TIME AND CONTEXT SENSITIVE OPTIMIZATION OF MACHINE LEARNING MODELS FOR SEQUENTIAL DATA PREDICTION

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF

MASTER OF SCIENCE

IN

ELECTRICAL AND ELECTRONICS ENGINEERING

By

Arda Fazla

July 2024

Time and Context Sensitive Optimization of Machine Learning Models
for Sequential Data Prediction
By Arda Fazla
July 2024

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Süleyman Serdar Kozat(Advisor)

Sinan Gezici

Ramazan Gökberk Cinbiş

Approved for the Graduate School of Engineering and Science:

Orhan Arıkan
Director of the Graduate School

ABSTRACT

TIME AND CONTEXT SENSITIVE OPTIMIZATION OF MACHINE LEARNING MODELS FOR SEQUENTIAL DATA PREDICTION

Arda Fazla
M.S. in Electrical and Electronics Engineering
Advisor: Süleyman Serdar Kozat
July 2024

We investigate the nonlinear prediction of sequential time series data through the mixture/combination of machine learning models. First, we introduce a novel ensemble learning approach that effectively combines multiple base learners in a time-aware and context-sensitive manner. This process involves a weight optimization problem targeting a specific loss function while considering (non)convex constraints on the linear combination of base learners. These constraints are theoretically analyzed under known statistics and are automatically incorporated into the meta-learner as part of the optimization process during training. Next, we introduce a direct two-stage approach based on the combination of linear and nonlinear models, where we jointly optimize the parameters of both models to minimize the final regression error. By this joint optimization, we alleviate the well-known underfitting and overfitting problems in modeling sequential data. As the linear model, we use a traditional linear time series forecasting model (SARIMAX) and as the nonlinear model, we use boosted soft decision trees (Soft GBDT). For both of our approaches, we illustrate notable performance improvements on real-life data and well-known competition datasets compared to traditional ensemble/mixture techniques and state-of-the-art forecasting models in the machine learning literature. Additionally, we make the source code of both of our approaches publicly available to facilitate further research and comparison.

Keywords: ensemble learning, prediction/regression, time series, stochastic gradient descent (SGD), online learning, artificial neural network (ANN), light gradient boosting machine (LightGBM), seasonal auto-regressive integrated moving average with eXogenous factors (SARIMAX), soft gradient boosting decision tree (Soft GBDT).

ÖZET

MAKİNE ÖĞRENİMİ MODELLERİNİN SIRALI VERİ TAHMİNİ İÇİN ZAMAN VE BAĞLAM DUYARLI OPTİMİZASYONU

Arda Fazla
Elektrik ve Elektronik Mühendisliği, Yüksek Lisans
Tez Danışmanı: Süleyman Serdar Kozat
Temmuz 2024

Gerçek hayattaki zamansal serilerin, makine öğrenimi modellerinin ortak kombinasyonu yoluyla, doğrusal olmayacak şekilde tahminlenmesini araştırıyoruz. Bu bağlamda ilk olarak, zaman bilincine ve ortam duyarlılığına sahip yeni bir birlikte öğrenme metodu tanıtıyoruz. Bu süreç, belli bir kayıp fonksiyonunu hedef alan bir ağırlık optimizasyon problemi içerirken aynı zamanda temel öğrenicilerin doğrusal kombinasyonu üzerindeki kısıtlamaları da göz önünde bulunduruyor. Bu kısıtlamaları teorik olarak analiz ediyor ve birlikte öğrenim sırasında optimizasyon sürecinin bir parçası olarak ana öğreniciye otomatik olarak entegre ediyoruz. İkinci yöntemimiz olarak, makine öğrenimi modellerinin ortak optimizasyonu yoluyla eğitilmesini baz alarak, doğrusal ve doğrusal olmayan modellerin bir kombinasyonunu tanıtıyoruz. Burada gerçekleştirilen eğitim sırasında, her iki modelin parametrelerini son regresyon hatasını en aza indirecek şekilde ortaklaşa optimize ediyoruz. Bu ortak optimizasyon metodunu kullanarak, zamansal serilerin modellemesinde sıkça karşılaşılan yetersiz uyum ve ezberleme sorularının önüne geçiyoruz. Doğrusal modelimiz olarak geleneksel bir zamansal seri tahminleme modeli ve doğrusal olmayan modelimiz olarak ise gradyan destekli yumuşak karar ağaçları kullanıyoruz. Tanıttığımız bu iki yeni yöntemi kullanarak, gerçek hayat verileri ve bilinen yarışma veri setleri üzerinde, geleneksel birlikte öğrenme yöntemleri ve literatürdeki en iyi bilinen makine öğrenimi tahmin modellerine kıyasla belirgin performans iyileştirmeleri sergiliyoruz. Ayrıca, bu konu üzerinden ileriye yönelik araştırma ve karşılaştırma yapılmasını kolaylaştırmak amacıyla yöntemlerimizin kaynak kodlarını paylaşıyoruz.

Anahtar sözcükler: birlikte öğrenme, tahminleme, olasılıksal gradyan inişi, çevrimiçi öğrenme, yapay sinir ağı, gradyan destekli hafif makine, harici değişkenli

ve tümlevli mevsimsel otoregresif hareketli ortalamalar, gradyan destekli yumuşak karar ağaçları.

Acknowledgement

I would like to thank Prof. Süleyman Serdar Kozat for his great supervision during my M.S. studies. His consistent guidance and encouragement greatly enhanced my studies and helped me grow as a researcher.

I would like to thank Prof. Sinan Gezici and Prof. Ramazan Gökberk Cinbiş for kindly accepting to be part of my thesis committee and for their valuable time. I would also like to thank Prof. Aydın Alatan for introducing me to the world of research during my undergraduate years.

I would like to express gratitude to my fellow researchers in Prof. Kozat's research group: Ahmet Berker Koç, Furkan Burak Mutlu, Mehmet Yiğit Turalı, Mustafa Enes Aydın and Selim Furkan Tekin. I believe our collaborative work environment has helped us all advance further in our respective areas of research. I extend my thanks to all my beloved friends who have significantly eased my stress and supported me throughout my academic and personal life.

Finally, and most importantly, I greatly thank my family for their incredible faith and encouragement throughout my journey. Without them, I am certain that I would not have come this far in my academic career. I have no doubt that they will continue to support me in the future, and for this, I express my deepest gratitude.

I would also like to acknowledge Türk Telekom for their generous sponsorship under the 5G and Beyond Graduate Scholarship Programme and the Scientific and Technological Research Council of Turkey (TUBITAK) for providing me with financial support under the BIDEB 2210-A program.

Contents

List of Figures

List of Tables

List of Publications

This thesis includes content from the following publications:

1. A. Fazla, M. E. Aydin, and S. S. Kozat, "Time-Aware and Context-Sensitive Ensemble Learning for Sequential Data", *IEEE Transactions on Artificial Intelligence,* 2023. [1]

2. A. Fazla, M. E. Aydin, and S. S. Kozat, "Joint Optimization of Linear and Nonlinear Models for Sequential Regression", *Digital Signal Processing,* 2023. [2]

Chapter 1

Introduction

We investigate the field of nonlinear time series regression, specifically focusing on prediction, learning, and related tasks. In this context, we are provided with a target sequence and our goal is to forecast the future samples of this sequence. To make accurate predictions, we utilize the past values of the sequence and a data-specific side-information vector. This issue is thoroughly examined in the fields of machine learning, computational learning theory, and signal processing across several applications [3–5]. Linear modeling approaches are frequently employed in such regression problems due to their low computing complexity and resilience when dealing with limited data [6]. Nevertheless, in the majority of practical scenarios, the data exhibits nonlinear properties [7], making linear models insufficient [8]. Thus, nonlinear models such as decision trees [9] and neural networks [10] are widely utilized. Note that, as linear models are a subset of nonlinear models, nonlinear models can theoretically model both the linear and nonlinear data. Nevertheless, nonlinear approaches are hindered by well-known shortcomings, as they are hard to optimize due to their high computational complexity [11] and due to their tendency to overfit [12].

Hence, ensemble or mixture models are extensively studied in the fields of signal processing and machine learning [13], as they provide reliability and diversity by integrating multiple models, including both linear and nonlinear models [14].

Ensemble learners generally outperform individual learners, as long as the individual learners are sufficiently accurate and diverse [15, 16]. The superior performance of ensemble models has been demonstrated in numerous well-known forecasting competitions [17, 18]. Popular techniques for ensembling include boosting [19] and bagging [20]. The boosting technique involves the integration of a series of weak models in a sequential fashion, where each model compensates for the errors made by the preceding models. Bagging, also known as bootstrap aggregation, involves training weak models independently -with samples and/or features reduced- and then combining them using a deterministic process.

Here, we focus on two separate state-of-the-art combination methodologies. Initially, we examine the technique of combining a collection of machine learning models by employing a weighted average of their predictions, commonly known as stacking [14]. Combining forecasts using a weighted average so that a certain performance metric is minimized has been shown to be a successful approach for hindering the effects of selecting a weak base model for the specified learning task [21] and has been extensively investigated in the literature [14,21,22]. Combining forecasts using a weighted average so that a particular performance metric is minimized has been shown to be a successful approach for hindering the effects of selecting a weak base model for the specified learning task [21] and has been extensively studied in the literature [14, 21, 22].

The primary obstacle in the stacking strategy is in the selection of suitable combination weights. Two straightforward approaches are to either combine all models with equal weight or to choose the best base learner at each time sample. A more advanced alternative approach involves constructing a separate machine learning model, commonly known as the "meta-learner" (or ensemble learner) [23], which is trained to linearly combine the predictions made by the basic models. However, previous simple strategies have consistently proven to be more effective than complex weight combination techniques in many competitions, which is referred to as the "forecast combination puzzle" in the literature [24]. Therefore, we tackle the meta-learning problem of selecting the most optimal combination weights for each time sample, using a side-information vector that is particular to the data.

The meta-learning problem has been thoroughly examined in the literature based on two common approaches. The initial approach involves providing the predictions of the base model to an ensemble learner, which is often chosen to be Artificial Neural Network (ANN) [25], Support Vector Regressor (SVR) [26], Linear Regressor [27], etc. Here, the ensemble learner makes direct predictions of the target sequence, while the base learners' predictions are directly fed to the meta-learner. This approach has well-known limitations. For example, the ensemble learner frequently produces non-optimal predictions because it lacks the necessary context for making accurate predictions. In other words, certain basic models may outperform others in specific contexts or states. However, the ensemble learner lacks access to context-specific knowledge, hence, this context-specific information tends to get lost during the ensembling process. Moreover, even when incorporating additional contextual information as input to the ensemble learner, the base predictions themselves tend to dominate the other features, i.e., the context information is hardly utilized [23]. Consequently, the weights in these combinations have a tendency to be biased towards the prediction of the "best" base predictor, resulting in the meta-learner overfitting to this particular base model.

On the other hand, the latter method [21, 28] relies on solving a weight optimization problem that minimizes a specific loss function and learns to predict the best combination weights by considering the base model predictions and a data-specific side-information vector. This method inherently addresses the issue of overfitting, which is commonly associated with previous direct combination strategies. This is achieved by excluding the predictions of the base learners from the ensemble feature vector. In the context of current research on this particular approach, the ensemble learner is typically trained on multiple time series. In this case, the ensemble model learns a single combination weight vector for each time series [21, 28]. However, in the majority of real-life situations, time series data is not solely generated by a single stochastic process due to the fact that the environment can exhibit varying features [29]. Specifically, the behavior of the underlying system that produces the desired sequence can exhibit temporally varying statistics [29]. Although it is possible to analyze these properties

using a time-varying feature vector, the variance information is not retained in the combination stage of the state-of-the-art ensemble algorithms in the literature [21, 28]. This is because only a single weight vector is generated and utilized by the ensemble learner for each time series.

In Chapter 1, we resolve these concerns by approaching the weight optimization problem in a time-varying manner. At each time sample, we generate the combination weight vector based on the side-information vector, i.e., based on the context. Furthermore, our ensembling approach has the ability to determine which base models are more effective in specific states or contexts compared to others. This addresses a key component of the model combination problem that has been neglected in the literature. In order to achieve this objective, we propose two novel meta-learners that are designed for specific learning tasks. These meta-learners effectively merge the base predictions in a manner that is reliant on both the context and time. For the linear combination of the base predictions, our ensemble learners provide a weight vector for each time sample, which can be unconstrained or can satisfy the affine or convex constraints. Specifically, we utilize the LightGBM (a gradient boosting machine [30]) and an Artificial Neural Network (ANN) [31] as the meta-learners to demonstrate the effectiveness of our ensembling technique. Our ensembling approach is generic, allowing for the utilization of any differentiable machine learning model, given the appropriate learning task. In addition, we utilize a novel training scheme for the meta-learners, which mitigates the possible overfitting problem of the ensemble learners to a specific base model. The main contributions of Chapter 1 to the literature are as follows.

- We find the most optimal combination weights for the base learners while considering three different weight constraints. As far as we know, we are the first to tackle the problem both in a time-varying and side-information dependent context. In this setting, we find the optimal combination weights for each time sample based on the target-specific side-information vector.
- In order to learn the most optimal time and side-information dependent weights, we introduce two novel and generic ensembling algorithms built

upon neural networks [31] and boosted decision trees [19], where we provide all the relevant equations. The introduced novel ensemble training scheme effectively addresses any training-related challenges encountered by the base models, including co-linearity, high correlation, and overfitting.

- In order to demonstrate the superiority of our technique, we compare our ensemble framework against the state-of-the-art machine learning models and the conventional ensembling approaches in the literature. We provide our results in many widely recognized benchmark time series datasets [18, 32].

- We openly share the implementation of our algorithm to facilitate model design, comparisons and experimental reproducibility[1].

In this thesis, we also discuss another state-of-the-art model mixture method for combining linear and nonlinear models, which involves fitting one of the models to the residuals of the other model [33]. This methodology is referred to as direct two-stage modeling and has been extensively researched in the fields of signal processing and machine learning [33, 34]. This direct approach [7] aims to capture both the linear and nonlinear data components in real-life scenarios by fitting one model to the residuals of the other. Nevertheless, straight two-stage models are inadequate for modeling real-life sequential data due to the lack of joint optimization. Here, one model is initially fitted to the data without considering the other model, and then the second model is fitted to the residuals of the first model. Therefore, the models are not trained together in order to minimize the final error, resulting in suboptimal performance [35].

In Chapter 2, we adopt an alternate ensemble/mixture technique that involves jointly optimizing the basic learners. We model the underlying time series data as an ensemble or combination of linear and nonlinear models and optimize the model parameters together to minimize the regression error. Our approach utilizes both linear and nonlinear models to generate a single joint forecast for the data samples. Thus, the correlation between the models at each time sample is

[1] https://github.com/ardafazla/context-time-aware-ensemble

not pre-established but acquired through joint optimization. Therefore, we utilize the distinctive benefits of both models, thereby enhancing the overall regression performance as shown in our simulations. We employ the widely recognized SARIMAX (Seasonal Auto-Regressive Integrated Moving Average with eXogenous components) model as the linear model [36]. For the nonlinear model, we utilize the boosted soft decision trees (Soft GBDT) [37]. We include the corresponding gradient computations for both structures. Our system is designed to be versatile, allowing for the adoption of any differentiable nonlinear or linear model in place of SARIMAX or Soft GBDT. The main contributions of Chapter 2 to the literature are as follows.

- We propose a two-stage model that combines the linear SARIMAX model and the nonlinear Soft GBDT model for sequential time series regression. Our model can be jointly optimized using any gradient-based optimization algorithm.
- Our introduced structure is versatile, as the linear and nonlinear models in our method can be easily substituted with any other differentiable regression model.
- Through a series of tests on real-life data, we demonstrate substantial enhancements in performance compared to the mixture approaches in the literature.
- We openly share the implementation of our algorithm to facilitate model design, comparisons and experimental reproducibility[2].

Notation: All the vectors in this work are represented by bold lowercase letters. Matrices are represented by capital bold letters. For example, $\boldsymbol{x}$ denotes a vector while $\boldsymbol{X}$ denotes a matrix. x_k represents the k^{th} element of the vector $\boldsymbol{x}$, whereas $x_{t,k}$ represents the k^{th} element of the vector $\boldsymbol{x}_t$ at time t. $X_{i,j}$ denotes the entry at the i^{th} row and j^{th} column of the matrix $\boldsymbol{X}$. Finally, $\boldsymbol{x}^T$ and $\boldsymbol{X}^T$ denote the transpose of the vector $\boldsymbol{x}$ and the matrix $\boldsymbol{X}$, respectively.

[2]https://github.com/ardafazla/jointoptimization

All vectors in this work are represented by bold lowercase letters. Matrices are represented by capital bold characters. For example, the sign $\boldsymbol{x}$ denotes a vector, but the symbol $\boldsymbol{X}$ denotes a matrix. x_k represents the k^{th} element of the vector $\boldsymbol{x}$, whereas $x_{t,k}$ designates the k^{th} member of the vector $\boldsymbol{x}_t$ at time t. The symbol $X_{i,j}$ denotes the element located at the intersection of the i^{th} row and j^{th} column in the matrix $\boldsymbol{X}$. Finally, $\boldsymbol{x}^T$ and $\boldsymbol{X}^T$ denote the transposition of the vector $\boldsymbol{x}$ and the matrix $\boldsymbol{X}$, respectively.

Chapter 2

Ensemble Learning with Time Awareness and Context Sensitivity for Sequential Data

The structure of this chapter is as follows. In Section 2.1, we provide a brief literature review, with a specific emphasis on ensemble learning. In Section 2.2, we presentthe problem of finding the time-dependent optimal base model combination weights based on the side-information vector under different constraints. In Section 2.3, we describe the training structure of our novel ensemble framework. Our framework aims to identify the best linear combination weights by minimizing a specific loss function. In Section 2.4, we provide two novel and generic ensemble learners. We demonstrate the effectiveness of our algorithms through comprehensive experiments using both real-life and synthetic data in Section 2.5. Furthermore, we analyze the associated learning costs of the optimal combination weight vectors for all three constraints in Appendix A, and present the explicit backpropagation equations for our ensemble algorithms in Appendix C.

