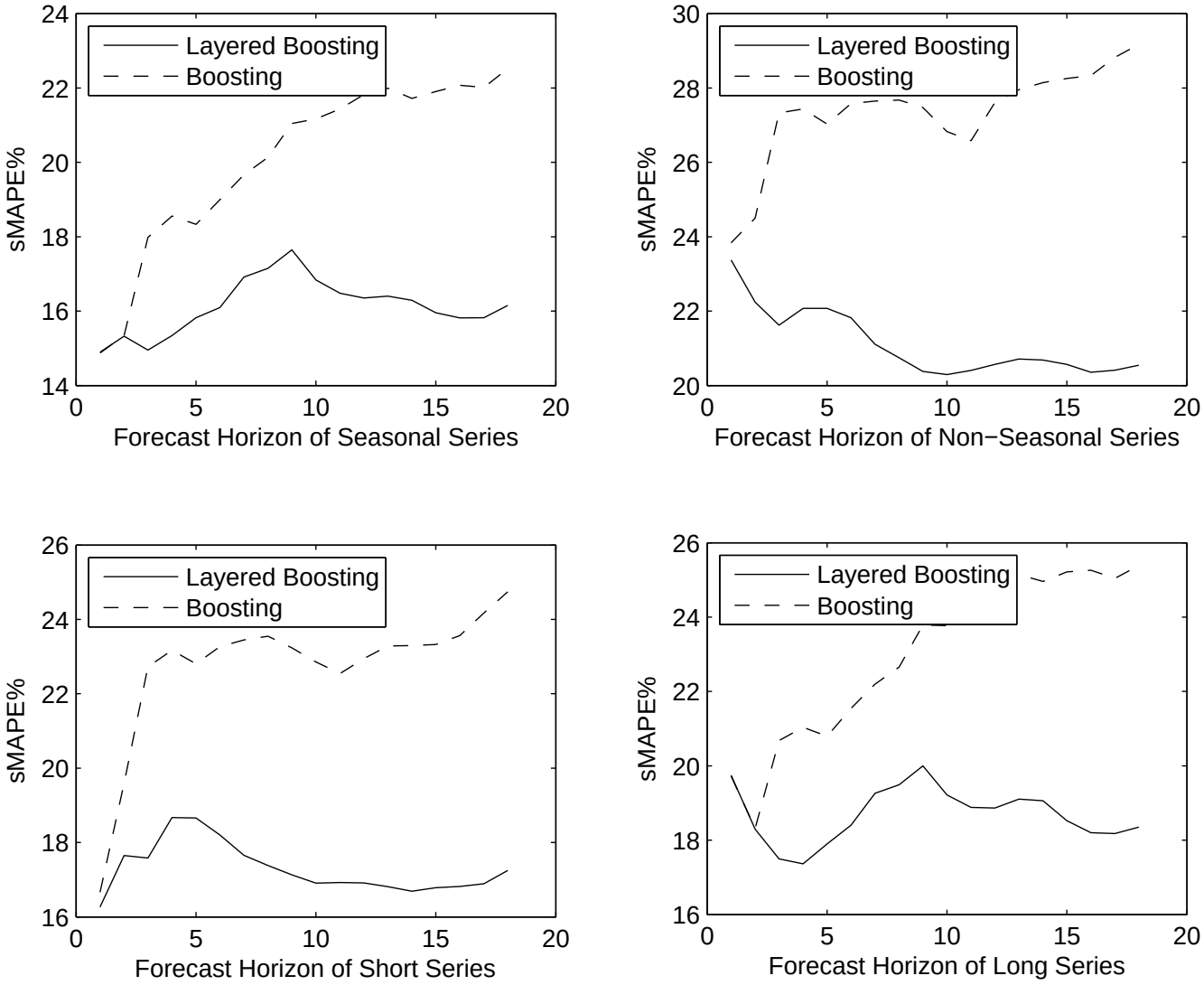

Figure 4.9: Comparison between basic boosting and layered boosting in terms of forecast horizon wise *sMAPE* for seasonal, non-seasonal, short and long time series data of NN3 [1] competition.