2.1 Related Work

Ensemble models are extensively studied in the fields of machine learning, computational learning theory, statistics, and signal processing [38–40]. This is because ensemble models outperform individual models, given that the individual learners are both accurate and diverse [15, 16]. Popular techniques for ensembling include boosting [19], bagging [20], and stacking [14]. We are specifically interested in a kind of stacking known as meta-learning, which involves training a separate machine learning model to linearly combine the predictions from base models. Simpler methods for tackling the model combination problem exist, such as training an ensemble model using the base model predictions as the feature vector without the need for a separate optimization problem. For example, a Support Vector Regressor (SVR) was used to predict electrical load demand [26]. However, these techniques are prone to overfitting to the predictions of the base models due to the absence of any side-information related to the target sequence in the input vector. While supplemental features can be utilized as inputs for the ensemble learner to give contextual information, the predictions made by the basic models tend to overshadow these features, resulting in limited utilization of the context information [23].

Conversely, the issue of determining the most optimal weights to combine predictions from different base models has received significant focus in recent literature. More precisely, recent studies have concentrated on addressing a weight optimization problem that aims to minimize a certain loss function and learns to predict the optimal combination weights [21, 28]. These studies vary in several aspects, including the choice of the ensemble learner, the optimization problem to be addressed, the structure of the feature vector used for training the ensemble learner, and the constraint on the linear combination space. In the given context, a super-ensemble model [28] has been suggested for predicting daily streamflow. The ensemble learner is formed by combining different machine learning methods using convex weights. The weights are determined by minimizing the root mean squared error of the ensemble prediction, which is the weighted linear sum of the predictions made by the base models. The setback of the proposed method is

that the combination weights are set the same for all the test samples. Therefore, this argument fails to account for the potential fluctuating dynamics of the target sequence. Nevertheless, real-world datasets frequently exhibit alternating dynamics [29]. Hence, certain base models may outperform others in specific time samples (or states), a factor that is neglected during the ensembling process.

Another recent ensemble model is the FFORMA model [21], which is introduced in the M4 Forecasting Competition [21]. Here, the authors employ the XGBoost [41] as the meta-learner, which learns to produce combination weights by minimizing the weighted average of the base model losses in a convex optimization constraint. The authors attract attention to the fact that they employ the target-specific side-information vector as the input to their ensemble model. Hence, the model learns to produce combination weights dependent on the side-information vector. However, in the given method, the ensemble learner is trained on multiple time series with similar characteristics, all strongly dependent on the environmental factors aggregated into the side-information vector. In the end, the ensemble model learns to predict the optimal combination weights for each time series, independent of time. Hence, the given model does not consider the data's time-varying characteristics while producing combination weights, such as the seasonality and trend of the target sequence [17]. In addition, the linear combination search space is only considered under a convex constraint. However, affine and/or unconstrained combinations can also lead to optimal results in real-life scenarios, as we illustrate in our simulations.

Another recent ensemble model, known as the FFORMA model [21], was launched in the M4 Forecasting Competition [21]. In this study, the authors utilize the XGBoost algorithm as the meta-learner. The XGBoost algorithm minimizes the weighted average of the losses of the base models under a convex optimization constraint in order to learn the optimal combination weights. The authors emphasize their use of the target-specific side-information vector as input to their ensemble model. Therefore, the model acquires the ability to generate combination weights that are dependent on the side-information vector. Nevertheless, in the provided approach, the ensemble learner is trained on numerous

time series that possess similar characteristics, all strongly dependent on the environmental factors aggregated into the side-information vector. Ultimately, the ensemble model acquires the ability to forecast the most advantageous combination weights for each time series, regardless of the time factor. Therefore, the provided model does not take into account the time-dependent attributes of the data while generating combination weights, such as the seasonal patterns and trend of the target sequence [17]. Furthermore, the search space for linear combinations is exclusively taken into account under a convex constraint. However, affine and/or unconstrained combinations can also lead to optimal outcomes, as we demonstrate in our simulations.

2.2 Problem Statement

Our research focuses on forecasting time series data by employing a linear combination of various learning algorithms. The primary sequence we seek to forecast is $\{y_t\}$; to this end, we utilized M machine learning models, referred to as base models, each of which incorporates a side-information sequence $\{\boldsymbol{s}_k^{(i)}\}$ for $k \leq t$ and $i = 1, \ldots, M$. In this context, the term "side-information sequence" denotes the input vector that is provided to the model at each time sample. It is important to note that each base model has the freedom to employ a distinct side-information vector, e.g., $\{\boldsymbol{s}_k^{(i)}\}$ could include the observed past information $\{y_k\}$ along with model-specific features. At each time t, the base models generate predictions $\hat{y}_t^{(i)}$. We combine the M predictions by using a linear scheme. The ensemble prediction $\hat{y}_t^E$ of y_t is given as

$$\hat{y}_t^E = \boldsymbol{w}_t^T \hat{\boldsymbol{y}}_t, \tag{2.1}$$

where $\hat{\boldsymbol{y}}_t = [\hat{y}_t^{(1)}, \ldots, \hat{y}_t^{(M)}]^T$ is the base prediction vector of size M which consists of the individual predictions made by the base models and $\boldsymbol{w}_t \in \mathbb{R}^M$ is the ensemble weight vector at time t. Even though the ensembling scheme is linear, the adaptive learning procedure, which produces $\boldsymbol{w}_t$, can be highly nonlinear, as

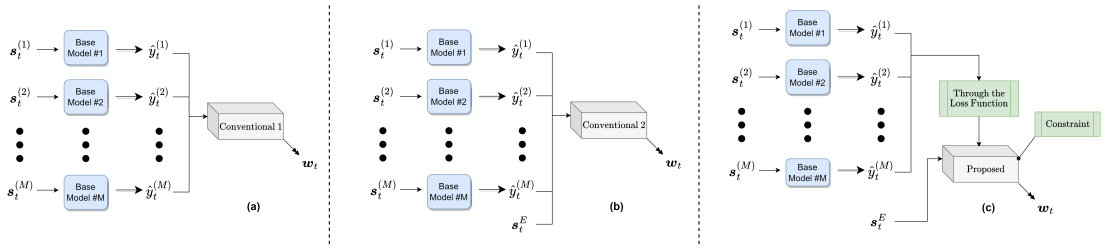


Figure 2.1: Comparison between our ensembling approach and the conventional models described in the literature. In scenario (a), the ensemble learner receives the base model predictions as the feature vector. In (b), in addition, a data-specific side-information vector is given to the ensemble learner. In neither of the two scenarios, the conventional models exploit the errors of the base models. Instead, the predictions of the base models are only used as features for the ensemble learner. In contrast, our strategy (c) does not include the base model predictions in the feature vector that is given to the ensemble learner. Instead, we leverage the errors of the base models through our generic ensemble loss function. In this context, our model is designed to learn and generate the most optimal combination vector for each time sample t, based on the provided combination constraint.

we show in Section 2.3. When we observe y_t, we suffer the loss

$$L_t = \ell(y_t, \hat{y}_t^E) = \ell(y_t, \boldsymbol{w}_t^T \hat{\boldsymbol{y}}_t) \\ = \ell\big(y_t, \sum_{i=1}^{M} w_t^{(i)}\, \hat{y}_t^{(i)}\big), \tag{2.2}$$

where ℓ is a chosen loss function, such as the squared error loss.

Commonly in the literature, the weight vector $\boldsymbol{w}_t$ is learned based solely on the predictions of the base models, i.e., $\boldsymbol{w}_t = f(\hat{\boldsymbol{y}}_t)$, where f represents a learning model [42, 43]. In this case, the ensemble learner makes direct predictions of the target sequence using the base model predictions as the input vector at time t, as shown in Fig. 2.1 (a). Within the specified scenario, the ensemble learner has a tendency to generate non-optimal predictions due to its limited understanding of the context in which these predictions are made. For instance, certain base models may exhibit superior performance compared to others in specific contexts or states. However, this information is not taken into account during the ensembling process because the meta-learner lacks awareness of this contextual

information [23]. Moreover, even if additional context information is included as inputs to the ensemble learner, as illustrated in Fig. 2.1 (b), the base predictions themselves have a tendency to overshadow the other features, i.e., the context information is hardly utilized [23]. Consequently, the weights in these combinations often prioritize the prediction of the "best" base predictor, causing the meta-learner to overfit to the aforementioned base model.

Thus, we propose a novel ensembling method, where the meta-learner solves a weight optimization problem to minimize a specific loss function. The meta-learner then learns to predict the optimal combination weights using a data-specific side-information vector. In this setting, the predictions of the base models are not directly integrated into the feature vector of the base model. Instead, they are employed through minimizing a specific loss function, as shown in Fig. 2.1 (c). In our suggested setting, we examine three different mixture techniques, distinguished by how they constrain the combining weight vector $\boldsymbol{w}_{\boldsymbol{s}_t^E}$. These are the unconstrained linear combination where each $w^{(i)}_{\boldsymbol{s}_t^E}$ can take any real value, the affine linear combination where the components of the weight vector is bound to sum up to 1, i.e., $\boldsymbol{w}^T_{\boldsymbol{s}_t^E}\mathbf{1} = 1$ and finally the convex linear combination where the weight vector not only has its components sum to 1 but also all nonnegative, i.e., $\boldsymbol{w}^T_{\boldsymbol{s}_t^E}\mathbf{1} = 1$ and $w^{(i)}_{\boldsymbol{s}_t^E} \geq 0\,\forall i$; all of these restrictions apply to all time steps. Therefore, our objective is to dynamically determine the "best" weights that linearly combine the predictions of M parallel running base models in order to minimize a specified loss ℓ at each time step t. This is achieved by utilizing the ensemble side-information vector $\boldsymbol{s}_t^E$ under three different constraints on the weight vector.

To demonstrate the process of ensembling and the utilization of the side-information vector, let's consider the case of predicting hourly wind power and employing two base learners running in parallel. The initial model could incorporate three features: wind power, wind speed, and wind direction from the preceding hour. The second model might utilize two features as the side-information: the wind power and direction from 24 hours prior. The ensemble model can then include all features utilized by the base learners, in addition to an hour-of-day indicator to identify the learning patterns of each base learner. Thus, "side-information" pertains to the data and model-specific input vector that each

model is fed with at each time sample, which can differ depending on the unique learning architectures of the models. We note that, during the ensembling process, the side-information, or the input vector, of the ensemble learners does not include the predictions made by the individual base learners. In our simulations, we illustrate that this prevents our ensemble learners from overfitting to the predictions of a single base model.

To demonstrate the process of ensembling and the utilization of the side-information vector, let's consider the case of predicting hourly wind power in real-time and employ two base learners running in parallel. The initial model could incorporate three variables: wind power, wind speed, and wind direction from the preceding hour. The second model might incorporate two variables as side-information: the wind power and direction from 24 hours prior. The ensemble model can include all the features utilized by the base learners, in addition to an hour-of-day indicator that identifies the learning patterns of each base learner. Thus, the term "side-information" pertains to the individual data and input vector that is sent to each model at each time sample, and this can differ depending on the unique learning architectures of the models. It is important to note that when combining many learners into an ensemble, the side-information or input vector of the ensemble learners does not incorporate the predictions made by the individual base learners. In our simulations, we demonstrate that our ensemble learners are protected from overfitting to the predictions of a single base model.

When the statistics of both the combined predictions and the target sequence are completely known, the unconstrained combination of the base predictors produces the minimum error possible [44]. Nevertheless, in reality, those statistics are rarely known. Therefore, the process of determining the optimal linear combination weights using a meta-learner is susceptible to learning difficulties. This means that there is no guarantee that all the parameters will be accurately learned, and as a result, an optimal solution may not be achieved. In the unconstrained case, there are no limitations on the search space for linear combinations. This means that the associated learning problem is larger compared to when the search space is restricted to a specific subspace based on constraints. Consequently, unconstrained combinations may overfit or may not have sufficient data to identify

the best optimal combination, resulting in a sub-optimal solution. By imposing constraints that reduce the size of the linear combination search space, we can effectively lower the learning difficulty. Nevertheless, the probability of disregarding the most suitable combination weight vector within the subspace rises proportionally. This phenomenon is commonly referred to as the bias-variance trade-off in the field of machine learning. Thus, we examine three separate weight constraints: unconstrained, affine, and convex, which vary in the complexity of the associated learning problem. A more detailed analysis of the reasoning behind using an ensemble learner for the given constraints is provided in Appendix A.

Remark. *The meta-learning approach, which involves solving a weight optimization problem to minimize a specific loss function and predict optimal combination weights using base model predictions and a data-specific side-information vector, has been used extensively in previous studies [21, 28]. However, in the context of previous studies on this method, the ensemble learner is commonly trained on multiple time series. In this scenario, the ensemble model learns a single optimal combination weight vector for each time series, regardless of the time sample t. Therefore, the resulting combination weights are independent of time. In contrast, our method focuses on a single time series data and tackles the weight optimization problem in a time-varying manner. At each time sample t, we generate the combination weight vector by considering the side-information vector and the current context. This technique enables the combination weights to vary over time and adapt dynamically according to the context.*

In the next section, we outline the learning and training process of our novel ensembling method, along with the generic ensemble algorithms utilized to find the optimal combination weight vectors based on the specified constraint.

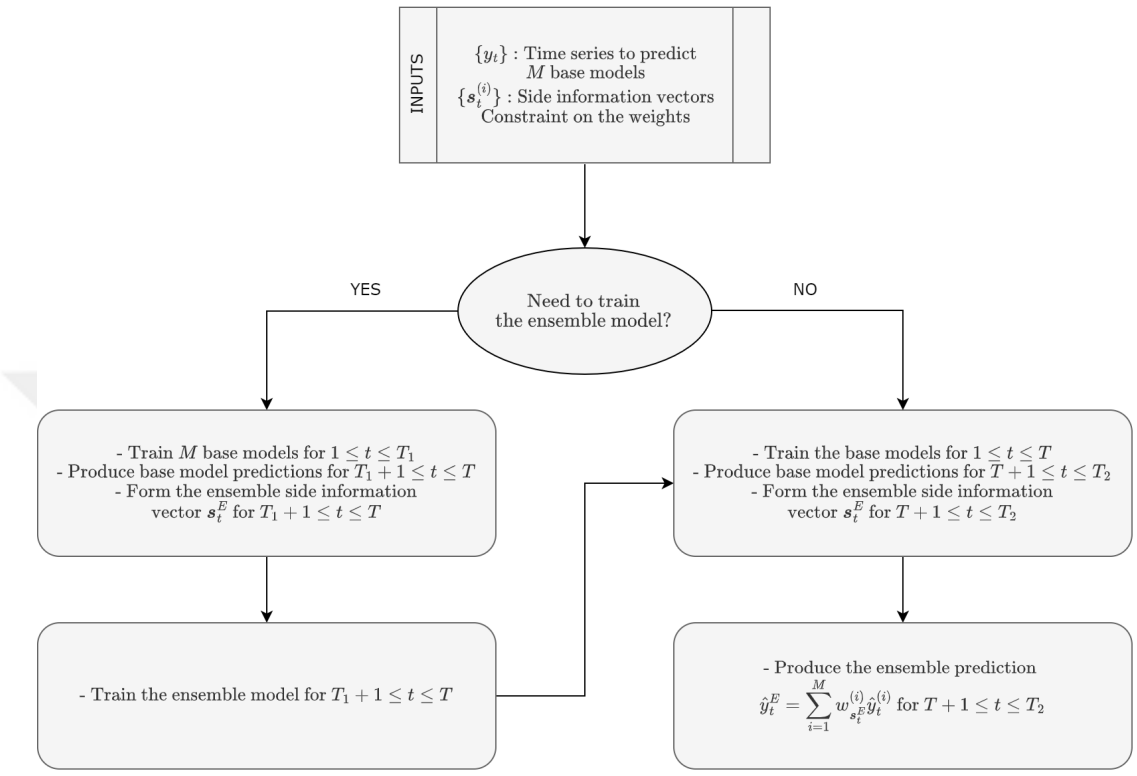


Figure 2.2: Our ensemble training scheme flowchart.

2.3 Introduced Approach

We employ ensemble algorithms to linearly combine the predictions of various base models by giving weights to each model. The ensemble algorithms in our system learn to merge the predictions made by several base algorithms by minimizing the loss function specified in equation (2.2), where y_t is the observed sequence at time t, ℓ is a given differentiable loss function such as the squared error loss, $\hat{y}_t^E$ is the prediction made by the ensemble algorithm, M is the total number of base models, $\hat{y}_t^{(i)}$ represents the predictions made by the individual base models and $w_{\boldsymbol{s}_t^E}^{(i)}$ are the weights assigned to each base model based on the side-information sequence $\boldsymbol{s}_t^E$.

To train our ensemble algorithms, we propose a novel ensemble training

method, as shown in Fig. 2.2. Our training scheme is divided into two distinct phases: the offline phase and the online phase. During the offline phase, the base models are initially trained independently on a certain subset of the training dataset, denoted as $1 \leq t \leq T_1$. Next, the base models' predictions are obtained on the remaining partition of the training data, given as $T_1+1 \leq t \leq T$. These predictions are then utilized to train our ensemble algorithm. Therefore, by collecting predictions on data that has not been seen before, we are able to address the potential problem of overfitting that is commonly seen in traditional ensemble methods [45]. Each base model i utilizes a side-information vector $\boldsymbol{s}_t^{(i)}$ to generate predictions, which may consist of the historical observations of the sequence y_t as well as any relevant information to the observed sequence or the base model. Our ensemble model is trained using the errors of the base models while incorporating the side-information sequence $\boldsymbol{s}_t^E$. This sequence can include past observations of the sequence y_t, side-information related to the base models, and additional information specific to the observed sequence.

During the online phase, we utilize our pre-trained ensemble algorithm from the offline phase to generate combination weights for the base algorithms on the new data. In our example, this new data consists of the observations that come after the training samples of the sequence y_t. Initially, we proceed to retrain the base models using the entire training dataset, $1 \leq t \leq T$. Next, we generate the initial model predictions for the test dataset, $T+1 \leq t \leq T_2$. Next, we proceed to update the ensemble side-information vector $\boldsymbol{s}_t^E$ for the test dataset. Ultimately, we generate the ensemble forecast $\hat{y}_t^E = \sum_{i=1}^{M} w_{\boldsymbol{s}_t^E}^{(i)} \hat{y}_t^{(i)}$, for $T+1 \leq t \leq T_2$. Appendix B provides a comprehensive explanation of our ensemble learning technique.

Next, we introduce two novel and generic ensemble learners to obtain the optimal linear combination weights, given the three combination constraints introduced in Section 2.2.

2.4 Ensemble Models

2.4.1 LightGBM Ensemble

Gradient boosting decision trees (GBDT) refer to ensembles of decision trees that are trained sequentially. During each iteration, Gradient Boosting Decision Trees (GBDT) aim to minimize the residual error by the previous trees in the sequence. LightGBM is an advanced GBDT model that minimizes a specific loss function for the entire dataset during training to find the best split points. It is known for its efficiency in terms of memory usage and training speed compared to other similar models [30].

We utilize LightGBM to explore for the most optimal combination weights, as outlined in **Algorithm 1**. LightGBM requires the gradient and hessian of the objective function (2.2) to minimize the loss with respect to the combined weight vector $\boldsymbol{w}_{\boldsymbol{s}_t^E}$. As explained in Section 2.2, we cover the linear combination scenario under unconstrained, convex-constrained, and affine-constrained cases. LightGBM generates an output value $p^{(i)}_{\boldsymbol{s}_t^E}$ for each base learner i. Then, we apply the necessary transformation to obtain $w^{(i)}_{\boldsymbol{s}_t^E}$ for the specified weight constraint. Hence, we study and provide the gradient and hessian of (2.2) with respect to $p^{(i)}_{\boldsymbol{s}_t^E}$, for each base learner i under all three constraints. We designate the loss function ℓ as the squared error loss, however, any differentiable loss function can be chosen. We assume that have the base predictions for $1 \leq t \leq T$, as explained in Fig. 2.2. Thus, for any specific time t in the training dataset, LightGBM ensemble minimizes the loss

$$L_t = \ell(y_t, \hat{y}_t^E) = (y_t - \hat{y}_t^E)^2 = (y_t - \sum_{i=1}^{M} w^{(i)}_{\boldsymbol{s}_t^E} \hat{y}_t^{(i)})^2,$$

where y_t is the observed target sequence, and $\hat{y}_t^E$ is the prediction generated by our ensemble algorithm for time t. The transformation from $p^{(i)}_{\boldsymbol{s}_t^E}$ to $w^{(i)}_{\boldsymbol{s}_t^E}$ is obtained by $w^{(i)}_{\boldsymbol{s}_t^E} = \tau(p^{(i)}_{\boldsymbol{s}_t^E})$, where τ indicates the transformation operation and is specific to the given weight constraint. Therefore, the gradient and hessian that need to be calculated at time t for the base learner i are expressed as

$$G_i = \frac{\partial L_t}{\partial p^{(i)}_{\boldsymbol{s}^E_t}}, H_i = \frac{\partial G_i}{\partial p^{(i)}_{\boldsymbol{s}^E_t}}. \tag{2.3}$$

In Appendix C.1, we provide a detailed explanation of the required transformations and present the computations for the gradient and hessian of all three constraints for the LightGBM ensemble.

2.4.2 MLP Ensemble

Artificial Neural Networks (ANN) are computational models that mimic the processing organization of brain cells and are employed to simulate intricate patterns and solve challenging problems [46]. The model's learning structure is built around interconnected nodes arranged in layers, which facilitate the transmission of information through weighted connections. Our ensemble approach utilizes a specific type of artificial neural network (ANN) known as a Multilayer Perceptron (MLP), which is an ANN with more than two layers. MLPs are commonly utilized in the field of machine learning because of their capacity to learn complex and non-linear relations with high training speed in comparison to other models [47].

As a specific instance, we present the equations for a two-layered network structure comprising an input layer, a hidden layer and an output layer. Our derivations can be straightforwardly extended to increase the number of layers. At the last stage of the output layer, we apply a transformation operation that maps the weights of the ensemble algorithm based on the given weight constraint. As explained in Section 2.2, we cover the linear combination scenario under unconstrained, convex-constrained, and affine-constrained cases. The feed-forward equations of our model can be expressed as follows

$$\begin{aligned} \boldsymbol{v} &= \boldsymbol{U}^{(1)}\boldsymbol{x} \\ \boldsymbol{z} &= f(\boldsymbol{v}) \\ \boldsymbol{p} &= \boldsymbol{U}^{(2)}\boldsymbol{z} \\ \boldsymbol{w} &= \tau(\boldsymbol{p}), \end{aligned}$$

where $\boldsymbol{x} \in \mathbb{R}^K$ is the input vector, $\boldsymbol{U}^{(1)} \in \mathbb{R}^{L\text{x}K}$ are the layer weights that connect the input layer to the hidden layer, $\boldsymbol{z} \in \mathbb{R}^L$ is hidden layer input, given as the rectified linear activation function (ReLU) f applied to $\boldsymbol{v} \in \mathbb{R}^L$. $\boldsymbol{U}^{(2)} \in \mathbb{R}^{K\text{x}M}$ are the layer weights that connect the hidden layer to the output layer, $\boldsymbol{p} \in \mathbb{R}^M$ is the output vector of the ensemble model. $\boldsymbol{w} \in \mathbb{R}^M$ is the linear combination weight vector, obtained by applying the transformation operation τ to $\boldsymbol{p}$. K, L, M are the lengths of the input, hidden, and output layers, respectively.

Our ensemble algorithm learns to generate the optimal linear combination weights by minimizing the loss function (2.2). For all constraints, we designate ℓ as the squared error loss, although any differentiable loss function can be chosen. We also assume that we have the base predictions for $1 \leq t \leq T$, as explained in Fig. 2.2. Thus, at any specific time t in the training data, the MLP ensemble minimizes the loss function expressed by

$$L_t = \ell(y_t, \hat{y}_t^E) = (y_t - \hat{y}_t^E)^2 = (y_t - \sum_{i=1}^{M} w_{\boldsymbol{s}_t^E}^{(i)} \hat{y}_t^{(i)})^2,$$

where y_t is the observed target sequence, and $\hat{y}_t^E$ is the prediction made by the ensemble model, at time t. For simplicity, we used the notations $x_i \triangleq x_{\boldsymbol{s}_t^E}^{(i)}$, $w_i \triangleq w_{\boldsymbol{s}_t^E}^{(i)}$ and $p_i \triangleq p_{\boldsymbol{s}_t^E}^{(i)}$. We iteratively minimize the given loss by updating the layer weights $\boldsymbol{U}^{(1)}$ and $\boldsymbol{U}^{(2)}$ through backpropagation. Hence, the backpropagation equations can be expressed as

$$\frac{\partial L_t}{\partial U_{l,k}^{(1)}} = \sum_{j=1}^{M} \sum_{i=1}^{M} \frac{\partial L_t}{\partial w_i} \frac{\partial w_i}{\partial p_j} \frac{\partial p_j}{\partial z_l} \frac{\partial z_l}{\partial v_l} \frac{\partial v_l}{\partial U_{l,k}^{(1)}},$$
$$\frac{\partial L_t}{\partial U_{m,l}^{(2)}} = \sum_{i=1}^{M} \frac{\partial L_t}{\partial w_i} \frac{\partial w_i}{\partial p_m} \frac{\partial p_m}{\partial U_{m,l}^{(2)}},$$

where $1 \leq l \leq L$, $1 \leq k \leq K$, $1 \leq m \leq M$. Next, we employ stochastic gradient descent (SGD) to update $\boldsymbol{U}^{(1)}$ and $\boldsymbol{U}^{(2)}$. The update equations are provided in the following form

$$U_{l,k}^{(1)} = U_{l,k}^{(1)} - \alpha \frac{\partial L_t}{\partial U_{l,k}^{(1)}}, \quad U_{m,l}^{(2)} = U_{m,l}^{(2)} - \alpha \frac{\partial L_t}{\partial U_{m,l}^{(2)}},$$

where α is a hyperparameter of SGD, which is referred to as the learning rate. In Appendix C.2, we provide a detailed explanation of the required transformations

and present the computations for the gradient and hessian of all three constraints for the MLP ensemble.

Hence, we have provided the update equations to train our MLP and LightGBM ensemble algorithms according to Fig. 2.2.

Remark. *Note that our motivation for introducing two novel ensemble learning algorithms is due to the different learning structures of the given ensemble models. Our LightGBM ensemble determines the optimal linear combination weights by identifying the most effective splitting points for the data samples. This is accomplished by utilizing the gradient and hessian of the loss function with respect to the base model outputs, as explained in (2.3). On the other hand, our MLP ensemble solely relies on the gradient of the loss function with respect to the base model outputs when adjusting the weights of its layers. Therefore, the outcomes of our meta-learners may differ based on the complexity of the data and the weight optimization problem, as illustrated in Section 2.5. Furthermore, while we offer two distinct models for use as our ensemble learners, any machine learning model, such as linear regression models [48] etc., can be utilized as long as they are capable of supporting a custom differentiable loss function.*

Next, we showcase the effectiveness of our ensemble models under different constraints, employing a diverse collection of datasets.

2.5 Experiments

We assess the performance of our ensemble learners, namely the LightGBM ensemble 2.4.1 and the MLP ensemble 2.4.2, across a range of time series datasets. First, we provide a detailed description of the datasets, benchmarks, models, optimization methods, hyperparameter tuning and error metrics that we employ in our simulations. After that, we present an in-depth examination of our results.

2.5.1 Datasets

During the initial phase of our simulations, we create a synthetic dataset that strongly depends on a specific side-information vector (context) and interchanges certain characteristics dependent on time and context. Here, we illustrate that our ensemble learners effectively utilize the base learners to accurately capture all patterns in the data. Subsequently, to demonstrate the effectiveness of our models, we employ three widely recognized benchmark time series datasets that exhibit various patterns/seasonalities and rely on a data-specific side-information vector.

Below is a concise overview of the real-life datasets.

2.5.1.1 Turkish Natural Gas Demand

This dataset provides daily data on the total residential natural gas demand in Turkey from 2018 to 2020. The dataset consists of 1000 samples and exhibits both weekly and monthly seasonality trends. The side-information vectors, which serve as input to the models, can utilize a total of 58 features. The features include historical observations on the total residential natural gas consumption, weather-related variables such as wind speed and temperature, as well as categorical attributes such as indications for the day of the week, day of the month, and day of the year.

2.5.1.2 Istanbul Stock Exchange

This dataset contains the daily return values of the Istanbul Stock Exchange [32]. The side-information vectors can utilize seven other international indexes, along with the historical observations of the target sequence. The dataset spans from Jun 5, 2009 to Feb 22, 2011. The dataset was acquired from the UCI Machine Learning Repository [49].

Table 2.1: Real-Life Dataset Statistics

Dataset	Length	Seasonality	Feature Size
Natural Gas Demand	1000	[7, 30.5]	58
Istanbul Stock Exchange	536	[5, 20]	23
M5 Forecasting	1941	[7]	56

2.5.1.3 M5 Forecasting

This dataset contains the daily data on the number of unit sales of products sold by the USA retail company Walmart [18]. The total number of products is 3049, which are categorized into 3 distinct categories and 7 distinct departments. The products are distributed throughout 10 Walmart locations located in 3 separate states. We use the total number of products sold in a randomly chosen category in a randomly chosen Walmart store. The dataset includes a total of 56 features, which include the historical observations of the target sequence and categorical features such as indications for the day of the week, day of the month, and day of the year. The data exhibits weekly seasonality patterns with strong nonlinearity in specific places.

The statistics of the datasets used in our experiments are summarized in Table 2.1. Each dataset is divided into 60% training, 20% validation, and 20% test splits. All of the features explained in the datasets can be employed within the side-information vectors (inputs) of the models utilized in the experiments. In order to determine the optimal feature subspace for each model, we employ statistical tests and utilize model-specific features, such as feature importance graphs.

2.5.2 Models

Our ensemble structure incorporates four basis learners from diverse backgrounds to ensure diversity. These base leaners include SARIMAX [36], LightGBM [30],

Naïve and MLP. Subsequently, we proceed to train our two ensemble models, the LightGBM ensemble and the MLP ensemble, with the objective of linearly combining the predictions made by these base learners. Next, we evaluate the effectiveness of our ensemble learners by comparing their performance to a set of state-of-the-art techniques in forecasting with multiple patterns. To that extent, we include RNN [50], LSTM [51], GRU [52], TBATS [53], MLP [46], and two MLP meta-learners optimized following the conventional methods given in Fig. 2.1. It is important to emphasize that all of the models discussed above use a distinct side-information vector as the input, which is unique to each model. While training our ensemble learners, we specify the constraints on the linear optimization weights as hyperparameters, in addition to determining the number of base learners. Therefore, when training our ensemble learners, we initialize two extra hyperparameters in addition to the parameters of the employed model. We provide more clarification on the structures of the models in Table 2.2. To avoid unnecessary repetition, we will use the names provided in Table 2.2 to refer to the models.

2.5.3 Hyper Parameter Tuning and Optimization

During our simulations, while training the models, we initially search for the ideal hyperparameters that produce the minimal total squared error loss on the validation set. After identifying the optimal hyperparameters, we proceed to train the models again using the optimal hyperparameter configuration on both the training and validation datasets. It is important to mention that the hyperparameter tuning for the SARIMAX and TBATS models does not involve using a validation mechanism, as they are based on minimizing a certain information criterion. After training the base models on the provided dataset, we proceed to train our ensemble models and the conventional ensembles. It is important to understand that both the traditional MLP ensemble and the base MLP model have identical search spaces. Our MLP and LightGBM ensembles also share the hyperparameter search space of their base model counterparts, however, the optimal constraint and the best set of base models are additionally searched. The

Table 2.2: Explanations of the models employed in our simulations

Model Name	Explanation
SARIMAX	Statistical forecasting model with seasonality and external variables, introduced by Box & Jenkins [36]
LightGBM	Tree-based gradient boosting model
Naïve	Prediction of the future value based on the most recent observation
MLP	Artificial Neural Network with multiple layers
RNN	Recurrent Neural Network introduced in [50].
LSTM	Long-Short Term Memory introduced in [51].
GRU	Gated Recurrent Unit introduced in [52].
LightGBM Ensemble	Ensemble learner introduced in Section 2.4.1
MLP Ensemble	Ensemble learner introduced in Section 2.4.2
TBATS	Exponential smoothing state space model that utilizes ARIMA errors, Box-Cox transformation, Trend and Seasonal components
Conventional 1	MLP meta-learner trained by the conventional approach illustrated in Fig. 2.1 (a)
Conventional 2	MLP meta-learner trained by the conventional approach illustrated in Fig. 2.1 (b)

training process for our ensemble models is conducted according to Fig. 2.2. Table 2.3 includes the search space for the hyperparameters of the models that we use in our experiments.

2.5.4 Error Metrics

We evaluate the precision of the models included in our simulations by assessing various error measures, including Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). We exclusively focus on one-step-ahead forecasting, and during the learning process, all models are designed to minimize the loss function defined in equation (2.2), which measures the

squared error. During the model validation process, we choose the best collection of hyperparameters by evaluating the MSE on the validation set.

2.5.5 Synthetic Data Experiments

We conduct our initial experiments on synthetic data to assess the effectiveness of our ensembling approach in detecting all different regions in the data by exploiting the side-information in a time-dependent manner. The synthetic data comprises two manually generated and independent data components that are created separately as

$$\begin{aligned} y_t^{\{1\}} &= -0.5y_{t-1}^{\{1\}} + 0.4y_{t-2}^{\{1\}} + v_t^{\{1\}} \\ y_t^{\{2\}} &= y_{t-1}^{\{2\}} + v_t^{\{2\}}, \end{aligned}$$

where $v_t^{\{1\}}$ and $v_t^{\{1\}}$ are sample functions derived from a stationary white Gaussian process with a variance of one. To this end, we create synthetic data with a deterministic switching mechanism, which consists of two distinct regions. These regions are defined by the following set of equations:

$$y_t = \begin{cases} 0.2y_t^{\{1\}} + 0.8y_t^{\{2\}} & (\text{mod } t, 200) < 100 \\ 0.8y_t^{\{1\}} + 0.2y_t^{\{2\}} & (\text{mod } t, 200) \geq 100. \end{cases}$$

Fig. 2.3 depicts the synthetic data that we generated. The data exhibits two distinct patterns/regions alternating in time. It should be noted that our ensemble models, as described in Section 2.3, utilize a side-information vector that is specific to the data. The base learner predictions are combined linearly based on the side-information vector to achieve superior prediction. For this purpose, our two data components, $y_t^{\{1\}}$ and $y_t^{\{2\}}$, imitate the predictions of the base learners that we would typically obtain for the real-life datasets. Furthermore, we incorporate the time information t as the side-information vector into our ensemble

models. Therefore, as we stated in Section 2.2, our ensemble models must possess the capability to accurately forecast the target sequence y_t by thoroughly understanding the switching regions and generating the appropriate linear combination weights for the provided data components (predictions from the base learners).

The final linear combination weight vector on the whole dataset under convex constraint is shown in Fig. 2.4. We also replicate the identical experiments for the Conventional 1 and Conventional 2 ensemble models and contrast the outcomes in relation to the error metrics on the test dataset in Table 2.4. It is evident that the traditional ensembles are unable to understand the areas where the data switches, whereas our ensembling system accurately adapts to the switching patterns, hence supporting the importance of our ensemble structure. Our ensemble models have the capability to generate the optimal weight vector for linear combination at each time step, using the context information that is currently accessible. However, traditional ensemble techniques are unable to reliably identify these switching regions since the context information is not effectively utilized due to the limitations of the model architecture. We emphasize that both the MLP ensemble and the LightGBM achieve a perfect fit for the data. However, for the sake of simplicity, we will only demonstrate the resulting weight combination vector of the MLP ensemble.

2.5.6 Real-Life Data Experiments

We assess the precision of our ensemble framework in relation to the base models that we utilize, traditional ensembling methods and state-of-the-art forecasting methodologies. We showcase our results using three benchmark time series datasets that display complex patterns and a significant reliance on context. The results of our simulations are presented in Table 2.5. Our MLP and LightGBM ensembles outperform all other approaches in all three datasets used in our simulations.