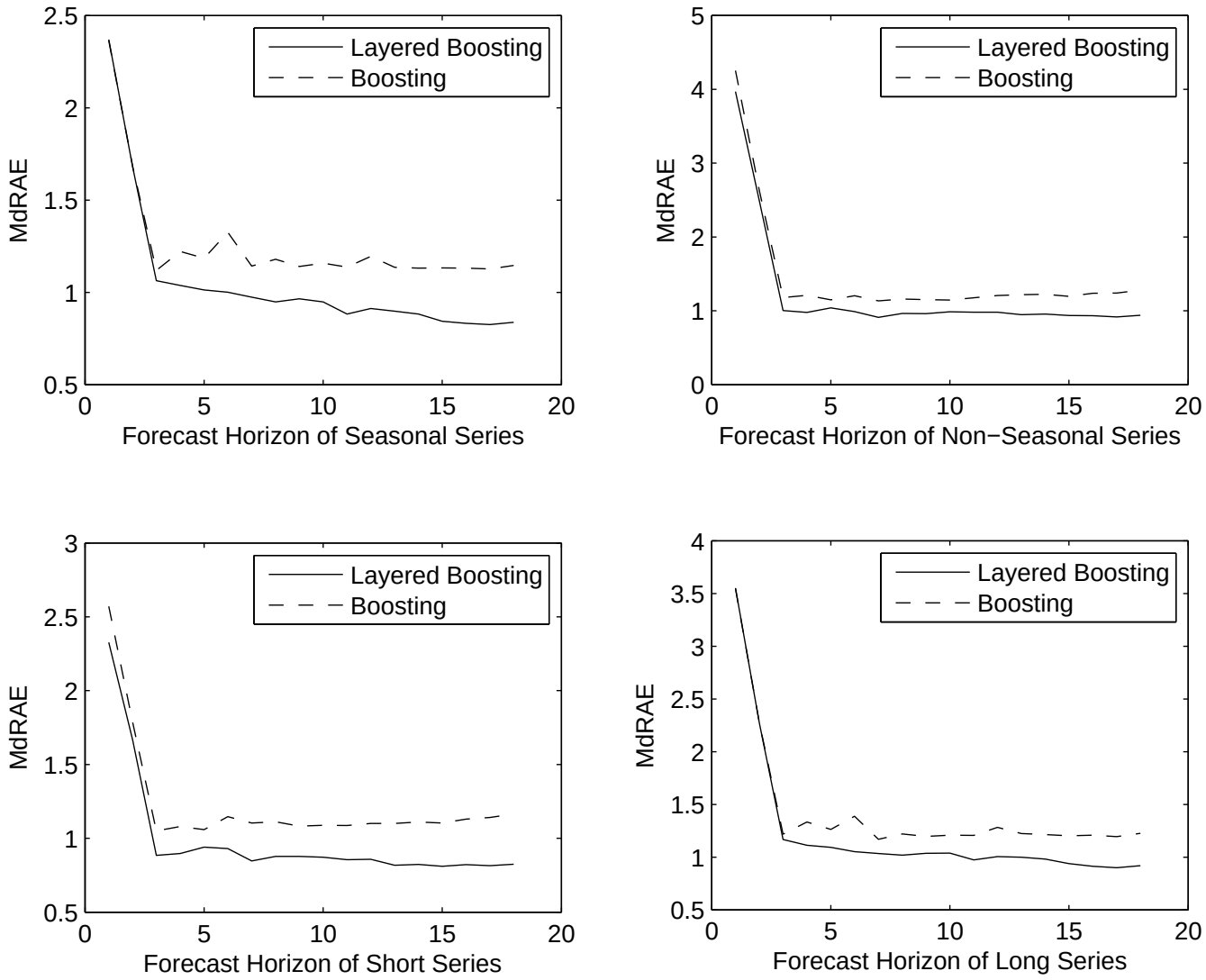

Figure 4.10: Comparison between basic boosting and layered boosting in terms of forecast horizon wise *MdRAE* for seasonal, non-seasonal, short and long time series data of NN3 [1] competition.

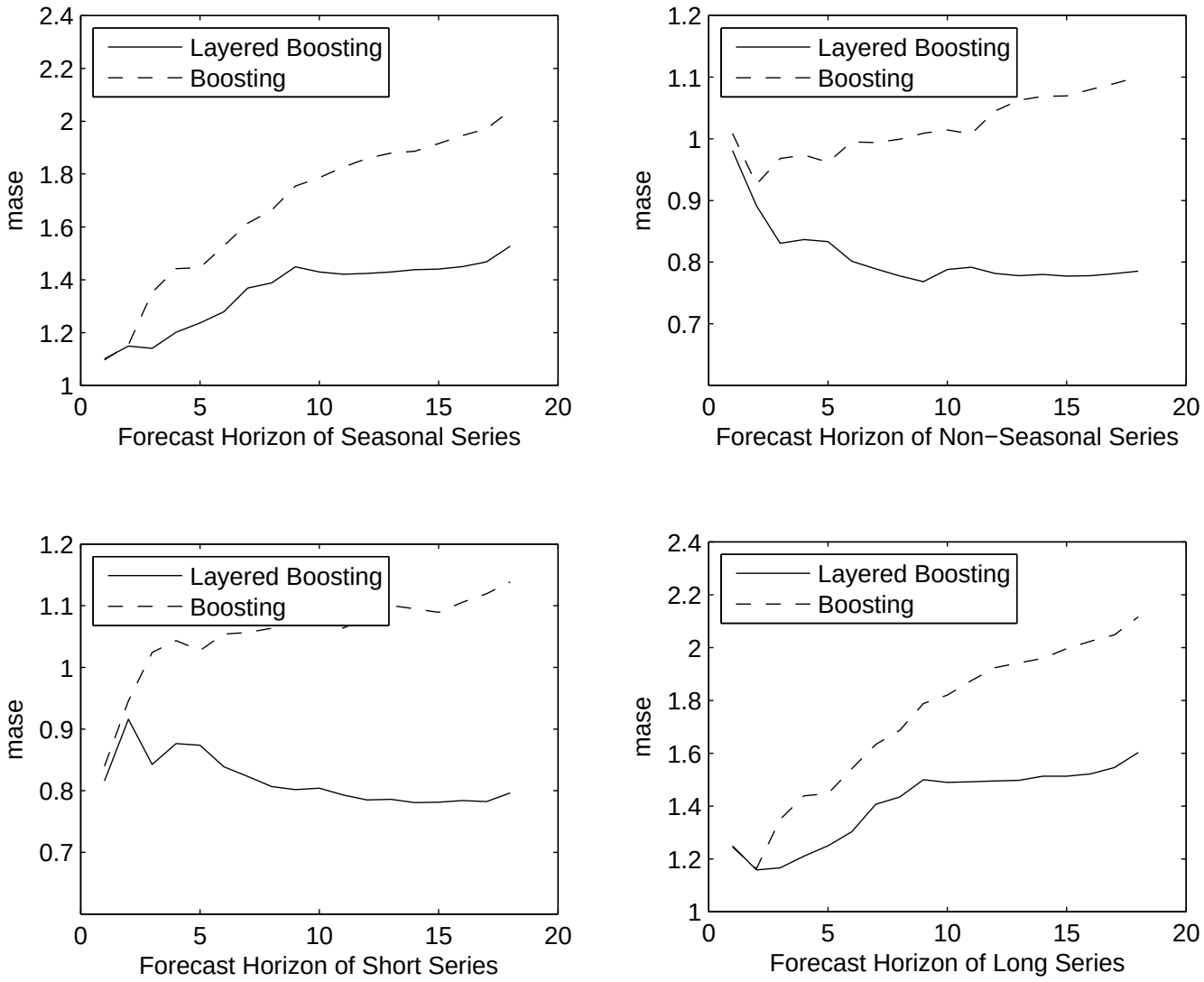

Figure 4.11: Comparison between basic boosting and layered boosting in terms of forecast horizon wise *MASE* for seasonal, non-seasonal, short and long time series data of NN3 [1] competition.

Table 4.13: Wilcoxon Signed Rank Test summary between layered bagging and naïve forecast for the time series data of NN3 [1] competition.

	Performance Metrics	R^+	R^-	p-value	hypothesis Significance level=0.05
Seasonal	sMAPE	2296	50	6.79E-12	Rejected for layered bagging
	MASE	2073	273	3.81E-08	Rejected for layered bagging
	MdRAE	2337	9	1.14E-12	Rejected for layered bagging
Non-seasonal	sMAPE	946	0	1.12E-08	Rejected for layered bagging
	MASE	884	62	6.95E-07	Rejected for layered bagging
	MdRAE	946	0	1.12E-08	Rejected for layered bagging
Short	sMAPE	1275	0	7.56E-10	Rejected for layered bagging
	MASE	1180	95	1.63E-07	Rejected for layered bagging
	MdRAE	1275	0	7.56E-10	Rejected for layered bagging
Long	sMAPE	1843	48	1.14E-10	Rejected for layered bagging
	MASE	1661	230	2.76E-07	Rejected for layered bagging
	MdRAE	1880	11	1.92E-11	Rejected for layered bagging

significant result than naïve forecast. We here use the same Wilcoxon signed rank test with per-series *sMAPE*, *MASE* and *MdRAE* for two different comparison, one layered bagging vs. naïve forecast and another is layered boosting vs. naïve forecast.