We note that in the M5 Forecasting and Total Natural Gas Demand datasets, the base models have the ability to accurately capture certain patterns in the

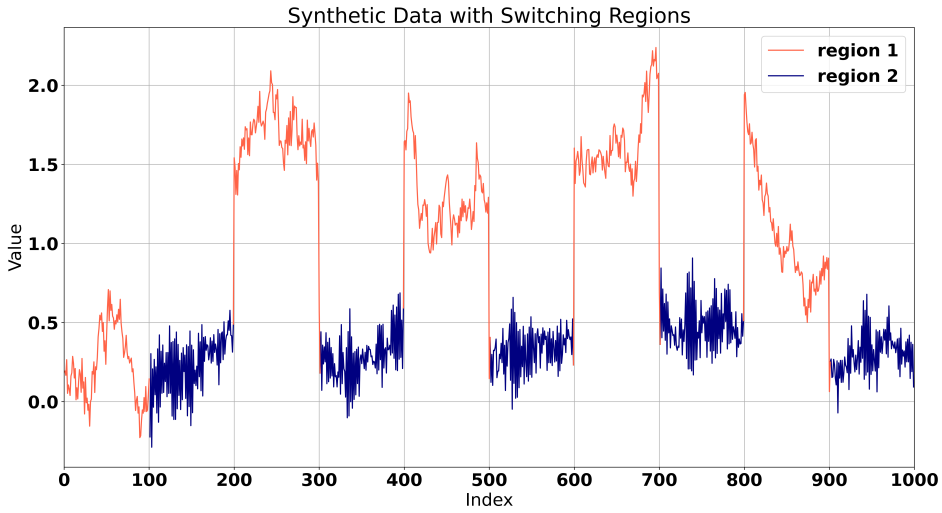


Figure 2.3: The synthetic data consists of two distinct regions that have been generated using a deterministic switching procedure.

data, resulting in better performance compared to other base models in certain regions. Consequently, our ensemble models outperform other models by effectively predicting each location by prioritizing the significance of the relevant base model in those specific areas. The presence of this effect is clear, as there exists a substantial disparity between the performance of our model and the base models. In contrast, the Istanbul Stock Exchange dataset has a smaller feature vector than the other datasets, as indicated in Table 2.1. Furthermore, the time series data is heavily influenced by its most recent observations, as evidenced by the performance of the Naïve model. Therefore, although our ensemble models still surpass other models in performance, the degree of improvement compared to the base models is not as substantial. Moreover, it is crucial to recognize that although we evaluate the effectiveness of our ensemble models by comparing them to the state-of-the-art forecasting models in the literature, these models can also serve as the base learners for our ensemble models. Therefore, our ensemble models can be used to enhance the effectiveness of the state-of-the-art forecasting models, as long as the individual models within the ensemble are both diverse and accurate [15, 16].

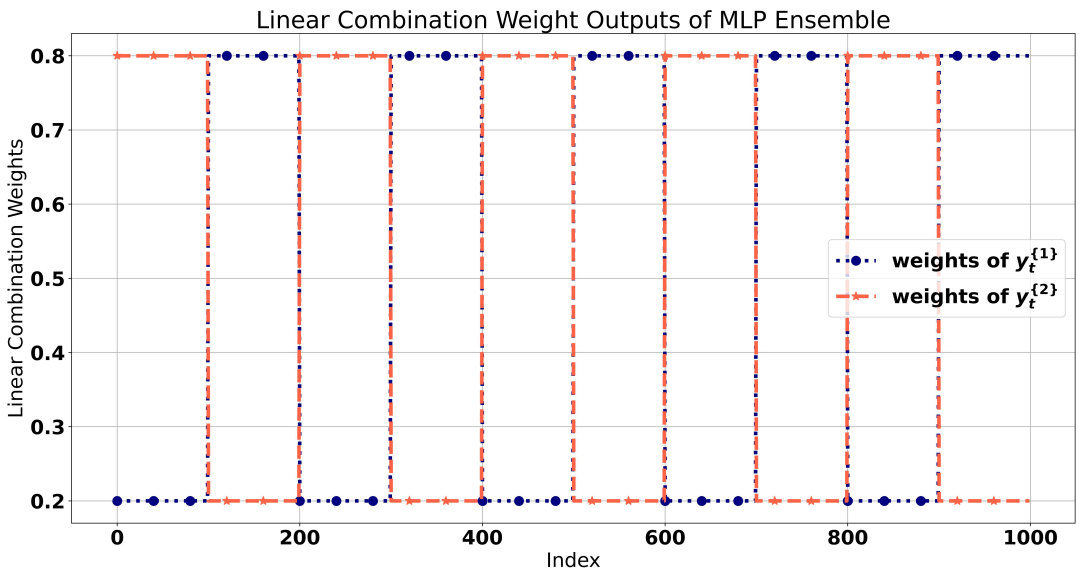


Figure 2.4: The MLP ensemble produces linear combination weight outputs that are subject to a convex constraint. The results from the training and test datasets are combined to ensure visual consistency.

The comparative analysis of our LightGBM and MLP ensembles under all three constraints is presented in Table 2.6. We emphasize that the effectiveness of each constraint is highly reliant on the characteristics of the datasets. For example, in the dataset of the Istanbul Stock Exchange, both ensemble models produce optimal results when the affine constraint is applied. In contrast, the application of the convex constraint in the Natural Gas Demand dataset results in much better performance for both ensembles compared to other configurations. Therefore, we conclude that the optimal weight combination constraint for our ensemble techniques depends on the complexity, context, and size of the datasets. This is apparent from the information provided in Section 2.2 and Appendix A, as all three constraints vary in their restrictions on the search space for linear combinations. While the unconstrained weight scenario theoretically yields the greatest results in the optimal learning scenario, the actual statistics of the target sequence are rarely known. Therefore, restricting the search space may result in better outcomes, as demonstrated in our simulations.

Table 2.3: Search Space of Hyperparameters

Model Name	Search Space
SARIMAX	p:{0,1,2,3}, d:{0,1}, q:{0,1,2,3} P:{0,1,2}, D:{0,1}, Q:{0,1,2}
LightGBM	n_estimators:{100:2500} learning_rate:{0.001:0.1} num_leaves:{7:255} max_depth:{3:7} min_child_samples:{5:50}
MLP	hidden_layer_neurons:{8:128} activation:{"logistic", "tanh", "relu"} learning_rate:{0.0001:0.1} max_iterations:{100:1000}
RNN	n_layers:{1,2,3} learning_rate:{0.0001:0.1} n_epochs:{100:2000} dropout:{0.0:0.5} batch_size:{16:128}
LSTM	n_layers:{1,2,3} learning_rate:{0.0001:0.1} n_epochs:{100:2000} dropout:{0.0:0.5} batch_size:{16:128}
GRU	n_layers:{1,2,3} learning_rate:{0.0001:0.1} n_epochs:{100:2000} dropout:{0.0:0.5} batch_size:{16:128}
Our Ensemble	constraint: {"unconstrained", "convex", "affine"} base_learner_set: {all possible combinations of SARIMAX, LightGBM, MLP and Naïve}
Conventional Ensemble	base_learner_set: {all possible combinations of SARIMAX, LightGBM, MLP and Naïve}

Table 2.4: Ensemble model comparisons using the generated synthetic dataset

Model	MSE	RMSE	MAE
MLP Ensemble	**0.0**	**0.0**	**0.0**
LightGBM Ensemble	**0.0**	**0.0**	**0.0**
Conventional 1	0.18	0.43	0.41
Conventional 2	0.02	0.12	0.15

We show the best results for each metric in bold.

Table 2.5: Real-life dataset results.

Data	Natural Gas Demand			Istanbul Stock Exchange			M5 Forecasting		
Model\Metric	MSE(10^{11})	MAE(10^4)	RMSE(10^4)	MSE(10^{-5})	MAE(10^{-4})	RMSE(10^{-4})	MSE(10^3)	MAE	RMSE
MLP	47.13	119.64	217.09	21.43	121.78	146.4	46.889	154.65	216.54
SARIMAX	24.87	100.12	157.69	12.35	89.12	111.14	40.297	150.65	200.74
LightGBM	18.49	81.04	135.98	12.24	87.83	110.65	34.247	140.27	185.06
Naïve	53.3	126.88	230.88	33.43	143.5	182.83	213.997	351.93	462.6
TBATS	42.12	107.98	205.23	18.1	105.04	134.56	52.606	163.21	229.36
RNN	28.3	106.14	168.23	12.01	89.16	109.59	33.387	139.37	182.72
LSTM	30.19	106.91	173.76	12.3	88.66	110.91	32.37	137.05	179.91
GRU	30.54	97.78	174.75	12.31	88.15	110.96	33.096	137.67	181.92
MLP Ensemble	14.66	75.91	121.09	12.09	**86.96**	109.96	31.367	133.56	177.11
LightGBM Ensemble	**12.21**	**75.5**	**114.95**	**11.55**	87.12	**107.47**	**30.014**	**131.49**	**173.25**
Conventional 1	19.13	82.12	138.32	12.87	91.3	113.46	34.826	142.43	186.62
Conventional 2	18.86	81.66	137.33	15.27	96.48	123.56	36.424	148.64	190.85

We show the best results for each metric in bold. The ensemble weight constraint that yields the highest validation score has been individually chosen for each dataset in our LightGBM and MLP ensembles.

Table 2.6: Evaluation of our ensemble algorithms over all three constraints.

Data	Natural Gas Demand			Istanbul Stock Exchange			M5 Forecasting		
Model\Metric	MSE(10^{11})	MAE(10^4)	RMSE(10^4)	MSE(10^{-5})	MAE(10^{-4})	RMSE(10^{-4})	MSE(10^3)	MAE	RMSE
MLP Ens. Convex	14.66	75.91	121.09	12.12	87.66	110.09	31.544	136.59	177.61
MLP Ens. Affine	19.96	83.18	141.28	12.09	86.96	109.96	31.367	133.56	177.11
MLP Ens. Unconstrained	18.04	80.97	134.32	13.16	92.21	114.73	34.969	142.29	187
LightGBM Ens. Convex	**13.21**	**75.5**	**114.95**	12.09	**86.87**	109.95	**30.014**	**131.49**	**173.25**
LightGBM Ens. Affine	15.99	78.85	126.49	**11.55**	87.12	**107.47**	35.681	143.89	188.89

We show the best results for each metric in bold. The LightGBM ensemble under unconstrained weight condition is not included in this analysis due to its high error rates in comparison to other models.

Chapter 3

Combination of Linear and Nonlinear Models Through Joint Optimization

3.1 Related Work

Real-world time series data frequently exhibit a combination of linear and nonlinear patterns [7]. In such cases, traditional linear models like SARIMAX [36] and nonlinear models like decision trees [9] are insufficient. To address this issue, various models are combined to effectively capture distinct patterns present in real-world data [54]. The most prevalent method utilized is the utilization of ensemble or mixed models. Ensemble models, which integrate the predictions of multiple models, are extensively studied in various domains including signal processing [13,55], speech recognition [56], computational intelligence, statistics, and machine learning [14,34]. There are multiple approaches to designing ensembles. For example, one approach involves training individual models separately and then merging them together [13]. Another approach involves training individual models in a sequential manner, where each model aims to correct the mistakes made by the previous models [33].

Ensembling methods are susceptible to overfitting and underfitting problems that exist in the individual base models [35]. In order to address these problems, various methods are explored, including introducing separate random variations to the training data of each model, employing distinct preprocessing techniques for each model, reducing the size of the training data for each model and utilizing an extra model to merge the predictions of the base models [54]. All of these methods result in sub-optimal solutions since the base models are individually optimized for the data, meaning that the training process is independent of the model combination stage [35]. Nevertheless, our simulations demonstrate that it is essential to optimize both linear and nonlinear models simultaneously, as joint optimization yields advantages for all models. Hence, we present an approach that jointly optimizes the parameters of both a linear and a nonlinear model in order to minimize the final error.

Prior attempts have been made to jointly optimize the base models in ensembling. The negative correlation learning approach [35] aims to train and merge individual models simultaneously by establishing a negative correlation between the error of each model and the rest of the ensemble at each sample. Despite the introduction of interaction among the various models during training, all the base models still make predictions on the same data, and the final prediction is generated by averaging the predictions of the base models. Furthermore, the negative correlation learning model [35] necessitates a pre-established λ term. This term serves as a parameter to regulate the strength of the penalty imposed in the loss function for the correlation between the errors of the individual models and can take any value between 0 and 1. Hence, the value of the λ term has a direct impact on the loss of each individual model for every sample, thereby influencing the learning process of the ensemble method. This results in a biased ensemble structure. It is important to acknowledge that there have been other methods suggested for modeling the λ term [57, 58]. However, these approaches do not completely remove the bias in the structure, but rather decrease it while simultaneously increasing the complexity of the optimization process. Our methodology distinguishes itself from the joint optimization methods discussed in the literature [57, 58] by training both linear and nonlinear models to generate a joint

prediction for the data samples. Therefore, the relationship between the models is not pre-established but acquired through joint optimization. Therefore, we observe a substantial enhancement in the regression performance, as demonstrated in our simulations.

3.2 Preliminaries

First, we provide a literature analysis of the existing two-stage methodologies that specifically aim to combine linear and nonlinear models. Next, we continue by explaining our problem description.

3.2.1 Literature Review

Within the literature on time series forecasting, researchers have explored different methodologies for combining linear and nonlinear models. These methods are based on the assumption that real-life time series data frequently exhibit both linear and nonlinear patterns [59]. In such cases, linear models are insufficient for accurately representing complicated real-life data due to their inability to capture nonlinear patterns in the data. While nonlinear models have the ability to theoretically represent both linear and nonlinear data components, they are challenging to optimize due to their high computational complexity and have a tendency to overfit.

A former ensemble technique that has been used in the past is the two-stage method, where one model is fitted on the residuals (error series) of another model. The ultimate predictions are derived by aggregating the predictions from both models [33, 60–62]. The underlying assumption of this technique is that the real-life time series data may be represented as the combination of its separate linear and nonlinear components in a certain way [59]. This methodology has motivated numerous academics to utilize diverse models for predicting real-world time series in different domains. G. Peter Zhang [33] suggested training a neural network

on the differences between the observed values and the predicted values of a linear ARIMA model and demonstrated substantial enhancements in performance relative to the original models across several datasets. Qiang Wang et al. [60] suggested employing the linear ARIMA (Auto-Regressive Integrated Moving Average) and the nonlinear BPNN (Back Propagation Neural Network) to construct two distinct models, namely ARIMA-BPNN and BPNN-ARIMA, for simulating the carbon emissions of India, China, U.S and EU. Nevertheless, when fitting the models to the data and subsequently to the residuals of the previous model separately, it becomes challenging to guarantee that the linear and nonlinear patterns observed in the data are exclusively captured by the linear and nonlinear models, respectively [59]. Furthermore, the correlation between the predictions of the previous model in the approach and its residuals, as well as the correlation between the predictions of the two models, remains unclear [59]. Therefore, the two models do not completely use the advantages of one another.

To address this issue, several researchers have suggested incorporating an ensembling technique into this mixing approach [59, 63–66]. Once the residuals of the earlier model are fitted by the latter model, an ensemble method is incorporated into the scheme. This approach combines the forecasts of the two models in either a linear or nonlinear manner. Paulo S.G. de Mattos Neto et al. [59] suggested merging the predictions of two distinct models using a Multi-Layer Perceptron (MLP) for the purpose of forecasting time series data in the field of public health. Similarly, Domingos S. de O. Santos Júnior et al. [63] presented two distinct models. In the first model, they utilized a Support Vector Regression (SVR) to merge the predictions of an ARIMA-MLP model. In this case, the MLP model was trained on the residuals of the ARIMA model. In the second model, they employed a Multilayer Perceptron (MLP) to combine the forecasts of an ARIMA-SVR model. In this case, the SVR model was trained on the residuals of the ARIMA model. Paulo S.G. de Mattos Neto et al. [64] suggested training an LSTM (Long Short-Term Memory) on the residuals of multiple linear statistical models, and subsequently merging the predictions of the two models using SVR for wind speed prediction. Diogo M. F. Izidio et al. [65] employed a similar approach to predict energy consumption. They utilized a nonlinear

model to analyze the discrepancies in a seasonal ARIMA model. Additionally, they employed a separate nonlinear model to combine the forecasts from both models. João Fausto Lorenzato de Oliveira et al. [66] examined various methods for integrating the predictions of diverse linear and nonlinear models using PSO (Particle Swarm Optimization). Despite the wide usage of this approach by researchers in numerous domains, the two basic models are still individually optimized for the data. As a result, the training process is independent of the point where the models are combined [35]. Hence, incorporating an ensemble technique to model the connection between the predictions of the initial model and the residual model is inherently sub-optimal. Our simulations demonstrate that it is necessary to optimize both linear and nonlinear models simultaneously during training, as joint optimization yields benefits for all models.

Another approach combining linear and nonlinear models that employs joint optimization among the models is the negative correlation learning [35], where the individual models are trained in the same learning process by negatively correlating the error of each model with the rest of the ensemble at each sample. This is done by introducing an additional penalty to the loss employed in the training of each base model, which evaluates the correlation between the errors of the current base model and the rest of the models in the ensemble. In addition, a λ term is introduced to the introduced component, which is used to adjust the strength of the correlation penalty and can take any value between 0 and 1. One of the main issues with this approach is the determination of the value of the λ term, which directly affects the loss of each individual model, thus changing the learning procedure of the whole ensemble. Even though there have been efforts to reduce the impact of this parameter [57, 58], its effects have not been eliminated but rather reduced while increasing the complexity of the optimization. In addition, even though the negative correlation approach introduces interaction among the individual models during training, all the base models still predict the same data and the final prediction is produced by averaging the predictions of the base models. Thus, conflicting with the assumption of the previous linear and nonlinear model combination approaches the real-life time series data can be modeled as the aggregation of its complementing linear and nonlinear components

[59].

Another method called the negative correlation learning approach [35], combines linear and nonlinear models and utilizes joint optimization among the models. In this approach, the individual models are trained together in a single learning process, with the error of each model being negatively correlated with the rest of the ensemble at each sample. This is achieved by incorporating an extra penalty into the loss function used during the training of each individual base model. This penalty evaluates the relationship between the errors of the current base model and the errors of the other models in the ensemble. Furthermore, a lambda term is incorporated into the introduced component, serving to modulate the intensity of the correlation penalty. This lambda term can range from 0 to 1, depending on the context. An important challenge with this strategy is the estimation of the value of the λ term, which has a direct impact on the loss of each individual model. Consequently, it alters the learning process of the entire ensemble. Despite attempts to mitigate the influence of this factor [57, 58], its impact has not been eliminated but rather reduced, while simultaneously increasing the complexity of the optimization process. Furthermore, despite the introduction of interaction among the individual models during training, the negative correlation approach ensures that all the base models still make predictions on the same data. The final prediction is then generated by averaging the predictions of the base models. Thus, contrary to the assumption made by previous linear and nonlinear model combination approaches that the real-life time series data can be better modeled as the summations of its complementing nonlinear and linear parts [59].