Tables 4.13 and 4.14 summarizes the results of the rank test. It can be seen that layered bagging is significantly different from the naïve forecast. In fact the associated p-values indicate that even a significance level of 0.001 would have resulted in the rejection of null hypothesis in favor of layered bagging for seasonal, non-seasonal, short and long time series irrespective of which performance metric is used. Layered boosting also provides significant difference than naïve forecast for any type of time series.

4.2.5 Comparative performance against different combining strategy

In ensemble literature, there are several different methods for combining the output of different member of ensemble. Majority voting, winner takes all, averaging are the most common methods for combining the output of ensemble for classification problem. In regression ensemble most of the literature suggested the averaging or weighted averaging as a strategic method for combination. In table 4.15 we present our analysis using two combination strategies namely averaging and our combination algorithm 3.1. It is clear from the table 4.15 *averaging* strategy is not a very good strategy

Table 4.14: Wilcoxon Signed Rank Test summary between layered boosting and naïve forecast for the time series data of NN3 [1] competition.

	Performance Metrics	R^+	R^-	p-value	hypothesis Significance level=0.05
Seasonal	sMAPE	1830	516	5.96E-05	Rejected for layered boosting
	MASE	1714	632	0.00094	Rejected for layered boosting
	MdRAE	1636	710	0.00466	Rejected for layered boosting
Non-seasonal	sMAPE	648	298	0.03459	Rejected for layered boosting
	MASE	768	178	0.00036	Rejected for layered boosting
	MdRAE	648	298	0.03459	Rejected for layered boosting
Short	sMAPE	993	282	0.000600	Rejected for layered boosting
	MASE	1124	151	2.65E-06	Rejected for layered boosting
	MdRAE	834	441	0.047284	Rejected for layered boosting
Long	sMAPE	1337	554	0.0049	Rejected for layered boosting
	MASE	1243	648	0.0326	Rejected for layered boosting
	MdRAE	1364	527	0.0026	Rejected for layered boosting

for TSF specially for NN3 [1] competition dataset. Out of 111 time series averaging wins only 9 times whereas the our combination algorithm 3.1 wins 102 times. Besides, for those 9 series, the difference of *sMAPE* between *averaging* and our combination algorithm 3.1 is very negligible. Furthermore, the average *sMAPE* using *averaging* is 23.27% whereas our combination algorithm 3.1 method provides a *sMAPE* of only 14.42%. So our combination algorithm 3.1 is 38.46% better than *averaging* which is justification why we have adopted this method in our proposed LEA scheme.

4.2.6 Comparative performance using different ensemble size

Determining the optimal size of the ensemble (i.e. number of NN in ensemble) is an important research issue. The size of the ensemble may vary depending on the nature of the dataset. Different ensemble size may result in a different performance for the same dataset because of the convergence issue and the unstable nature of the base predictors (i.e. MLPs). For example, in bagging algorithm, the number of base predictors should be selected by using trail and error for each application. It is common to use from 50 to 100 base predictors in the literature, but when an applications cannot handle this amount of processing time, this number must be better chosen.

Now considering, the large size and different types of time series data of NN3 competition [1], determining the optimal size of ensemble is a huge challenge. The size of ensemble may vary from one

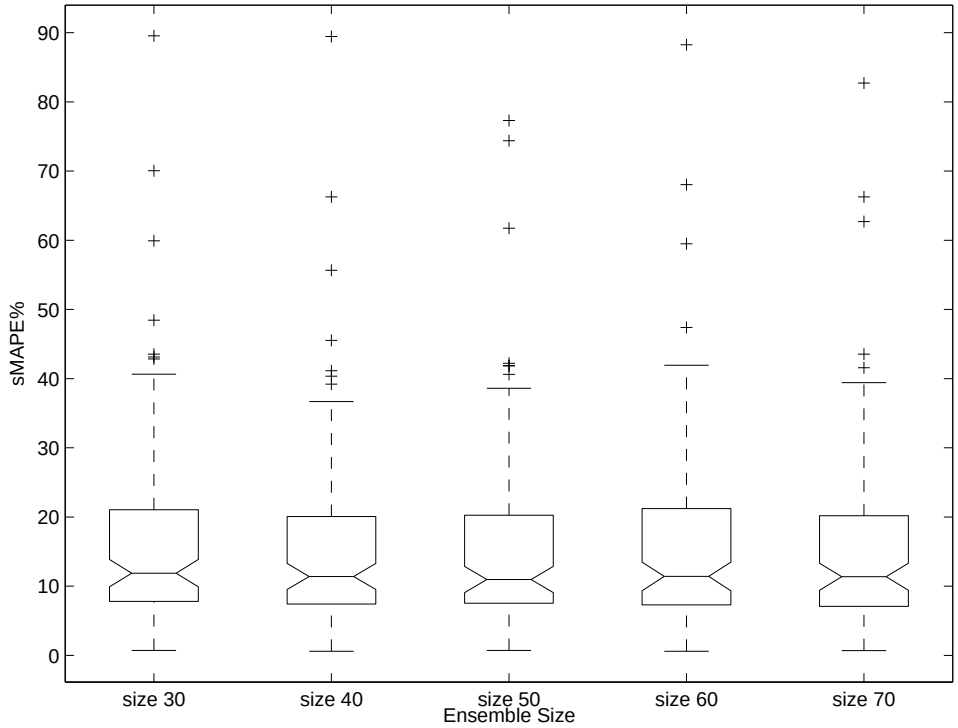

Figure 4.12: Boxplot Comparison of *sMAPE* of our proposed scheme using different ensemble size on the time series data of NN3 competition [1]

type of time series data to another type. The size of ensemble may also get changed from one time series to another. To determine the optimal size of ensemble for each of 111 time series of NN3 is computationally very expensive. Besides, another important requirement of NN3 competition is that for all the 111 time series the same experimental setup must be used. Since our main concern is to obtain better average *sMAPE* across all the 111 time series, to obtain the optimal size of ensemble we do not change it for every time series rather for all the 111 time series we keep the size of ensemble fixed. Now we vary the size of ensemble across all the 111 time series, for which we obtain the better forecast. Fig. 4.12 illustrates the result obtained using average *sMAPE* for different size of ensemble. From the Fig. 4.12 it is evident that increasing size of ensemble improves the performance. For example, the average *sMAPE* reduces from 16.47% to 14.42% when we increase the size of ensemble from 30 to 50. However, increasing the size beyond 50 is not upgrading the performance rather in some case giving inferior result which is due to the unstable nature of base predictor. So, for the sake of computational cost, we prefer to choose 50 is the optimal size of ensemble for the dataset of NN3 competition [1].