3.2.2 Problem Description

Our research focuses on the nonlinear forecasting of sequential time series data. We observe the target sequence $\{y_t\}_{t\geq 1}$, along with the accompanying side information sequence $\{\boldsymbol{s}_t\}_{t\geq 1}$, where $y_t \in \mathbb{R}$ and $\boldsymbol{s}_t \in \mathbb{R}^m$. At each time t, we make predictions on y_{t+1}, based on $\{y_1, ..., y_t\}$ and $\{\boldsymbol{s}_1, ..., \boldsymbol{s}_t\}$, in a purely online

manner

$$\hat{y}_{t+1} = f\big(\{\ldots, y_{t-1}, y_t\}, \{\ldots, \boldsymbol{s}_{t-1}, \boldsymbol{s}_t\}\big), \tag{3.1}$$

where f is chosen to minimize the total loss L, given as

$$L = \frac{1}{T}\sum_{t=1}^{T} l(y_t, \hat{y}_t), \tag{3.2}$$

for any T, where l is a given differential loss function. The differential loss function that we consider in our case is the squared error loss.

Next, we provide our proposed methodology in Section 3.3.

3.3 Proposed Approach

We first propose our joint optimization approach in Section 3.3.1 and indicate the advantages and disadvantages over the mixture approaches in the literature that focus on combining linear and nonlinear models, based on residual fitting and/or ensemble learning. Next, we introduce the linear and the nonlinear models to be employed in our approach, in Section 3.3.2 and Section 3.3.3, respectively. Finally, training for the joint optimization of the two models is given in Section 3.3.4.

In Section 3.3.1, we present our joint optimization methodology and discuss its merits and limitations compared to existing mixing approaches in the literature that focus on combining linear and nonlinear models using residual fitting and/or ensemble learning. Next, we provide the linear and nonlinear models that are used within our method. The linear model is discussed in Section 3.3.2, while the nonlinear model is covered in Section 3.3.3. Section 3.3.4 provides training for the joint optimization of the two models.

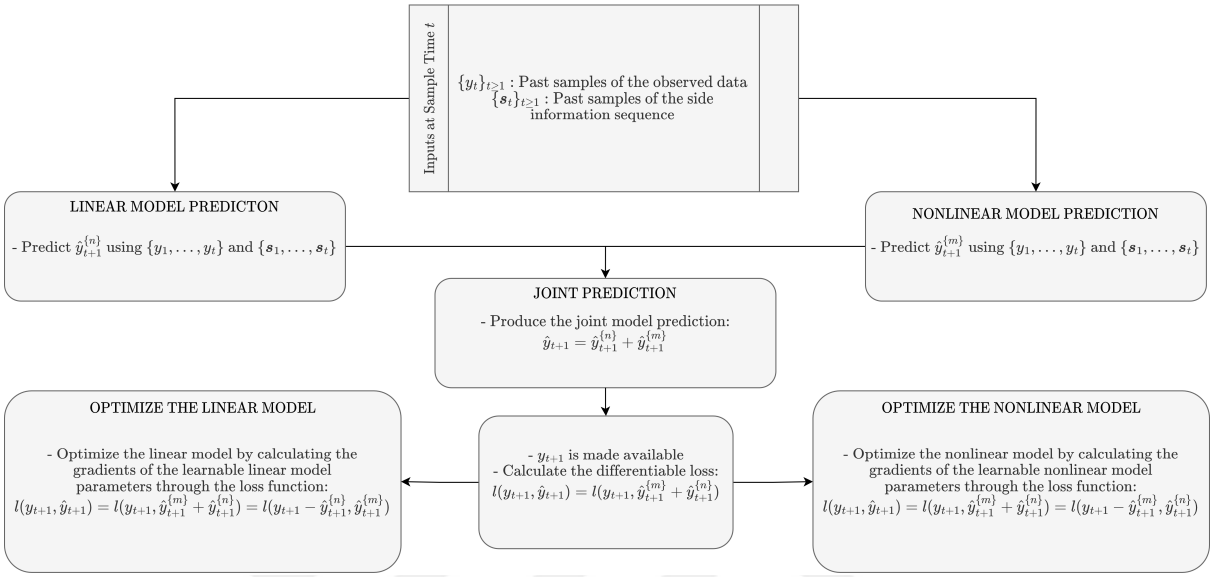


Figure 3.1: The online training of our method that focuses on the joint optimization of both linear and nonlinear models.

3.3.1 Joint Optimization Method

Figure 3.1 shows the online training architecture of our proposed approach. At each time point t, our technique utilizes the observed data $\{y_t\}_{t\geq 1}$ and the past side information vectors $\{\boldsymbol{s}_t\}_{t\geq 1}$ to predict $\hat{y}_{t+1}^{\{n\}}$ and $\hat{y}_{t+1}^{\{m\}}$ using linear and nonlinear models, respectively. Therefore, our combined model prediction is represented as $\hat{y}_{t+1} = \hat{y}_{t+1}^{\{n\}} + \hat{y}_{t+1}^{\{m\}}$. Once the predicted sequence is seen for the sample time $t+1$, denoted as y_{t+1}, we proceed to optimize both the linear and nonlinear models. The objective is to minimize the loss function, which is defined as $l(y_{t+1},\hat{y}_{t+1}) = l(y_{t+1},\hat{y}_{t+1}^{\{m\}} + \hat{y}_{t+1}^{\{n\}})$. Next, we proceed to update/optimize the learnable parameters of each model using a gradient based optimization technique by utilizing the corresponding gradients obtained from the computed differentiable loss function.

We highlight that the optimization of the linear and nonlinear models is based on residual fitting. As we utilize the squared error loss function as l (which can be changed with error metrics such as mean absolute error (MAE) or root mean squared error (RMSE)), the calculated loss for the parameters of the linear model

takes the form $l(y_{t+1}, \hat{y}_{t+1}^{\{m\}} + \hat{y}_{t+1}^{\{n\}}) = l(y_{t+1} - \hat{y}_{t+1}^{\{n\}}, \hat{y}_{t+1}^{\{m\}})$, and similarly, the loss function for the parameters of the nonlinear model takes the form $l(y_{t+1}, \hat{y}_{t+1}^{\{m\}} + \hat{y}_{t+1}^{\{n\}}) = l(y_{t+1} - \hat{y}_{t+1}^{\{m\}}, \hat{y}_{t+1}^{\{n\}})$. This is because the linear model predictions are solely based on the linear model parameters, while the nonlinear model predictions are solely based on the nonlinear model parameters. Therefore, the linear (nonlinear) model's prediction serves as a fixed value for the gradient calculations of the nonlinear (linear) model, as demonstrated in Section 3.3.4.

Therefore, at each sample time t, our model generates the joint prediction $\hat{y}_{t+1}$. Once the observed value y_{t+1} is available, we optimize our linear and nonlinear models using their gradients through a chosen differentiable loss function. Here, the predictions of each individual model directly impact the optimization process of the others, resulting in joint optimization. After completing the optimization for the single sample loss, we proceed to forecast the next sample. As a result, our predictions are generated exclusively in an online manner.

Hence, we represent f in (3.1) as the procedure of generating a joint prediction using both linear and nonlinear models. These models are simultaneously optimized to minimize the overall loss in an online manner, updating f at each time sample or iteration accordingly. Therefore, the appropriate relation may be expressed as:

$$\hat{y}_{t+1} = f\big(\{\ldots, y_{t-1}, y_t\}, \{\ldots, \boldsymbol{s}_{t-1}, \boldsymbol{s}_t\}\big) = \hat{y}_{t+1}^{\{n\}} + \hat{y}_{t+1}^{\{m\}}, \tag{3.3}$$

where f represents the joint optimization approach described in Figure 3.1, $\hat{y}_{t+1}^{\{m\}}$ denotes the forecast of the linear model and $\hat{y}_{t+1}^{\{n\}}$ represents the forecast of the nonlinear model for the time sample $t+1$.

Our proposed method is founded on the well-established residual fitting methods in the literature. Several researchers have put forward various approaches, as detailed in Section 3.2.1, with the primary focus being on residual fitting. Our approach differs from the current residual fitting approaches in the literature as our model directly learns the correlation between the model predictions through joint optimization. For instance, the approach of using the residuals (error series)

of one model to fit another model that is directly fitted to the observed sequence does not effectively capture the learning process and the relationship between the forecasts of the two models [33, 60–62]. Furthermore, while the latter model may accurately represent the residuals of the former model, the former model does not gain any advantages from the learning process of the latter model. However, the incorporation of an ensemble method into the system, which captures the connection between the residual fit model and the previous model [59, 63–66], also presents certain disadvantages. Despite the existence of a model that learns the relationship between the predictions of the two models, our work demonstrates that the learning process between the two models can be directly accomplished through joint optimization. Therefore, the inclusion of an ensemble algorithm is unnecessary, as it also leads to an increase in computing complexity, as seen in our simulations. While some existing approaches in the literature involve interaction between models during the learning process, such as negative correlation learning [35], none of these approaches achieve unbiased interaction between the models. Our joint optimization approach, however, does satisfy this requirement. Nevertheless, a drawback of employing our suggested methodology, in contrast to other investigations, is that the foundational models in our approach necessitate complete differentiability. Therefore, model architectures that lack differentiability, such as the hard decision trees [67], cannot be used directly.

First, we present the models used in our approach. Then, we provide the gradient equations for the proposed joint optimization.

3.3.2 Linear Model

As our linear model, we utilize the widely used model SARIMAX (Seasonal Auto-Regressive Integrated Moving Average with eXogenous factors) created by Box and Jenkins [36]. The SARIMAX model is a statistical method used to predict future values of sequential time series data. It achieves this by utilizing linear relationships between previously observed values of the sequential data, as well as incorporating side-information and error factors. The SARIMAX model is

commonly represented by the orders of its parameters, denoted as SARIMAX $(p, d, q)(P, D, Q)s$, and is defined as

$$\phi(B)\Phi(B^s)\Delta^d\Delta_s^D y_t = c + \theta(B)\Theta(B^s)\epsilon_t + \sum_{i=1}^{m} \beta_i s_{t,i}, \tag{3.4}$$

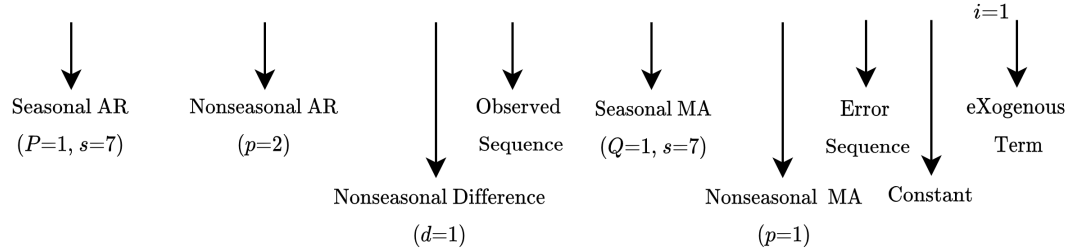
where c represents a constant value, ϵ_t refers to a white noise process, evaluated as the error sequence, B is defined as the back-shift operator, where $B^i y_t = y_{t-i}$ and s represents the seasonality of the model. For instance, $s = 7$ indicates a weekly seasonality pattern in daily data, while $s = 365$ indicates a yearlong pattern. The term Δ refers to the process of differencing applied to the sequential data to reduce the impact of the trend to achieve stationarity. $\Delta^d = (1 - B)^d$ is the nonseasonal difference in the order of d and $\Delta_s^D = (1 - B^s)^D$ is the seasonal difference in the order of D. $\phi(B)$ and $\theta(B)$ represent the nonseasonal auto-regressive and moving average parts of the model with the orders of p and q, respectively. $\Phi(B^s)$ and $\Theta(B^s)$ indicate the seasonal auto-regressive and moving average parts of the model with the orders of P and Q, respectively. The $\phi(B)$, $\theta(B)$, $\Phi(B^s)$ and $\Theta(B^s)$ represent the lag operations given as

$$\begin{aligned}
\phi(B) &= 1 - \phi_1 B - \cdots - \phi_p B^p \\
\theta(B) &= 1 - \theta_1 B - \cdots - \theta_q B^q \\
\Phi(B^s) &= 1 - \Phi_1 B^s - \cdots - \Phi_P B^{sP} \\
\Theta(B^s) &= 1 - \Theta_1 B^s - \cdots - \Theta_Q B^{sQ}.
\end{aligned} \tag{3.5}$$

The summation term $\sum_{i=1}^{m} \beta_i s_{t,i}$ on the right side of (3.4) represents the eXogenous term. It models/captures the relationship between the observed sequence y_t and the side information vector $\boldsymbol{s}_t$. To provide a clearer explanation of the model's structure, an illustration of the SARIMAX $(2, 1, 1)(1, 0, 1)[7]$ model can be represented as:

Note that the parameters of $\phi(B)$, $\theta(B)$, $\Phi(B^s)$ and $\Theta(B^s)$, as well as the constant term c and the side-information coefficients β_i are learned by our joint optimization technique, which is discussed in Section 3.3.1. The orders of these parameters, denoted as $(p, d, q)(P, D, Q)s$, represent the hyperparameters of the model. These hyperparameters need to be determined prior to optimizing the linear model and are crucial for effectively modeling the observed data.

$$(1-\Phi_1 B^7)(1-\phi_1 B-\phi_2 B^2)(1-B)y_t = (1-\Theta_1 B^7)(1-\theta_1 B)\epsilon_t + c + \sum_{i=1}^{m}\beta_i s_{t,i}$$



Remark. *While SARIMAX is our preferred linear model, alternative linear models can also be employed directly. We want to highlight that the Box & Jenkins models family [36] ensures accurate linear modeling of the observed data sequence, as the error series of the models, which is the difference between the model forecasts and the predicted sequence, does not exhibit significant linear patterns in relation to the predicted sequence. Hence, the models belonging to the Box & Jenkins family can be seamlessly substituted with the SARIMAX model in our methodology. It should be noted that this is not a guarantee provided by alternative models, such as decision trees [9] or neural networks [10]. To add other linear models, one just needs to modify the gradient computations in equations* (3.10) *and* (3.11).

3.3.3 Nonlinear Model

We employ regression decision trees as our nonlinear model [68]. Typically, the literature uses hard decision trees [67], which strictly partition the input space using hard decision boundaries [69]. The input vector $\boldsymbol{x}_t$, which in our case is represented as $[\ldots, y_{t-1}, y_t, \ldots, s_{t-1}, s_t]^T$, is initially introduced to the tree at the root node. It is then evaluated by a decision rule, which determines whether it should proceed to the left or right child nodes, referred to as the internal nodes. The decision-making process is iterated at every internal node until it reaches a leaf node. A leaf node is a terminal node that does not have any other nodes attached to it. The leaf node generates a scalar output value, which serves as the model's prediction in time series regression. This output value is obtained by

partitioning the input space.

Hard decision trees are susceptible to overfitting [12] and feature a strict decision rule where only one leaf node contributes to the output. Furthermore, the hard decision tree model's hard decisions result in its intrinsic lack of differentiability. However, for our joint optimization approach to effectively optimize the outputs of both the linear and nonlinear models, as shown in Figure 3.1, it is necessary to have a model structure that is fully differentiable. Thus, we utilize the soft version of decision trees, as the nonlinear model in our method, as described in the paper [37].

Soft decision trees distinguish themselves from hard decision trees by not having a strict rooting at the nodes. Instead, they create probabilities for each child node, allowing all the leaves to contribute to the final output. At every internal node $n \in N$ in the tree, we utilize a decision procedure that generates a Bernoulli random variable p_n. The value of p_n represents the "probability" of the input being routed to the left child node, given by

$$p_n = \sigma(\boldsymbol{w_n}^T \boldsymbol{x}_t + b_n), \tag{3.6}$$

where $\boldsymbol{w_n}$ and b_n are the learnable parameters of the internal node $n \in N$, $\boldsymbol{x}_t$ is the input vector to the tree and σ is the sigmoid function defined as $\sigma(x) = 1/(1+e^{-x})$. Likewise, the probability of assigning/rooting the input to the right child node is determined by $1 - p_n$. The sigmoid function transforms/maps the outputs of the internal nodes into a range of values between 0 and 1, allowing the output values of each internal node to be represented as probabilities.

Figure 3.2 depicts the prediction mechanism of the soft decision tree, specifically showing 4 leaf nodes and 3 internal nodes. The symbols N_1, N_2, and N_3 are used to indicate the first, second and third internal nodes of the tree, respectively. Every node that is not the root node is considered a leaf node. The input vector traverses through each internal node and generates probability. Each leaf node has a "path probability" which is calculated by multiplying the probabilities of all the internal nodes that go to the leaf node, starting from the root node. Unlike hard decision trees, where a single leaf produces the output value, in our case, all

the leaves contribute to the output based on their path probabilities. Therefore, the final output of a single decision tree is determined by

$$
\begin{aligned}
o(\boldsymbol{x}_t) &= \sum_{\ell=1}^{\mathcal{L}} \mathrm{P}_\ell \, \boldsymbol{\phi}_\ell \\
&= \sum_{\ell=1}^{\mathcal{L}} \Big(\prod_{n \in \mathrm{N}(\ell)} p_i \Big) \, \boldsymbol{\phi}_\ell \\
&= \sum_{\ell=1}^{\mathcal{L}} \Big(\prod_{n \in \mathrm{N}(\ell)} \sigma(\boldsymbol{w}_n^T \boldsymbol{x}_t + b_n) \Big) \, \boldsymbol{\phi}_\ell,
\end{aligned}
\tag{3.7}
$$

where $\mathcal{L}$ denotes the set of all leaves of the tree, the path probability of the ℓ^{th} leaf node is represented by P_ℓ, the nodes that lead to the ℓ^{th} leaf node are denoted by $\mathrm{N}(\ell)$ and ϕ_ℓ is the scalar output value generated at the ℓ^{th} leaf node. The probabilities assigned to each internal node $n \in N$ in the tree, given by p_n, enable the creation of a completely differentiable tree structure, where the learnable parameters are $\boldsymbol{w_n}$ and b_n of the internal nodes, and ϕ_ℓ of the leaf nodes. Hence, as depicted in Figure 3.2, every internal node possesses learnable parameters $\boldsymbol{w_n}$ and b_n, and generates probabilities given by p_n. As illustrated by the bold arrows, all leaves contribute to the final output of the soft decision tree by generating the associated leaf output ϕ_l, for a given input vector $\boldsymbol{x}_t$. We also note that the scalar values ϕ_l are also learnable.