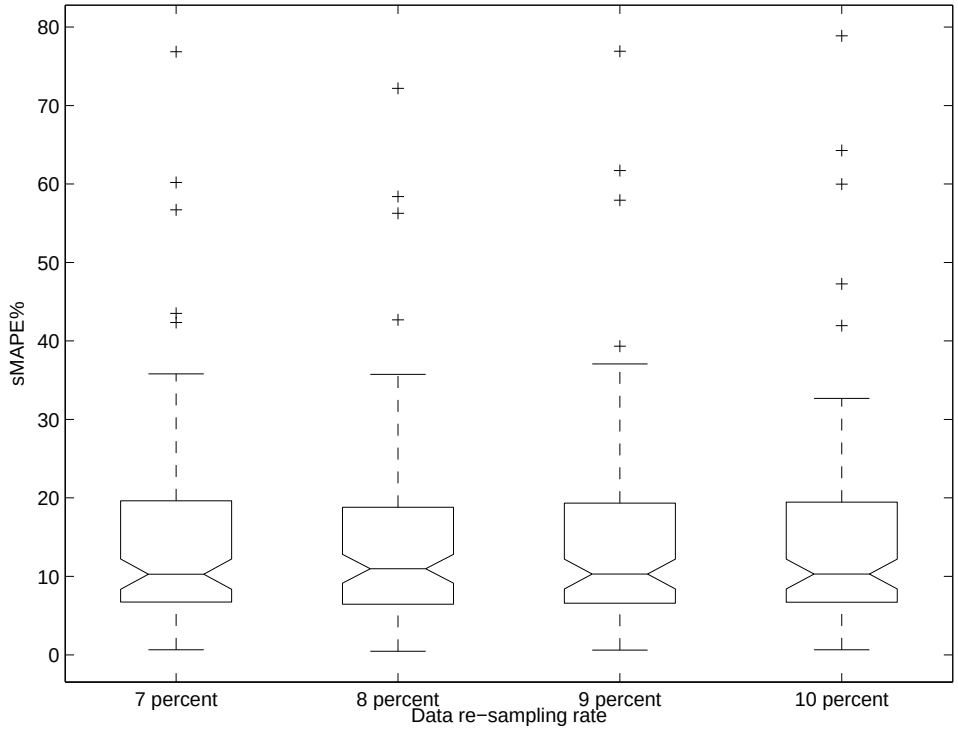

Figure 4.13: Boxplot Comparison of *sMAPE* of our proposed scheme using different data re-sampling rate on the time series data of NN3 competition [1]

4.2.7 Comparative performance using different data re-sampling rate

In ensemble layer 2 of LEA, we apply random re-sampling on the dataset D_{tr} to obtain a set of dataset to train each member of ensemble. Unlike basic bagging which uses a data re-sampling rate of 36.2% percentage, we apply here an adaptive strategy to obtain a best data re-sampling rate.

Since the most of the time series data of NN3 [1] contains data point between 69 to 150, a data re-sampling rate of 36.2% will cause a huge loss of correlation information among the time series data. So we start with a data re-sampling rate of 7% and increments it up to 10. Fig. 4.13 illustrates the average *sMAPE* obtains over 111 time series data of NN3 competition for the data re-sampling rate of 7%, 8%, 9% and 10% respectively. From the fig. 4.13 it is evident that a data re-sampling rate of 9% gives the best result and thats why we uses a data re-sampling rate of 9% in our experiment.

4.2.8 Discussion

This section briefly explains why the performance of LEA is better than other ensemble algorithms bagging and boosting for TSF problems we tested.

method	Our combination algorithm	Average
Number of Win	102	9
sMAPE	14.42	23.2

Table 4.15: Comparison between different combining strategies for the time series data of NN3 [1] competition.

LEA emphasizes both accuracy and diversity among individual NNs in an ensemble, while bagging and boosting primarily emphasizes diversity. LEA performs better than bagging in 85 out of 111 cases in terms of *sMAPE*. LEA obtains multiple decisions from a set of both diverse and accurate base predictors, whereas the bagging ensemble relies on the decision of an individual diverse classifier. This accounts for the better performance of the LEA. Similarly, LEA outperforms boosting in 106 out of 111 cases in terms of *sMAPE*. LEA identifies difficult-to-predict by layering. This is where the proposed approach takes the lead. Overall, in terms of *sMAPE* the proposed approach performs 7.91% better than bagging, and 20.11% better than boosting on NN3 [1] competition dataset.

LEA alleviates the problem of generating diverse ensemble in ensemble layer 1 using random lag which ensures different training set and architecture for MLPs in ensemble layer 1. Besides using the best lag from layer 1, LEA generates the optimal dataset on which bootstrapped sampling is applied using adaptive strategy so that the correlation between the time series data does not get lost which implicitly ensures the accuracy.

4.2.9 Comparison with other work

The NN3 competition attracts 59 submissions from computational intelligence (CI) based methods and statistical methods, making it the largest CI competition on time series. We choose the best five benchmarked statistical methods and the best five CI based methods for our comparison. The detail description of these methods can be found in [3]. Furthermore, we choose recently proposed ensembles of RBF networks by Yan [2] for comparison.

Table 4.16 presents the average results over 111 time series of layered bagging, Yan [2] and 10 other algorithms [3]. The model IDs with letter C as prefix stand for CI based models and those with letter B stand for statistical benchmark models. It can be observed from this table that layered bagging beats not only CI based methods but also the benchmarked statistical methods in terms of average

sMAPE, *MASE* and *MdRAE*. If we compare CI based methods (excluding layered bagging) with statistical methods, the best two CI based methods secure the third position and the fifth positions, while the other CI based methods appear at the bottom of the comparison table. This comparison indicates that the layered ensemble approach with proper techniques for maintaining accuracy and diversity is useful for obtaining good forecasting accuracy.

4.2.10 Further experiments and comparison

In order to show the predictive power of LEA, we choose to perform experiments on the NN5 time series data [4], another large benchmark time series dataset. The difficulties of this data set include outliers, missing values, multiple overlying seasonalities, etc. All these make the NN5 competition is one of the most interesting one. Like the NN3 competition, the NN5 provides two datasets: A and B. The dataset A contains 111 daily time series representing roughly two years of daily cash money withdrawal amounts (735 data points) by ATM machines at various cities in the UK., while the dataset B contains only 11 series taken from the dataset A. The challenge of the competition is to forecast the values of the next 56 days using the given historical data points. *sMAPE* is also adopted in the NN5 competition to declare the final winner.

Comparison with Bagging

In this section, we try to perform the same analysis for NN5 competition [4] that we have performed for NN3 competition [1]. Unlike, NN3 competition, the dataset of NN5 competition can be categorized into two groups namely seasonal and non-seasonal. There are 71 seasonal time series in NN5 competition whereas this number is 40 for non-seasonal time series.

Tables 4.17-4.19 show the results of basic bagging and layered bagging. Following observations can be made from these tables.

- Layered bagging again outperforms basic bagging in terms of average, minimum and maximum *sMAPE*, *MASE* and *MdRAE* with some exceptions. This is true for two different types of time series we consider here (Table 4.17). We also conduct the same Wilcoxon signed rank test here. The superiority of layered bagging over basic bagging is again evident here from the table 4.19.

- Stability of layered approach again is established here (Table 4.17). Layered bagging is providing more stable result than basic bagging in terms of *sMAPE* and *MdRAE*. However, in terms of

Table 4.16: Comparison among layered bagging, Yan [2] and 10 other methods [3] based on average *sMAPE*, *MASE* and *MdRAE*. Note that the results are average of 111 time series data of NN3 competition and '-' represents data are not available. Here the best result is highlighted using boldface text.

ID	Method	sMAPE	MdRAE	MASE
-	**Layered bagging**	**14.42**	**0.55**	**1.01**
B09	Wildi	14.84	0.82	1.13
B07	Theta	14.89	0.88	1.13
C27	Illies	15.18	0.84	1.25
B03	ForecastPro	15.44	0.89	1.17
-	Yan	15.80	-	-
B16	DES	15.90	0.94	1.17
B17	Comb S-H-D	15.93	0.90	1.21
B05	Autobox	15.95	0.93	1.18
C03	Flores	16.13	0.93	1.20
B14	SES	16.42	0.96	1.21
B15	HES	16.49	0.92	1.31
C46	Chen	16.55	0.94	1.34
C13	D'yakonov	16.57	0.91	1.26
B00	Automated ANN	16.81	0.91	1.21
C50	Kamel	16.92	0.90	1.28
B13	Njimi	17.05	0.96	1.34
C24	Abou-Nasr	17.54	1.02	1.43
C31	Theodosiou	17.62	0.96	1.24
B06	Census X12	17.78	0.92	1.29
B02	nMLP	17.84	0.97	2.03
C38	Adeodato	17.87	1.00	1.35
C26	de Vos	18.24	1.00	1.35
B01	nSVR	18.32	1.06	2.30
C44	Yan	18.58	1.06	1.37
C11	Perfilieva	18.62	0.93	1.57
C37	Duclos	18.68	0.99	1.30
C49	Schliebs	18.72	1.06	1.37
C59	Beliakov	18.73	1.00	1.36
C20	Kurogi	18.97	0.99	1.31
B10	Beadle	19.14	1.04	1.41
B11	Lewicke	19.17	1.03	1.43
C36	Sorjamaa	19.51	1.13	1.42
C15	Isa	20.00	1.12	1.53
C28	Eruhimov	20.19	1.13	1.50
C51	Papadaki	22.60	1.27	1.77
B04	Naive	22.69	1.00	1.48
B12	Hazarika	23.72	1.34	1.80
C17	Chang	24.09	1.35	1.81
C30	Pucheta	25.13	1.37	1.73
C57	Corzo	32.66	1.51	3.61

Table 4.17: Comparison between basic bagging and layered bagging in terms of *sMAPE*, *MASE* and *MdRAE* for seasonal and non-seasonal time series data of NN5 [4] competition. Here the best result is highlighted using boldface text.

		Basic Bagging			Layered Bagging		
		sMAPE	MASE	MdRAE	sMAPE	MASE	MdRAE
seasonal	mean	24.025	0.987	0.715	**19.993**	**0.982**	**0.705**
	min	**12.903**	0.695	0.224	13.090	**0.684**	**0.193**
	max	38.110	1.659	**1.079**	**31.302**	**1.651**	1.141
	std	5.498	**0.171**	0.204	**4.601**	0.174	**0.196**
non seasonal	mean	23.306	0.885	0.639	**19.528**	**0.880**	**0.639**
	min	15.357	0.618	0.233	**12.637**	**0.615**	**0.209**
	max	32.613	1.178	**0.962**	**26.505**	**1.157**	1.008
	std	4.279	**0.126**	0.199	**3.743**	0.128	**0.181**

Table 4.18: Comparison between basic bagging and layered bagging in terms of Win-loss count for the time series data of NN5 [4] competition. Here the best result is highlighted using boldface text.

Performance Metrics	Number of Wins	
	Basic Bagging	Layered Bagging
sMAPE	25	**86**
MASE	52	**59**
MdRAE	46	**65**

MASE basic bagging is marginally beating the layered bagging.

- The statistics using the win-loss count over *sMAPE*, *MASE* and *MdRAE* for the 111 time series of NN5 competition is depicted in the table 4.18. And like other performance measures basic bagging is providing inferior result than layered bagging.

Comparison with other work

The average *sMAPE* over 111 time series of layered bagging and 10 other algorithms of NN5 competition is furnished in the table 4.20. The result of other algorithms is compiled from [4]. As NN3 competition, the model IDs with letter C as prefix stand for CI based models and those with letter B stand for statistical benchmark models. From the table 4.20, we can observe that LEA is again beating all the CI and statistical benchmark based methods. It is to be noted that for NN5 dataset,

Table 4.19: Wilcoxon Signed Rank Test summary between layered bagging and basic bagging for the time series data of NN5 [4] competition.

	Performance Metrics	R^+	R^-	p-value	hypothesis Significance level=0.05
Seasonal	sMAPE	2484	1	3.72E-13	Rejected for layered bagging
	MASE	2485	0	0.0200	Rejected for layered bagging
	MdRAE	1511	700	0.0095	Rejected for layered bagging
Non-seasonal	sMAPE	861	0	2.48E-08	Rejected for layered bagging
	MASE	496	365	0.0395	Rejected for layered bagging
	MdRAE	455	406	0.02489	Rejected for layered bagging

Table 4.20: Comparison among layered bagging and 10 other methods [4] based on average *sMAPE*. Note that the results are average of 111 time series data of NN5 [4] competition and '-' represents data are not available. Here the best result is highlighted using boldface text.