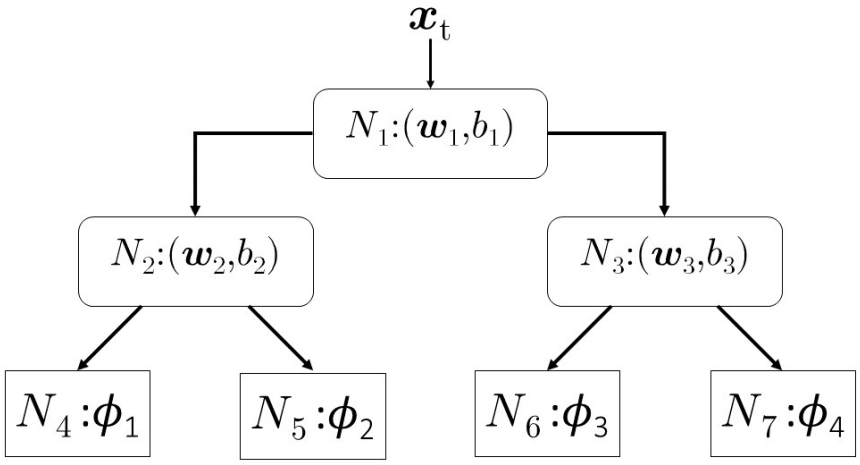


Figure 3.2: The architecture of the soft decision tree.

Many of these soft decision trees are combined to create soft gradient boosting decision trees (Soft GBDT). Each successive tree is fit to the residuals of the previous tree and generates output values based on (3.7). The final output value produced by the Soft GBDT is represented as

$$\hat{y}_{t+1}^{\{n\}} = \sum_{k=1}^{K} \nu o^{(k)}(\boldsymbol{x}_t) + c, \tag{3.8}$$

where K represents the total number of trees, $o^{(k)}$ is the scalar output value produced by the k^{th} soft decision tree, $\hat{y}_{t+1}^{\{n\}}$ is the final output value, the forecast generated by the Soft GBDT. $\boldsymbol{x}_t$ represents the input vector and the regularization parameter, ν, is used to balance the influence of each tree's predictions and prevent overfitting. The constant value c is independent of the input vector that can be chosen as the mean of the sequential time series data y_t, up to the current prediction time $t+1$. We emphasize that each soft decision tree $k \in K$ possesses its own set of learnable parameters $\boldsymbol{w}_{\boldsymbol{n}}^{(k)}$, $b_n^{(k)}$ and $\phi_\ell^{(k)}$.

Hence, we have developed a gradient boosting tree that is completely differentiable. This tree has learnable parameters $\boldsymbol{w}_{\boldsymbol{n}}^{(k)}$ and $b_n^{(k)}$ for every internal node $n \in N$, as well as ϕ_ℓ for each leaf node $\ell \in \mathcal{L}$, for any tree $k \in K$. It is important to consider the depth of the soft decision trees, the regularization parameter ν, and the number of trees in the Soft GBDT as hyperparameters that need to be chosen before the optimization stage. These hyperparameters are crucial in effectively modeling the observed data.

Remark. *The training process of a soft decision tree diverges from that of a hard decision tree due to the inclusion of learnable parameters, specifically weight vectors, at each node. The learnable parameters are initially assigned random values from a normal distribution. These parameters are then tuned at each iteration/sample of the online learning stage to improve the accuracy of the overall Soft GBDT in predicting the observed sequence.*

Remark. *While Soft GBDT is our chosen nonlinear model, alternative nonlinear models like Artificial Neural Networks (ANN), Polynomial Auto-Regressive (PAR) model, etc., can also be directly employed. To add alternative nonlinear models, it is sufficient to modify the gradient calculations in equations (3.12), (3.13), and (3.14).*

3.3.4 Model Training

Therefore, we have distinct structures that can be differentiated for both the linear and nonlinear models, which will be used in our suggested joint optimization technique. For the purpose of joint optimization, we use the stochastic gradient descent technique due to its speed, superior generalization compared to other gradient-based optimization algorithms [70], and its suitability for online learning, where a single sample is utilized for optimization in each iteration. Alternatively, several optimization methods that utilize gradient information, such as Adam, AdaGrad, and others, can be directly utilized as well. It is important to mention that both models are optimized for each individual data sample. This optimization is achieved using online learning, as described in Section 3.3.1 and depicted in Figure 3.1. Below, we present the gradient calculations for both the linear and nonlinear models.

The linear SARIMAX and the nonlinear Soft GBDT models are jointly optimized using stochastic gradient descent in an online manner. The loss that our joint model minimizes at time t is given as

$$l(y_t, \hat{y}_t), \tag{3.9}$$

where $\hat{y}_t = \hat{y}_t^{\{m\}} + \hat{y}_t^{\{n\}}$. $\hat{y}_t^{\{m\}}$ and $\hat{y}_t^{\{n\}}$ represent the linear and nonlinear prediction components of the joint model, respectively. In the case of the squared error loss, which is the error metric used in our simulations, the loss equation (3.9) can be expressed as

$$l(y_t, \hat{y}_t) = \frac{1}{2}(y_t - \hat{y}_t)^2.$$

Our model consists of two components: a linear model and a nonlinear model. When optimizing both models together, the parameters of both models are updated depending on the joint prediction error made by the two models. The linear model in our technique is defined by the structure given in equation (3.4), where the parameters to be estimated are

$$\boldsymbol{w} = \left[\phi_1, \ldots, \phi_p, \Phi_s, \ldots, \Phi_{sP}, \theta_1, \ldots, \theta_q, \Theta_s, \ldots, \Theta_{sQ}, \beta_1, \ldots, \beta_m\right]^T,$$

where ϕ_i denote the coefficients of the autoregressive terms with order p. Φ_i represent the coefficients of the seasonal autoregressive terms with order P, θ_i represent the coefficients of the moving average terms with order q, Θ_i represent the coefficients of the seasonal moving average terms with the order Q, seasonality term is given as s and m represent the length of the given eXogenous data. The size of these parameters is contingent upon the provided time series data and serves as the hyperparameters of the model that necessitate adjustment for precise data prediction. The weight vector $\boldsymbol{w}$ comprises all the adjustable parameters of the model that need to be updated. The gradients for the parameters in $\boldsymbol{w}$ are provided as

$$\begin{aligned}\frac{\partial l(y_t, \hat{y}_t)}{\partial w_j} &= \frac{\partial l(y_t, \hat{y}_t)}{\partial \hat{y}_t^{\{m\}}} \frac{\partial \hat{y}_t^{\{m\}}}{\partial w_j} \\ &= (\hat{y}_t - y_t) \frac{\partial \hat{y}_t^{\{m\}}}{\partial w_j} \\ &= (\hat{y}_t^{\{n\}} + \hat{y}_t^{\{m\}} - y_t) \frac{\partial \hat{y}_t^{\{m\}}}{\partial w_j},\end{aligned} \tag{3.10}$$

where w_j represents the j_{th} term in $\boldsymbol{w}$ that is to be estimated. The $\frac{\partial \hat{y}_t^{\{m\}}}{\partial w_j}$ term given in the equation is simply the term that w_j is multiplied with in (3.4). For instance, when there is no differencing or seasonal differencing in the equations, the gradients for $\boldsymbol{w}$ are provided as

$$\frac{\partial \hat{y}_t^{\{m\}}}{\partial \boldsymbol{w}} = \left[y_{t-1}, \ldots, y_{t-sP}, \epsilon_{t-1}, \ldots, \epsilon_{t-sQ}, s_{t,1}, \ldots, s_{t,m} \right]. \tag{3.11}$$

The result of the nonlinear model in our technique is provided as (3.8), where the equation for the result of each soft decision tree is supplied by (3.7). In contrast to the linear model, our technique utilizes a nonlinear model that comprises numerous individual weak learners, which collectively contribute to an ensemble outcome. Hence, we need to take into account the loss associated with each individual soft decision tree in the model. The loss is determined by

$$E_{total} = \frac{1}{2} \sum_{k=1}^{K} (r^{(k)} - \nu o^{(k)})^2, \tag{3.12}$$

where $o^{(k)}$ represents the output of k^{th} tree, ν gives the regularization parameter

and $r^{(k)}$ represents the residual entering the k^{th} tree, formed by aggregating the residuals from the previous trees up to the k^{th} tree. For any k^{th} tree, $r^{(k)}$ can be expressed as

$$r^{(k)} = y_t - \hat{y}_t^{\{m\}} - \nu o^{(0)} - \nu o^{(1)} - \cdots - \nu o^{(k-1)}.$$

We now represent the gradient equations based on the total loss function (3.12). The learnable parameters for the k^{th} soft decision tree consist of the leaf node outputs $\phi_\ell^{(k)}$ for each leaf node $\ell \in \mathcal{L}$, where $\mathcal{L}$ represents the set of leaves of the tree. Additionally, the weight term $\boldsymbol{w_n}^{(k)}$ and the bias term ${b_n}^{(k)}$ are included for each internal node $n \in \mathcal{N}$, where $\mathcal{N}$ is the set of internal nodes of the tree. The gradients of the total loss with respect to these parameters are provided as

$$\frac{\partial E_{total}}{\partial \phi_\ell^{(k)}} = \mathrm{P}_\ell^{(k)} \sum_{i=k}^{K} (\nu o^{(i)} - r^{(i)}) \tag{3.13}$$

$$\begin{aligned} \frac{\partial E_{total}}{\partial \boldsymbol{w}_n^{(k)}} = \Bigg(\sum_{\ell^{(k)} \in \mathrm{D}(n^{(k)})} \phi_\ell^{(k)} \frac{\mathrm{P}_\ell^{(k)} \mathrm{p}_n^{(k)} (1 - \mathrm{p}_n^{(k)})}{\mathrm{p}_n^{(k)} - \rho(\ell^{(k)})} \boldsymbol{x}_t \\ \sum_{i=k}^{K} (\nu o^{(i)} - r^{(i)}) \Bigg), \end{aligned} \tag{3.14}$$

$$\begin{aligned} \frac{\partial E_{total}}{\partial b_n^{(k)}} = \Bigg(\sum_{\ell^{(k)} \in \mathrm{D}(n^{(k)})} \phi_\ell^{(k)} \frac{\mathrm{P}_\ell^{(k)} \mathrm{p}_n^{(k)} (1 - \mathrm{p}_n^{(k)})}{\mathrm{p}_n^{(k)} - \rho(\ell^{(k)})} \\ \sum_{i=k}^{K} (\nu o^{(i)} - r^{(i)}) \Bigg), \end{aligned}$$

where K represent the total number of soft decision trees in the model, $\mathrm{D}(n^{(k)})$ refers to the set of all leaf nodes that descend from the internal node $n^{(k)}$, $\boldsymbol{x}_t$ is the input shared by all the soft decision trees of the model and $\rho(\ell^{(k)})$ is the piece-wise function

$$\rho(\ell^{(k)}) = \begin{cases} 0, & \text{if } \ell^{(k)} \in \mathrm{LD}(\mathrm{n}^{(k)}) \\ 1, & \text{if } \ell^{(k)} \in \mathrm{RD}(\mathrm{n}^{(k)}), \end{cases}$$

where $\mathrm{LD}(n^{(k)})$ represent the set of leaf nodes that descend from the left branch of the internal node n, and $\mathrm{RD}(n^{(k)})$ represent the set of leaf nodes that descend from the right branch of the internal node n. Hence, we have the equality $\mathrm{LD}(n^{(k)}) + \mathrm{RD}(n^{(k)}) = \mathrm{D}(n^{(k)})$.

Finally, the updates of the parameters of both models are provided by

$$\begin{aligned}
\boldsymbol{w} &= \boldsymbol{w} - \mu_m \frac{\partial l(y_t, \hat{y}_t)}{\partial \boldsymbol{w}} \\
\phi_\ell^{(k)} &= \phi_\ell^{(k)} - \mu_n \frac{\partial E_{total}}{\partial \phi_\ell^{(k)}} \\
\boldsymbol{w}_n^{(k)} &= \boldsymbol{w}_n^{(k)} - \mu_n \frac{\partial E_{total}}{\partial \boldsymbol{w}_n^{(k)}} \\
b_n^{(k)} &= b_n^{(k)} - \mu_n \frac{\partial E_{total}}{\partial b_n^{(k)}},
\end{aligned}$$

where μ_m and μ_n are the hyperparameters of the stochastic gradient descent algorithm for the linear and nonlinear models that control the rate at which the linear and nonlinear models adapt to the optimization problem. Consequently, we have derived the gradient equations for both the linear and nonlinear models in our combined optimization method.

Note that, the number of parameters to be optimized for each model is important, as it determines the complexity of the optimization problem, which depends on the hyperparameters of the models. Given a SARIMAX $(p, d, q)(P, D, Q)s$ model, the number of parameters(constants) to be optimized is given as $p+q+P+Q+m+1$, where p, q, P, Q determine the order of the auto-regressive and moving average terms of the model, m is the size of the side information vector at any sample time and the additional 1 comes from the constant c in the model. Similarly, consider a Soft GBDT model with K soft decision trees, where each tree consists of $\mathcal{L}$ number of leaves, $N = \mathcal{L} - 1$ number of internal nodes. Note that each soft decision tree $k \in K$ has its own learnable parameters $\boldsymbol{w}_{\boldsymbol{n}}^{(k)}$ of size Nm, $b_n^{(k)}$ of size N and $\phi_\ell^{(k)}$ of size L. Therefore, the number of learnable parameters(constants) of a Soft GBDT model is given as $K(Nm + N + \mathcal{L}) = K((\mathcal{L} - 1)(m + 1) + \mathcal{L})$.

We emphasize that joint optimization does not increase the number of parameters to be optimized, as the number of parameters to be optimized for the joint optimization approach is the sum of the number of parameters(constants) of the individual models in the approach. However, joint optimization does increase the time of the search for the optimal hyperparameters on the same level as a simple residual fitting operation, as both the hyperparameters of linear and nonlinear models should be set in both cases.

It is vital to consider the number of parameters to be optimized for each model, as this directly affects the difficulty of the optimization problem. The complexity is determined by the hyperparameters of the models. The SARIMAX $(p, d, q)(P, D, Q)s$ model requires optimizing a total of $p + q + P + Q + m + 1$ parameters. Here, p, q, P, and Q determine the order of the auto-regressive and moving average terms of the model, m represents the size of the side information vector at any sample time, and the additional 1 comes from the constant c in the model. Similarly, let's examine a Soft GBDT model that includes K soft decision trees. Each tree is composed of $\mathcal{L}$ leaves and $N = \mathcal{L} - 1$ internal nodes. It is important to understand that each soft decision tree, denoted as $k \in K$, possesses its own parameters that are to be learned. The vector $\boldsymbol{w}_{\boldsymbol{n}}^{(k)}$ has a size of Nm, the vector $b_n^{(k)}$ has a size of N, and the vector $\phi_\ell^{(k)}$ has a size of L. The formula for calculating the number of learnable parameters (constants) in a Soft GBDT model is expressed as $K(Nm+N+\mathcal{L}) = K((\mathcal{L}-1)(m+1)+\mathcal{L})$. It is important to note that joint optimization does not result in an increase in the number of parameters that need to be optimized. In this approach, the total number of parameters to be optimized is simply the sum of the parameters (constants) of the individual models involved. However, joint optimization does result in an increase in the duration of the search for the optimal hyperparameters, comparable to that of a straightforward residual fitting procedure. This is because, in the joint optimization case, the hyperparameters of both linear and nonlinear models need to be determined at the same time.

The next section demonstrates the experiments of our jointly optimized model.

Model Name	Explanation
SARIMAX	The fully differentiable SARIMAX base model
Soft GBDT	The fully differentiable Soft GBDT base model
Joint Approach	Our joint optimization approach introduced in Section 3.3.1 and Figure 3.1.
literature_model_1	Soft GBDT is fit on the residuals of SARIMAX, then the predictions of both models are summed for the final prediction.
literature_model_2	SARIMAX is fit on the residuals of Soft GBDT, then the predictions of both models are summed for the final prediction.
literature_model_3	SARIMAX and Soft GBDT are both fit on the data independently. Then a Linear Regressor is used to combine the predictions of the models.
literature_model_4	SARIMAX and Soft GBDT are both fit on the data independently. Then a 2-layer MLP is used to combine the predictions of the models.
literature_model_5	SARIMAX is fit on the residuals of Soft GBDT. Then a 2-layer MLP is used to combine the predictions of the models.
literature_model_6	Soft GBDT is fit on the residuals of SARIMAX. Then a 2-layer MLP is used to combine the predictions of the models.

Table 3.1: The descriptions of the models used in our experiments.

3.4 Simulations

This section showcases the performance of our joint model in various circumstances. We compare our technique with other mixing methods in the literature that focus on integrating linear and nonlinear models using residual fitting and/or ensemble learning. We employ a total of five distinct models, each of which is described in Table 3.1. It is important to mention that, for each of the mixture techniques, we utilize the SARIMAX and Soft GBDT models from our suggested methodology to ensure fairness. To avoid repetition, we use the names provided in Table 3.1 to refer to the given models.

In our simulations, we initially create artificial data that has both strong linear and nonlinear components. We then confirm the need to employ a combination of linear and nonlinear models rather than using them separately. In the subsequent section, we demonstrate the efficacy of our model in practical situations by analyzing the daily data of the overall residential natural gas consumption in Turkey from 2018 to 2020, along with the M5 Forecasting Dataset provided by Walmart (Makridakis et al., 2022). In all scenarios, we only focus on one-step-ahead forecasting and evaluate different models based on the total loss. The loss function, denoted as l, is assessed using various error metrics such as Mean Squared Error (MSE), Mean Absolute Percentage Error (MAPE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). We additionally demonstrate the cumulative normalized total error over time for all the models employed in the experiments, in order to illustrate the online learning progression of the model, given by

$$\sum_{k=1}^{t} \frac{(y_k - \hat{y}_k)^2}{t}, t = 1, \ldots, T,$$

where y_k represents the sample of the signal that needs to be forecasted, whereas $\hat{y}_k$ represents the prediction made by the corresponding model.

During the training of our models in simulations, we initially optimize the hyperparameters without using the test dataset, which comprises the final 20% of the data. We optimize the hyperparameters for each model in the simulations to minimize the total loss, measured by the squared error loss, on the first 80% of the data. It is important to note that the predictions are based on Figure 3.1, ensuring that there are no data leakage issues. Once the hyperparameters have been determined, we proceed to retrain the models using the optimal set of hyperparameters. This retraining is done on the full dataset using online learning, as described in Section 3.3.1 and illustrated in Figure 3.1. Next, we demonstrate and evaluate the effectiveness of our models by employing several error metrics. The range of hyperparameters for the models to be employed in our simulations is provided in Table 3.2.

SARIMAX	p:{0,1,2}, d:{0,1}, q:{0,1,2} P:{0,1,2}, D:{0,1}, Q:{0,1,2}, μ_n:{0.0005:0.1}
Soft GBDT	K:{5,10,25,50,100,200}, L:{2,4,8,16}, μ_m:{0.0005:0.1}, ν:{$\frac{1}{K}$:$\frac{10}{K}$}
MLP Ensemble	hidden layer length:{8,16,32,64,128}, number of epochs:{100,250,500,1000,2000}, learning rate:{0.005:0.5}

Table 3.2: The hyperparameter search space of the models utilized in our experiments.

We emphasize that, for the models discussed from the literature (reviewed in Section 3.2.1) which utilize an extra ensemble model to merge base model predictions (literature_model_3, literature_model_4, literature_model_5, literature_model_6), the ensembling method is not trained with online learning for fairness to the models. This is because the ensemble models we use (MLP and Linear Regressor) are not trained with online learning in the studies mentioned. Consequently, once we have completed the online learning of the base models, we proceed to train the ensemble models using the initial 80% of the data. Subsequently, we evaluate and showcase the performance of these ensemble models using the remaining 20% of the data.

3.4.1 Synthetic Data

The synthetic data comprises manually created linear and nonlinear data components. The equation for linear data is expressed as

$$y_t^{\{m\}} = m_1 y_{t-1}^{\{m\}} + (1 - m_1) y_{t-2}^{\{m\}} + v_t^{\{m\}},$$

where $v_t^{\{m\}}$ represents a sample function from a stationary white Gaussian process with variance 1 and m_1 is a uniformly distributed random variable between 0 and 1. The nonlinear data is produced by a piecewise linear model and is denoted as

$$
\begin{aligned}
y_t^{\{n\}} &= n_1 y_{t-1}^{\{n\}} + n_2 y_{t-2}^{\{n\}} + v_t^{\{n\}} \quad \text{, if } \; y_{t-1}^{\{n\}} > 0 \; \text{ and } \; y_{t-2}^{\{n\}} > 0 \\
y_t^{\{n\}} &= n_2 (y_{t-1}^{\{n\}})^2 - n_2 y_{t-2}^{\{n\}} + v_t^{\{n\}} \quad \text{, if } \; y_{t-1}^{\{n\}} > 0 \; \text{ and } \; y_{t-2}^{\{n\}} < 0 \\
y_t^{\{n\}} &= (n_3 + n_1) y_{t-1}^{\{n\}} + n_3 (y_{t-2}^{\{n\}})^2 + v_t^{\{n\}} \quad \text{, if } \; y_{t-1}^{\{n\}} < 0 \; \text{ and } \; y_{t-2}^{\{n\}} > 0 \\
y_t^{\{n\}} &= (n_1 + n_2) y_{t-1}^{\{n\}} - n_1 y_{t-2}^{\{n\}} + v_t^{\{n\}} \quad \text{, if } \; y_{t-1}^{\{n\}} < 0 \; \text{ and } \; y_{t-2}^{\{n\}} < 0,
\end{aligned}
$$

where $v_t^{\{n\}}$ is a sample function from a stationary white Gaussian process with unit variance and n_1, n_2 and n_3 are uniformly distributed random variables between 0 and 1. Ultimately, we produce our synthetic data using

$$
y_t = y_t^{\{m\}} + y_t^{\{n\}},
$$

where the data has a length of 1000 and the models' accuracy is assessed based on the final 200 samples of the data. We produce 10 random instances of y_t, excluding cases that diverge, and evaluate the performance of the models using different error measures throughout 10 experiments.

By utilizing the synthetic dataset, we fit the linear, nonlinear and joint models on the synthetically generated data. Note that, this data comprises both linear and nonlinear components. Figure 3.3 depicts the models' performance in terms of the cumulative normalized squared error over time, specifically on the test data. In Table 3.3, we present a performance comparison of our joint approach and the basic models using several error measures. Note that the comparison is based on an average of 10 randomly generated synthetic datasets. Our empirical analysis demonstrates that our joint model surpasses all other models, indicating that our joint optimization strategy effectively enhances the learning of data components compared to the base models. Therefore, the process of jointly optimizing both models is advantageous.

Now, we demonstrate the effectiveness of our joint optimization method using real-life datasets. Additionally, we compare our approach with state-of-the-art model mixture methods found in existing literature.

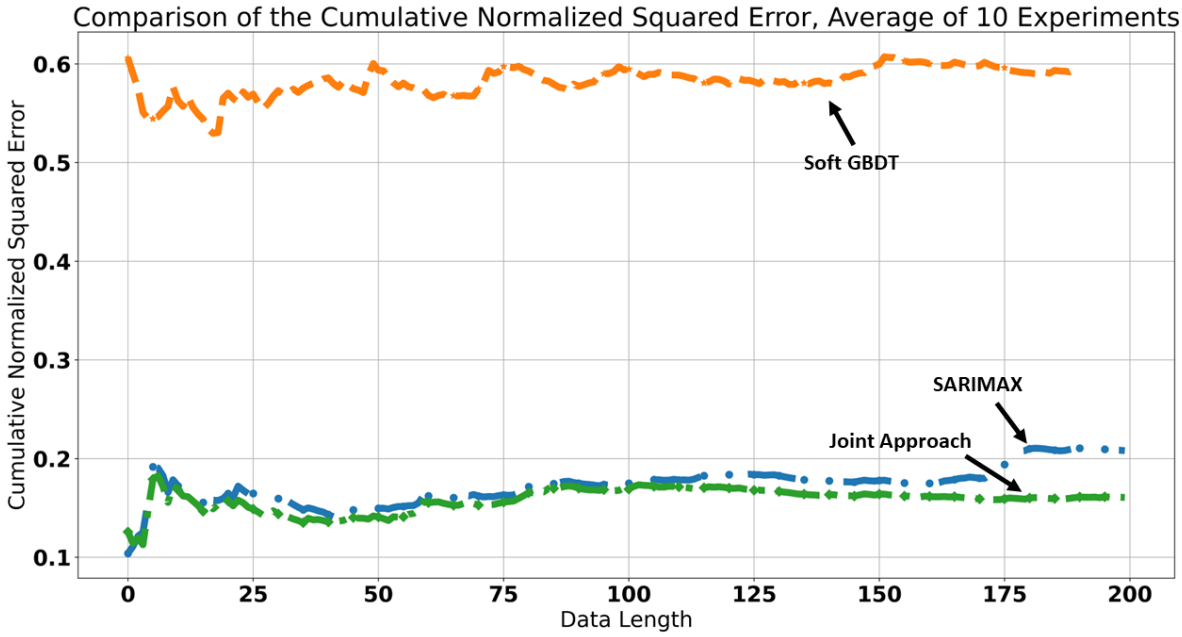


Figure 3.3: Comparison of the cumulative normalized squared error for the synthetic data of our jointly optimized model with the base models.

3.4.2 Total Residential Natural Gas Demand in Turkey

The data utilized in this part comprises the total residential natural gas consumption in Turkey during a span of 1000 days from 2018 to 2020. Every day is treated as an individual instance, and the data is measured on a logarithmic scale. The data exhibits both linear and nonlinear properties, making it a suitable choice for our model studies. We assess the precision of the model on the test set, which comprises the most recent 200 samples of the data. The Soft GBDT error is not shown in Figure 3.4 due to its notably inferior performance compared to other methods. It should be noted that the model mixtures that include the two basic approaches perform worse than the linear SARIMAX model because of the significant inaccuracy of the Soft GBDT. Nevertheless, our joint optimization method does not encounter this problem and surpasses both the original models and other combination models in terms of various error metrics, as seen in Table 3.4.

Model Name	MSE	RMSE	MAE	MAPE
SARIMAX	0.20809	0.44630	0.33566	0.13314
Soft GBDT	0.59441	0.76619	0.61431	0.19306
Joint Approach	**0.16067**	**0.39696**	**0.30932**	**0.12163**

Table 3.3: The performance of the models used in the synthetic datasets is compared using several error metrics.

Model Name	MSE	RMSE	MAE	MAPE
SARIMAX	0.004337	0.06586	0.04989	0.002957
Soft GBDT	0.0133116	0.11538	0.07293	0.004738
Joint Approach	**0.004178**	**0.06463**	**0.04951**	**0.002936**
literature_model_1	0.004559	0.06752	0.05055	0.002989
literature_model_2	0.005163	0.07185	0.05341	0.003163
literature_model_3	0.005205	0.07215	0.05213	0.003067
literature_model_4	0.005241	0.07239	0.05395	0.003174
literature_model_5	0.004826	0.06947	0.04845	0.002864
literature_model_6	0.004431	0.06657	0.04927	0.002915

Table 3.4: The comparison of models that are employed in our simulations of the total residential natural gas demand in Turkey.

3.4.3 M5 Forecasting Data

The M5 Forecasting Data is the dataset used in the M5 competition, which is the fifth competition held by the Makridakis Open Forecasting Centre (MOFC), widely cited in the prediction literature [18]. The dataset consists of the unit sales of products sold in the USA by the retail company Walmart. There are a total of 3049 products, which are classified into 3 different categories and 7 different departments. The products are sold in 10 different Walmart stores in 3 different states. For our experiments, we use the total number of product sales in store CA_1 and department FOODS_3, which is 1941 days long and each day is

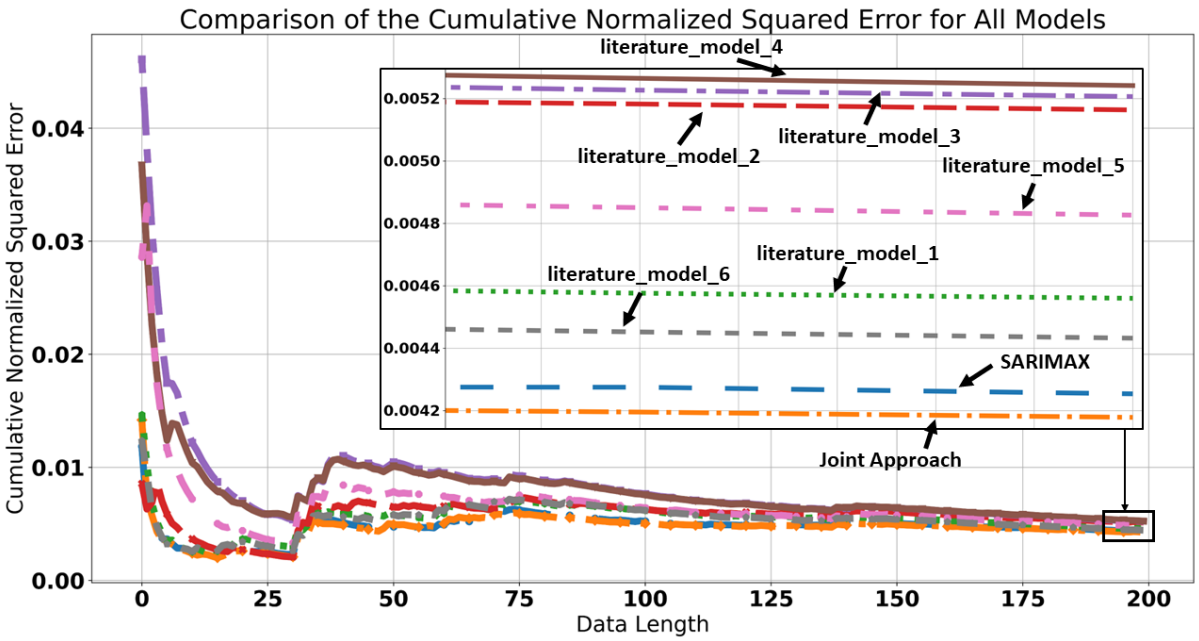


Figure 3.4: An analysis of the accumulated error in the natural logarithmic scale is conducted to compare the total residential natural gas demand in Turkey. The performance of our jointly optimized model, the base models and the mixture methods are illustrated. Note that the error plot for the Soft GBDT model is not included in this context due to the high error rate compared to other techniques.

evaluated as a single sample. As illustrated in Figure 3.5, the data shows strong weekly seasonality and also nonlinearity in certain regions. Therefore, the data is a strong candidate for our experiments. We illustrate only the last 100 samples, for visual clarity. We evaluate the model accuracy on the test set, consisting of the last 400 samples of the data.

The M5 Forecasting Data is the dataset utilized in the M5 competition, the fifth edition organized by the Makridakis Open Forecasting Centre (MOFC), which is well referenced in the field of prediction literature [18]. The dataset comprises the unit sales of products sold by the retail company Walmart in the United States. There is a grand total of 3049 products, which are classified into 3 distinct categories and 7 separate departments. The products are distributed throughout 10 Walmart stores across 3 distinct states. Our tests utilize the cumulative sales of products at store CA_1 and department FOODS_3, which

is randomly selected. The data spans 1941 days, with each day serving as an individual sample. The data, as depicted in Figure 3.5, exhibits strong weekly seasonality and also displays nonlinearity in certain areas. Hence, the data is highly suitable for our experiments. For the sake of visual clarity, we only depict the most recent 100 samples. We evaluate the model accuracy on the test set, which comprises the most recent 400 samples of the data.

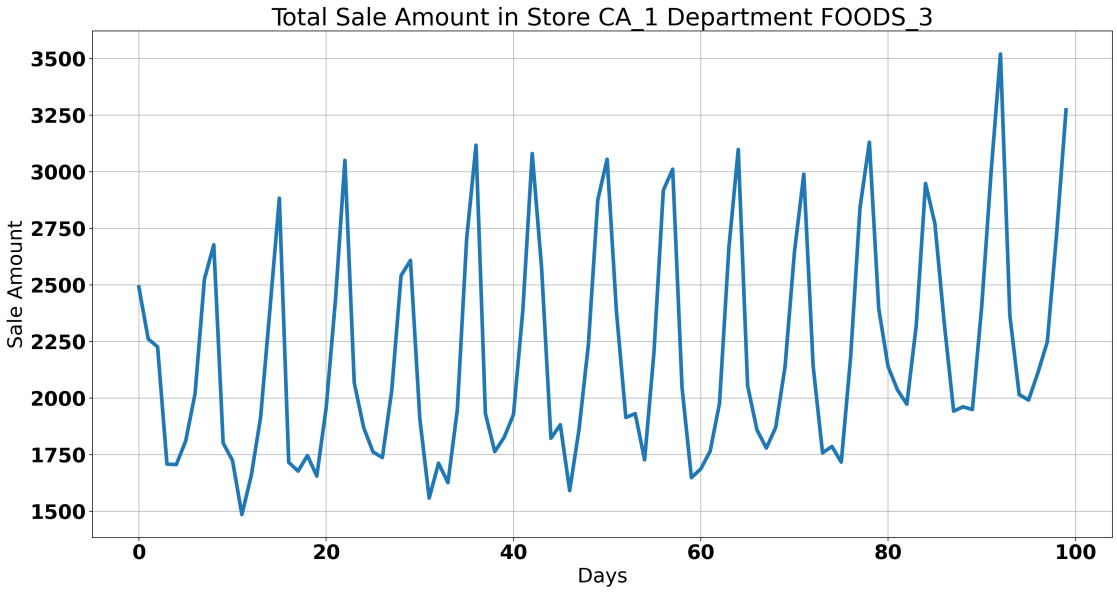


Figure 3.5: The overall sales count for the CA_1 store's FOODS_3 department at Walmart, USA. Only the test set is provided for visual clarity.

We fit our linear and nonlinear base models, our jointly optimized model, and the mixture approaches to the data. The cumulative error for these models is compared in Figure 3.6, and their errors on the test set under different error metrics are shown in Table 3.5. We note that the error metric MAPE cannot be compared due to the presence of zero values in the data. Our collaboratively optimized model surpasses all the models in the literature that are based on residual fitting. The only exception is the MLP ensemble combination approach literature_model_6, which outperforms our model.

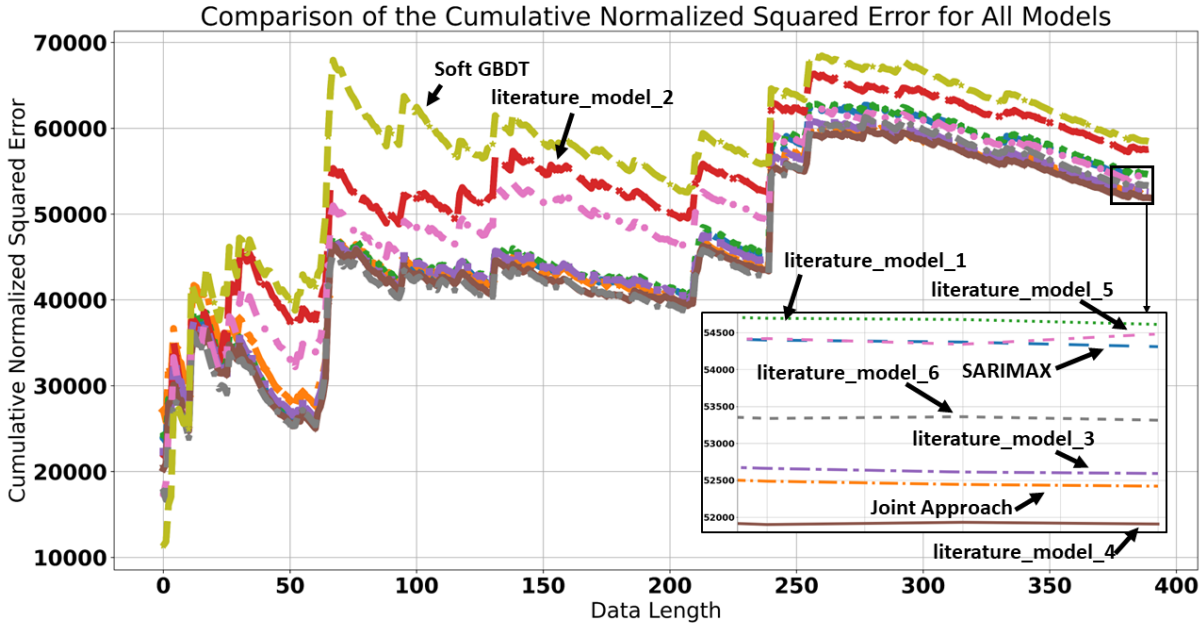


Figure 3.6: An analytic comparison of the cumulative error for the provided M5 data. The results of our joint approach, the base models, and the mixture techniques from the literature are displayed. The error plots commence with the 10th sample to enhance the clarity of the display.