ID	Method	sMAPE
-	**Layered bagging**	**19.82**
B02	Wildi	19.9
C23	Andrawis	20.4
C12	Vogel	20.5
C10	D'yakonov	20.6
B08	Noncheva	21.1
C06	Rauch	21.7
C19	Luna	21.8
B05	Lagoo	21.9
C01	Wichard	22.1
C17	Gao	22.3

we have used the same experimental configuration of NN3 except we change here the ensemble size to 100. We can also observe that the better performance of number of CI based methods compared to the NN3 competition, indicating that these methods are making improvement and will outperform statistical benchmark method very soon.

Chapter 5

Conclusion

Ensembles have been introduced to the machine learning community for nearly two decades. Many ensemble algorithms have been developed for classification problems. However, few algorithms exist that are developed for dealing with TSF problems. An important component of designing ensembles either for TSF (or classification) problems is the consideration of accuracy of the base predictors and diversity among the base predictors. Although most of the existing algorithms consider the diversity issue in designing ensemble for TSF, few attempts have been made in relation to accuracy. Most importantly, time series data have autocorrelation affect and the use of an appropriate lag is crucial for the accuracy of base predictors. Ensemble design still has to rely on either a tedious trial-and-error process or an human expert with rich prior knowledge about the lag parameter of a given series. In this thesis, we propose a layered ensemble architecture for efficiently forecasting time series data. Our layered architecture is consisted of two layers, each of which is an ensemble of neural networks.

Since lag parameter is vital for accurate forecast, LEA introduces a layer of NN ensemble to find out this information. Different training set using different lag are used to train the base predictors in ensemble layer 1 which eventually enforces diversity among them. However, to ensure accuracy in this layer, the only thing left to us is the use of a large number of NN (50 in the experiment of this thesis), and we can expect at least some of those come up with an optimal lag.

In ensemble layer 2 of the LEA, we already have the optimal lag from ensemble layer 1, so we have the quite "accurate" information about how many previous step will affect the prediction of the time series. Using this optimal lag, we are making sure all the member of the ensemble in the layer 2 is quite "accurate".

To introduce diversity in ensemble layer 2, LEA does not blindly change the architecture of the base predictors or randomly make some alternation in the dataset. In order to make a perfect forecast, it is necessary to handle those patterns, which might appear in the testing set, get more emphasized in the training set. LEA uses this important observation to incorporate diversity. Since only oracle has the knowledge which patterns will appear in the testing set, the only way around of this problem, is to randomly take some of the patterns, which appear in the training set and increase the probability of their appearance in the training of the base predictors. That's why during the training subset generation for the different member of the ensemble, LEA in this layer (layer 2) use the bagging to make random re-sampling of the optimal training set. But unlike bagging which randomly resample data points, LEA performs random re-sampling on the optimal lagged training set. So this technique serves the *three* important points for TSF using ensemble. *Firstly*, it is preserving the autocorrelation information of the time series. *Secondly*, it is increasing the probability to get better forecast, and *finally*, it is helping to generate a diverse ensemble.

The other important essence of LEA is that, it is completely independent of the architecture of the base predictors. Although in this thesis, we have used MLPs as base predictors, reader are encouraged to use any base predictor (Radial Basis Function, Elmen Neural Network etc).

Extensive experiments have been carried out in this thesis to evaluate how well LEA performed on different TSF problems in comparison with other ensemble and non-ensemble algorithms. In almost all cases, LEA was found better compared to popular ensemble algorithms, i.e., basic bagging, and basic boosting on different types of time series. LEA is providing an improvement of 9.08% in terms of *sMAPE* over basic bagging, whereas in terms of *MASE* and *MdRAE* the improvement is 13.04% and 15.64% respectively over basic bagging. LEA is also providing superior result than basic boosting where the improvement is 28.84%, 26.27% and 31.54% in terms of *sMAPE*, *MASE* and *MdRAE* respectively. These results indicate that the layering concept we introduce in this thesis can help to improve the forecasting accuracy of basic ensemble algorithms irrespective of the type of time series. When we compare LEA with other state-of-art statistical and CI methods, our algorithm was also found better. In fact LEA is improving the forecasting accuracy by 2.83% in terms of *sMAPE* against the top performer of NN3 [1] competition. For NN5 [4] competition the improvement is 0.40% in terms of *sMAPE*. Then, if we consider bias-variance-covariance decomposition, we will find that LEA is generating an ensemble system with less bias, variance and covariance than basic bagging and as a result LEA is exhibiting less *MSE* on the test set than basic bagging. Finally, considering the

diversity we find that LEA is also generating a reasonable amount of diverse members which helps to reduce the generalization error on the test set.

5.1 Future direction

Although LEA has performed very well for almost all TSF problems we tested, our experimental study appeared to have revealed a weakness of LEA in dealing with time series number 103 of NN3 [1] competition dataset, compared to other time series. The reasons for such performance is due to improper selection of ensemble size and parameters associated with the ensemble members. These and some other issues can be considered for future research, which are described as follows.

- **Parameter optimization:** Current implementation of LEA, obtains the parameter of MLPs using a trial and error basis rule. However, there exists several parameter optimization algorithm in the literature, using which might help to find a better parameter set for MLPs which ultimately helps us to find more accurate base predictors in both layers.
- **Data sampling in both layer:** LEA applies the bagging (i.e. bootstrapped sampling) in the ensemble layer 2 in order to enforce diversity. However, it might be a research question, whether we should apply the bagging in the layer 1 also. What kind of benefit we can achieve if we apply this in ensemble layer 1?
- **Varying the architecture of the base predictors in both layers:** In our current implementation of LEA, we vary the architecture of NNs in ensemble layer 1 only by using different randomly lagged training set. So, basically we are enforcing diversity in ensemble layer 1 by using variation in both training set and architecture. But in ensemble layer 2 we only vary the training set. Further experimental analysis should be performed to investigate whether maximizing the diversity using the variation on both training set and architecture really improves the performance of LEA.
- **Number of base predictors in the ensemble:** Our experimental analysis already provides the information that increasing the number of NNs in ensemble improves the performance. For a given time series data, a system is said to be fully automated, if it can determine the required number of base predictors necessary for better forecast rather than using the user defined input. So any future extension of LEA, should try to incorporate this feature.

- **Combining the output of the ensemble:** Although there exists several combining method for classification problem, for TSF there exists very little. In fact most of the existing methods are direct extension of their classification part. So extensive research efforts must be applied here for better forecast of time series.

- Apart from the above discussed phenomenon, more real-world time-series prediction problems have to be performed to further ascertain the applicability of LEA. The results have to be compared not only with those from NN based systems but also other established statistical techniques that have been widely used in undertaking TSF problems. Research efforts should also be devoted to the methods that can further reduce the correlation effect of the time series data in combining neural networks. Simulation and experimental design methodology should prove useful and necessary in these endeavors.

Bibliography

[1] NN3 competition. http://www.neural-forecasting-competition.com/index.htm, 2008.

[2] W. Yan. Toward automatic time-series forecasting using neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 23(7):1028–1039, 2012.

[3] Sven F Crone, Michele Hibon, and Konstantinos Nikolopoulos. Advances in forecasting with neural networks, empirical evidence from the nn3 competition on time series prediction. *International Journal of Forecasting*, 27(3):635–660, 2011.

[4] *NN5 competition.* URL: http://www.neural-forecasting-competition.com/index.htm.

[5] Justin Wolfers and Eric Zitzewitz. Prediction markets. Technical report, National Bureau of Economic Research, 2004.

[6] James W Taylor and Roberto Buizza. Neural network load forecasting with weather ensemble predictions. *IEEE Transactions on Power Systems*, 17(3):626–632, 2002.

[7] Dong C Park, MA El-Sharkawi, RJ Marks, LE Atlas, MJ Damborg, et al. Electric load forecasting using an artificial neural network. *IEEE Transactions on Power Systems*, 6(2):442–449, 1991.

[8] Thomas H Naylor, Terry G Seaks, and Dean W Wichern. Box-jenkins methods: An alternative to econometric models. *International Statistical Review/Revue Internationale de Statistique*, pages 123–137, 1972.

[9] Javier Contreras, Rosario Espinola, Francisco J Nogales, and Antonio J Conejo. Arima models to predict next-day electricity prices. *IEEE Transactions on Power Systems*, 18(3):1014–1020, 2003.

[10] Paulo Cortez, Miguel Rocha, Fernando Sollari Allegro, and José Neves. Real-time forecasting by bio-inspired models. *Artificial Intelligence and Applications*, pages 52–57, 2002.

[11] Paulo Cortez, Miguel Rocha, and José Neves. Evolving time series forecasting arma models. *Journal of Heuristics*, 10(4):415–429, 2004.

[12] Robert N Miller and Laura L Ehret. Ensemble generation for models of multimodal systems. *Monthly weather review*, 130(9):2313–2333, 2002.

[13] John G Carney and Pádraig Cunningham. The neuralbag algorithm: Optimizing generalization performance in bagged neural networks. In *Proceedings of the 7th European Symposium on Artificial Neural Networks*, pages 35–40. D-Facto: Bruges, Belgium, 1999.

[14] H. Chen and X. Yao. Ensemble regression trees for time series predicitions. *methods*, 11:6, 2007.

[15] I. Maqsood, M.R. Khan, and A. Abraham. An ensemble of neural networks for weather forecasting. *Neural Computing & Applications*, 13(2):112–122, 2004.

[16] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[17] R.E. Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[18] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[19] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[20] Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

[21] T.K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[22] Yong Liu and Xin Yao. Ensemble learning via negative correlation. *Neural Networks*, 12(10):1399–1404, 1999.

[23] Gavin Brown, Jeremy Wyatt, Rachel Harris, and Xin Yao. Diversity creation methods: a survey and categorisation. *Information Fusion*, 6(1):5–20, 2005.

[24] Giorgio Fumera, Fabio Roli, and Alessandra Serrau. A theoretical analysis of bagging as a linear combination of classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(7):1293–1299, 2008.

[25] Y. Freund, R.E. Schapire, et al. Experiments with a new boosting algorithm. In *Thirteen International Conference on Machine Learning*, pages 148–156, 1996.

[26] LI Kuncheva, Fabio Roli, Gian Luca Marcialis, and Catherine A Shipp. Complexity of data subsets generated by the random subspace method: an experimental investigation. In *Multiple Classifier Systems*, pages 349–358. Springer, 2001.

[27] Allan Timmermann. Forecast combinations. *Handbook of economic forecasting*, 1:135–196, 2006.

[28] Pilar Poncela, Julio Rodríguez, Rocío Sánchez-Mangas, and Eva Senra. Forecast combination through dimension reduction techniques. *International Journal of Forecasting*, 27(2):224–237, 2011.

[29] 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.

[30] Christiane Lemke and Bogdan Gabrys. Meta-learning for time series forecasting and forecast combination. *Neurocomputing*, 73(10):2006–2016, 2010.

[31] Carson E Agnew. Bayesian consensus forecasts of macroeconomic variables. *Journal of Forecasting*, 4(4):363–376, 1985.

[32] James H Stock and Mark W Watson. Combination forecasts of output growth in a seven-country data set. *Journal of Forecasting*, 23(6):405–430, 2004.

[33] Celal Aksu and Sevket I Gunter. An empirical analysis of the accuracy of sa, ols, erls and nrls combination forecasts. *International Journal of Forecasting*, 8(1):27–43, 1992.

[34] Philip Hans Franses. Model selection for forecast combination. *Applied Economics*, 43(14):1721–1727, 2011.

[35] Robert R Andrawis, Amir F Atiya, and Hisham El-Shishiny. Forecast combinations of computational intelligence and linear models for the nn5 time series forecasting competition. *International Journal of Forecasting*, 27(3):672–688, 2011.

[36] Jon Scott Armstrong. *Principles of forecasting: a handbook for researchers and practitioners*, volume 30. Springer, 2001.

[37] Agus Widodo and Indra Budi. Combination of time series forecasts using neural network. In *Electrical Engineering and Informatics (ICEEI), 2011 International Conference on*, pages 1–6. IEEE, 2011.

[38] Haimonti Dutta. Measuring diversity in regression ensembles. In *IICAI*, volume 9, page 17p, 2009.

[39] Ludmila I Kuncheva. Switching between selection and fusion in combining classifiers: An experiment. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 32(2):146–156, 2002.

[40] David H Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390, 1996.

[41] Thomas G Dietterich. Ensemble methods in machine learning. In *Multiple classifier systems*, pages 1–15. Springer, 2000.

[42] Fabricio A Breve, Moacir P Ponti-Junior, and Nelson DA Mascarenhas. Multilayer perceptron classifier combination for identification of materials on noisy soil science multispectral images. In *Computer Graphics and Image Processing, 2007. SIBGRAPI 2007. XX Brazilian Symposium on*, pages 239–244. IEEE, 2007.

[43] Guoqiang Zhang. *Linear and nonlinear time series forecasting with artificial neural networks*. PhD thesis, Kent, OH, USA, 1998.

[44] Cagdas Hakan Aladag and Erol Egrioglu. *Advances in Time Series Forecasting*. Bentham Science Publishers, 2012.

[45] Robert T Clemen. Combining forecasts: A review and annotated bibliography. *International Journal of Forecasting*, 5(4):559–583, 1989.

[46] Ran Avnimelech and Nathan Intrator. Boosting regression estimators. *Neural computation*, 11(2):499–520, 1999.

[47] C.P. Lim and W.Y. Goh. The application of an ensemble of boosted elman networks to time series prediction: A benchmark study. *International Journal of Computational Intelligence*, 3(2), 2005.

[48] W.Y. Goh, C.P. Lim, and K.K. 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.

[49] M. Assaad, R. Boné, and H. Cardot. A new boosting algorithm for improved time-series forecasting with recurrent neural networks. *Information Fusion*, 9(1):41–55, 2008.

[50] John R Koza, Forrest H Bennett III, and Oscar Stiffelman. *Genetic programming as a Darwinian invention machine*. Springer, 1999.

[51] Gregory Paris, Denis Robilliard, and Cyril Fonlupt. Applying boosting techniques to genetic programming. In *Artificial evolution*, pages 267–278. Springer, 2002.

[52] Luzia Vidal de Souza, Aurora TR Pozo, Joel MC da Rosa, and Anselmo Chaves Neto. The boosting technique using correlation coefficient to improve time series forecasting accuracy. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1288–1295. IEEE, 2007.

[53] Zhuo Zheng. Boosting and bagging of neural networks with applications to financial time series. Technical report, Working paper, Department of Statistics, University of Chicago, 2006.

[54] Shuichi Kurogi, Ryohei Koyama, Shinya Tanaka, and Toshihisa Sanuki. Forecasting using first-order difference of time series and bagging of competitive associative nets. In *International Joint Conference on Neural Networks (IJCNN)*, pages 166–171. IEEE, 2007.

[55] J.D. Wichard and M. Ogorzalek. Time series prediction with ensemble models. In *IEEE International Joint Conference on Neural Networks*, volume 2, pages 1625–1630. IEEE, 2004.

[56] Kin Keung Lai, Lean Yu, Shouyang Wang, and Huang Wei. A novel nonlinear neural network ensemble model for financial time series forecasting. In *Computational Science–ICCS 2006*, pages 790–793. Springer, 2006.

[57] Bo Qian and Khaled Rasheed. Stock market prediction with multiple classifiers. *Applied Intelligence*, 26(1):25–33, 2007.

[58] I Ilies, H Jaeger, O Kosuchinas, M Rincon, V Šakėnas, and N Vaškevičius. Stepping forward through echoes of the past: forecasting with echo state networks. short report on the winning entry to the nn3 financial forecasting competition (2007).

[59] D. Ruta, B. Gabrys, and C. Lemke. A generic multilevel architecture for time series prediction. *IEEE Transactions on Knowledge and Data Engineering*, 23(3):350–359, 2011.

[60] GP Zhang and VL Berardi. Time series forecasting with neural network ensembles: an application for exchange rate prediction. *Journal of the Operational Research Society*, pages 652–664, 2001.

[61] Giorgio Giacinto and Fabio Roli. An approach to the automatic design of multiple classifier systems. *Pattern recognition letters*, 22(1):25–33, 2001.

[62] Md Monirul Islam, Xin Yao, SM Shahriar Nirjon, Muhammad Asiful Islam, and Kazuyuki Murase. Bagging and boosting negatively correlated neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 38(3):771–784, 2008.

[63] Md M Islam, Xin Yao, and Kazuyuki Murase. A constructive algorithm for training cooperative neural network ensembles. *IEEE Transactions on Neural Networks*, 14(4):820–834, 2003.

[64] Amanda JC Sharkey and Noel E Sharkey. Combining diverse neural nets. *Knowledge Engineering Review*, 12(3):231–247, 1997.

[65] A Gilchrist. Long-range forecasting. *Quarterly Journal of the Royal Meteorological Society*, 112(473):567–592, 1986.

[66] Robert R Andrawis and Amir F Atiya. A new bayesian formulation for holt's exponential smoothing. *Journal of Forecasting*, 28(3):218–234, 2009.

[67] M. Adya, F. Collopy, J.S. Armstrong, and M. Kennedy. Automatic identification of time series features for rule-based forecasting. *International Journal of Forecasting*, 17(2):143–157, 2001.

[68] Jörg D Wichard. Forecasting the nn5 time series with hybrid models. *International Journal of Forecasting*, 27(3):700–707, 2011.

[69] Monica Adya and Fred Collopy. How effective are neural networks at forecasting and prediction? a review and evaluation. *J. Forecasting*, 17:481–495, 1998.

[70] Dymitr Ruta and Bogdan Gabrys. Classifier selection for majority voting. *Information fusion*, 6(1):63–81, 2005.

[71] Giorgio Giacinto and Fabio Roli. Design of effective neural network ensembles for image classification purposes. *Image and Vision Computing*, 19(9):699–707, 2001.

[72] Anders Krogh, Jesper Vedelsby, et al. Neural network ensembles, cross validation, and active learning. *Advances in neural information processing systems*, pages 231–238, 1995.

[73] Sherif Hashem. Optimal linear combinations of neural networks. *Neural networks*, 10(4):599–614, 1997.

[74] Rob J Hyndman and Anne B Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688, 2006.

[75] Time series forecasting competition for neural networks and computational intelligence. URL: http://www.neural-forecasting-competition.com/index.htm.

[76] J Scott Armstrong and Fred Collopy. Error measures for generalizing about forecasting methods: Empirical comparisons. *International Journal of Forecasting*, 8(1):69–80, 1992.

[77] Patrick A Thompson. An mse statistic for comparing forecast accuracy across series. *International Journal of Forecasting*, 6(2):219–227, 1990.

[78] Martin Stepnicka, Juan Peralta Donate, Paulo Cortez, Lenka Vavrickova, and German Gutierrez Sanchez. Forecasting seasonal time series with computational intelligence: contribution of a combination of distinct methods. 2011.

[79] Johannes Mager, Ulrich Paasche, and Bernhard Sick. Forecasting financial time series with support vector machines based on dynamic kernels. In *IEEE Conference on Soft Computing in Industrial Applications (SMCia'08).*, pages 252–257. IEEE, 2008.

[80] Marina Theodosiou. Forecasting issues: Ideas of decomposition and combination. Technical report, 2010.

[81] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83, 1945.

[82] Yong Liu and Xin Yao. Simultaneous training of negatively correlated neural networks in an ensemble. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 29(6):716–725, 1999.

[83] Robert A Jacobs. Bias/variance analyses of mixtures-of-experts architectures. *Neural computation*, 9(2):369–383, 1997.

[84] Marina Skurichina and Robert PW Duin. Bagging, boosting and the random subspace method for linear classifiers. *Pattern Analysis & Applications*, 5(2):121–135, 2002.