Model Name	MSE	RMSE	MAE
SARIMAX	54309.38	233.04	165.85
Soft GBDT	58460.0	241.79	174.97
Joint Approach	52419.90	228.95	163.98
literature_model_1	54611.06	233.69	166.40
literature_model_2	57555.69	239.91	174.87
literature_model_3	52591.76	229.33	163.84
literature_model_4	**51907.17**	**227.83**	**160.52**
literature_model_5	54481.61	233.41	169.58
literature_model_6	53313.32	230.90	162.75

Table 3.5: The comparison of model performances used in simulating the total unit sales in store CA_1 department FOODS_3 of Walmart, USA.

Chapter 4

Conclusion

In this thesis, we studied the problem of predicting sequential time series data through the mixture/combination of multiple machine learning models. In this context, we introduced two novel ensembling methods specifically tailored for time series forecasting. First, we studied the linear combination of predictions of multiple diverse machine learning models using a novel ensemble framework. To this end, for the first time in the literature, we solved the problem of finding the optimal combination weights of the base learners both in a time-varying and context-dependent manner. Based on this setting, we introduced two novel and generic ensemble algorithms to find the optimal combination weight vectors under unconstrained, affine constrained and convex constrained conditions. Additionally, we introduced a novel training scheme centered around solving a weight optimization problem that minimizes a specific loss function. This approach also mitigates potential issues related to the training of meta-learners, such as co-linearity, high correlation, and overfitting to the base algorithms. Next, we introduced a novel joint optimization approach consisting of linear and nonlinear models. In order to capture both the linear and nonlinear properties of real-life sequential data, we modeled the underlying data as an ensemble/combination of linear and nonlinear models, where we jointly optimized the parameters of both models in order to minimize the final regression error. Thus, we leveraged both models while avoiding the well-known overfitting and underfitting issues

associated with nonlinear and linear models, respectively. We used the widely known SARIMAX for the linear model and employed the soft gradient boosted decision trees (Soft GBDT) for the nonlinear model. We used the stochastic gradient descent algorithm to jointly optimize the two models and provided the related gradient calculations as part of our work. Through various experiments involving synthetic and well-known real-life sequential data, we showed significant performance gains achieved by our two distinct ensembling methods, compared to the base models used in the experiments, conventional ensembling methods and state-of-the-art forecasting models in the literature.

Bibliography

[1] A. Fazla, M. E. Aydin, and S. S. Kozat, "Joint optimization of linear and nonlinear models for sequential regression," *Digital Signal Processing*, vol. 132, p. 103802, 2023.

[2] A. Fazla, M. E. Aydin, and S. S. Kozat, "Time-aware and context-sensitive ensemble learning for sequential data," *IEEE Transactions on Artificial Intelligence*, 2023.

[3] X. Wang, H. Zhang, Y. Zhang, M. Wang, J. Song, T. Lai, and M. Khushi, "Learning nonstationary time-series with dynamic pattern extractions," *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 5, pp. 778–787, 2022.

[4] R. Lou, Z. Lv, and M. Guizani, "Wave height prediction suitable for maritime transportation based on green ocean of things," *IEEE Transactions on Artificial Intelligence*, vol. 4, no. 2, pp. 328–337, 2023.

[5] W. Liao, Z. Yang, X. Chen, and Y. Li, "Windgmmn: Scenario forecasting for wind power using generative moment matching networks," *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 5, pp. 843–850, 2022.

[6] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to linear regression analysis*. John Wiley & Sons, 2021.

[7] E. E. Kuruoğlu, "Nonlinear least lp-norm filters for nonlinear autoregressive α-stable processes," *Digital Signal Processing*, vol. 12, no. 1, pp. 119–142, 2002.

[8] A. C. Singer, G. W. Wornell, and A. V. Oppenheim, "Nonlinear autoregressive modeling and estimation in the presence of noise," *Digital signal processing*, vol. 4, no. 4, pp. 207–221, 1994.

[9] S. Thomassey and A. Fiordaliso, "A hybrid sales forecasting system based on clustering and decision trees," *Decision Support Systems*, vol. 42, no. 1, pp. 408–421, 2006.

[10] G. Montavon, W. Samek, and K.-R. Müller, "Methods for interpreting and understanding deep neural networks," *Digital Signal Processing*, vol. 73, pp. 1–15, 2018.

[11] J. D. Pintér, "How difficult is nonlinear optimization? a practical solver tuning approach, with illustrative results," *Annals of Operations Research*, vol. 265, no. 1, pp. 119–141, 2018.

[12] X. Ying, "An overview of overfitting and its solutions," in *Journal of Physics: Conference Series*, vol. 1168, p. 022022, IOP Publishing, 2019.

[13] Y. Ren, L. Zhang, and P. N. Suganthan, "Ensemble classification and regression-recent developments, applications and future directions," *IEEE Computational intelligence magazine*, vol. 11, no. 1, pp. 41–53, 2016.

[14] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural computation*, vol. 3, no. 1, pp. 79–87, 1991.

[15] R. T. Clemen, "Combining forecasts: A review and annotated bibliography," *International Journal of Forecasting*, vol. 5, no. 4, pp. 559–583, 1989.

[16] T. G. Dietterich, "Ensemble methods in machine learning," in *Multiple Classifier Systems*, (Berlin, Heidelberg), pp. 1–15, Springer Berlin Heidelberg, 2000.

[17] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The m4 competition: 100,000 time series and 61 forecasting methods," *International Journal of Forecasting*, vol. 36, no. 1, pp. 54–74, 2020.

[18] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "M5 accuracy competition: Results, findings, and conclusions," *International Journal of Forecasting*, vol. 38, no. 4, pp. 1346–1364, 2022.

[19] J. H. Friedman, "Greedy function approximation: A gradient boosting machine.," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189 – 1232, 2001.

[20] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[21] P. Montero-Manso, G. Athanasopoulos, R. J. Hyndman, and T. S. Talagala, "Fforma: Feature-based forecast model averaging," *International Journal of Forecasting*, vol. 36, no. 1, pp. 86–92, 2020.

[22] S. M. Mastelini, F. K. Nakano, C. Vens, A. C. P. de Leon Ferreira, *et al.*, "Online extra trees regressor," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[23] R. M. Cruz, R. Sabourin, G. D. Cavalcanti, and T. Ing Ren, "Meta-des: A dynamic ensemble selection framework using meta-learning," *Pattern Recognition*, vol. 48, no. 5, pp. 1925–1935, 2015.

[24] J. Smith and K. F. Wallis, "A simple explanation of the forecast combination puzzle," *Oxford bulletin of economics and statistics*, vol. 71, no. 3, pp. 331–355, 2009.

[25] Z.-H. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: many could be better than all," *Artificial intelligence*, vol. 137, no. 1-2, pp. 239–263, 2002.

[26] X. Qiu, L. Zhang, Y. Ren, P. N. Suganthan, and G. Amaratunga, "Ensemble deep learning for regression and time series forecasting," in *2014 IEEE symposium on computational intelligence in ensemble learning (CIEL)*, pp. 1–6, IEEE, 2014.

[27] J. Mendes-Moreira, C. Soares, A. M. Jorge, and J. F. D. Sousa, "Ensemble approaches for regression: A survey," *Acm computing surveys (csur)*, vol. 45, no. 1, pp. 1–40, 2012.

[28] H. Tyralis, G. Papacharalampous, and A. Langousis, "Super ensemble learning for daily streamflow forecasting: Large-scale demonstration and comparison with multiple machine learning algorithms," *Neural Computing and Applications*, vol. 33, no. 8, pp. 3053–3068, 2021.

[29] F. Ilhan, O. Karaahmetoglu, I. Balaban, and S. S. Kozat, "Markovian rnn: an adaptive time series prediction network with hmm-based switching for nonstationary environments," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[30] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017.

[31] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[32] O. Akbilgic, H. Bozdogan, and M. E. Balaban, "A novel hybrid rbf neural networks model as a forecaster," *Statistics and Computing*, vol. 24, pp. 365–375, 2014.

[33] G. P. Zhang, "Time series forecasting using a hybrid arima and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, 2003.

[34] H. Xu, R. Ma, L. Yan, and Z. Ma, "Two-stage prediction of machinery fault trend based on deep learning for time series analysis," *Digital Signal Processing*, vol. 117, p. 103150, 2021.

[35] Y. Liu and X. Yao, "Ensemble learning via negative correlation," *Neural networks*, vol. 12, no. 10, pp. 1399–1404, 1999.

[36] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[37] J. Feng, Y. Xu, Y. Jiang, and Z. Zhou, "Soft gradient boosting machine," *CoRR*, vol. abs/2006.04059, 2020.

[38] J. Lee, W. Wang, F. Harrou, and Y. Sun, "Wind power prediction using ensemble learning-based models," *IEEE Access*, vol. 8, pp. 61517–61527, 2020.

[39] S. T. U. Raju, A. Sarker, A. Das, M. M. Islam, M. S. Al-Rakhami, A. M. Al-Amri, T. Mohiuddin, and F. R. Albogamy, "An approach for demand forecasting in steel industries using ensemble learning," *Complexity*, vol. 2022, pp. 1–19, 2022.

[40] R. M. Cruz, R. Sabourin, G. D. Cavalcanti, and T. Ing Ren, "Meta-des: A dynamic ensemble selection framework using meta-learning," *Pattern Recognition*, vol. 48, no. 5, pp. 1925–1935, 2015.

[41] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

[42] P. Derbeko, R. El-Yaniv, and R. Meir, "Variance optimized bagging," in *Machine Learning: ECML 2002* (T. Elomaa, H. Mannila, and H. Toivonen, eds.), (Berlin, Heidelberg), pp. 60–72, Springer Berlin Heidelberg, 2002.

[43] S. Mao, L. Jiao, L. Xiong, S. Gou, B. Chen, and S.-K. Yeung, "Weighted classifier ensemble based on quadratic form," *Pattern Recognition*, vol. 48, no. 5, pp. 1688–1706, 2015.

[44] T. Moon and W. Stirling, *Mathematical Methods and Algorithms for Signal Processing*. Prentice Hall, 2000.

[45] J. Yang, X. Zeng, S. Zhong, and S. Wu, "Effective neural network ensemble approach for improving generalization performance," *IEEE transactions on neural networks and learning systems*, vol. 24, no. 6, pp. 878–887, 2013.

[46] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[47] R. Lin, Z. Zhou, S. You, R. Rao, and C.-C. J. Kuo, "Geometrical interpretation and design of multilayer perceptrons," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[48] G. A. Seber and A. J. Lee, *Linear regression analysis*. John Wiley & Sons, 2012.

[49] D. Dua and C. Graff, "Uci machine learning repository," 2021.

[50] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[51] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[52] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[53] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.

[54] G. P. Zhang, "A neural network ensemble method with jittered training data for time series forecasting," *Information Sciences*, vol. 177, no. 23, pp. 5329–5346, 2007.

[55] D. Kari, A. H. Mirza, F. Khan, H. Ozkan, and S. S. Kozat, "Boosted adaptive filters," *Digital Signal Processing*, vol. 81, pp. 61–78, 2018.

[56] M. Lee, J. Lee, and J.-H. Chang, "Ensemble of jointly trained deep neural network-based acoustic models for reverberant speech recognition," *Digital Signal Processing*, vol. 85, pp. 1–9, 2019.

[57] H. Chen and X. Yao, "Regularized negative correlation learning for neural network ensembles," *IEEE Transactions on Neural Networks*, vol. 20, no. 12, pp. 1962–1979, 2009.

[58] H. Chen and X. Yao, "Multiobjective neural network ensembles based on regularized negative correlation learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 12, pp. 1738–1751, 2010.

[59] P. S. de Mattos Neto, G. D. Cavalcanti, and F. Madeiro, "Nonlinear combination method of forecasters applied to pm time series," *Pattern Recognition Letters*, vol. 95, pp. 65–72, 2017.

[60] Q. Wang, S. Li, R. Li, and F. Jiang, "Underestimated impact of the covid-19 on carbon emission reduction in developing countries–a novel assessment based on scenario analysis," *Environmental Research*, vol. 204, p. 111990, 2022.

[61] Q. Wang, S. Li, and R. Li, "Forecasting energy demand in china and india: Using single-linear, hybrid-linear, and non-linear time series forecast techniques," *Energy*, vol. 161, pp. 821–831, 2018.

[62] Q. Wang, S. Li, and F. Jiang, "Uncovering the impact of the covid-19 pandemic on energy consumption: New insight from difference between pandemic-free scenario and actual electricity consumption in china," *Journal of Cleaner Production*, vol. 313, p. 127897, 2021.

[63] S. d. O. Domingos, J. F. de Oliveira, and P. S. de Mattos Neto, "An intelligent hybridization of arima with machine learning models for time series forecasting," *Knowledge-Based Systems*, vol. 175, pp. 72–86, 2019.

[64] P. S. de Mattos Neto, J. F. de Oliveira, S. d. O. Domingos, H. V. Siqueira, M. H. Marinho, and F. Madeiro, "An adaptive hybrid system using deep learning for wind speed forecasting," *Information Sciences*, vol. 581, pp. 495–514, 2021.

[65] D. M. Izidio, P. S. de Mattos Neto, L. Barbosa, J. F. de Oliveira, M. H. d. N. Marinho, and G. F. Rissi, "Evolutionary hybrid system for energy consumption forecasting for smart meters," *Energies*, vol. 14, no. 7, p. 1794, 2021.

[66] J. F. L. de Oliveira, L. D. S. Pacífico, P. S. G. de Mattos Neto, E. F. S. Barreiros, C. M. de Oliveira Rodrigues, and A. T. de Almeida Filho, "A hybrid optimized error correction system for time series forecasting," *Applied soft computing*, vol. 87, p. 105970, 2020.

[67] Q. Ding, "Long-term load forecast using decision tree method," in *2006 IEEE PES Power Systems Conference and Exposition*, pp. 1541–1543, IEEE, 2006.

[68] W.-Y. Loh, "Classification and regression trees," *Wiley interdisciplinary reviews: data mining and knowledge discovery*, vol. 1, no. 1, pp. 14–23, 2011.

[69] Y. Bengio, O. Delalleau, and C. Simard, "Decision trees do not generalize to new variations," *Computational Intelligence*, vol. 26, no. 4, pp. 449–467, 2010.

[70] P. Zhou, J. Feng, C. Ma, C. Xiong, S. C. H. Hoi, *et al.*, "Towards theoretically understanding why sgd generalizes better than adam in deep learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21285–21296, 2020.

Appendix A

Analysis on Constraints

In this analysis, we examine the optimal weight vectors for ensembling and the associated costs under known statistics for each of the three constraints placed on the ensembling weight vector $\boldsymbol{w}_{\boldsymbol{s}_t^E} \in \mathbb{R}^M$ where M represents the number of base models present in the ensemble. Define $\boldsymbol{C}(\boldsymbol{s}_t)$ as the conditional autocorrelation matrix at time t of the base prediction vector $\hat{\boldsymbol{y}}_t = [\hat{y}_t^{(1)}, \ldots, \hat{y}_t^{(M)}]^T$ given the superset of side-information vectors employed by all the base learners and specific to the ensemble and let $\boldsymbol{a}(\boldsymbol{s}_t)$ represent the conditional cross-correlation vector at time t between the target signal y_t and the base prediction vector considering the super set side-information, i.e.,

$$\begin{aligned} \boldsymbol{C}(\boldsymbol{s}_t) &= \mathbb{E}[\hat{\boldsymbol{y}}_t \hat{\boldsymbol{y}}_t^T \mid \boldsymbol{s}_t^E] \\ \boldsymbol{a}(\boldsymbol{s}_t) &= \mathbb{E}[y_t \hat{\boldsymbol{y}}_t \mid \boldsymbol{s}_t^E]. \end{aligned}$$

Considering the unconstrained case, components of $\boldsymbol{w}_{\boldsymbol{s}_t^E}$ can take any real number; hence, the optimization problem given $\boldsymbol{s}_t^E$ for a particular differentiable loss function ℓ is

$$\min_{\boldsymbol{w}_{\boldsymbol{s}_t^E} \in \mathbb{R}^M} \quad \ell\big(y_t, \sum_{i=1}^{M} w_{\boldsymbol{s}_t^E}^{(i)} \hat{y}_t^{(i)}\big), \tag{A.1}$$

where the feasible region is the entire M dimensional Euclidean space. If the statistics $\boldsymbol{C}(\boldsymbol{s}_t)$ and $\boldsymbol{a}(\boldsymbol{s}_t)$ are known at all times, the optimal unconstrained

ensembling vector at time t under the expected squared error loss, i.e., when $\ell(y_t, \hat{y}_t) = \mathbb{E}[(y_t - \hat{y}_t)^2 | \boldsymbol{s}_t^E] = \mathbb{E}[(y_t - \sum_{i=1}^M w_{\boldsymbol{s}_t^E}^{(i)} \hat{y}_t^{(i)})^2 | \boldsymbol{s}_t^E]$, is the solution to the normal equations and given as

$$\boldsymbol{w}_{\boldsymbol{s}_t^E,\text{unc}}^* = \boldsymbol{C}(\boldsymbol{s}_t)^{-1}\boldsymbol{a}(\boldsymbol{s}_t).$$

Furthermore, assuming the target signal has the conditional variance $\sigma(\boldsymbol{s}_t)^2$ at time t, the conditional mean squared error of the unconstrained ensembling is given by

$$L_{t,\text{unc}}^* := \sigma(\boldsymbol{s}_t)^2 - \boldsymbol{a}(\boldsymbol{s}_t)^T \boldsymbol{C}(\boldsymbol{s}_t)^{-1}\boldsymbol{a}(\boldsymbol{s}_t).$$

In the affine-constrained case, the weight vector components are required to sum up to 1. This translates to the optimization problem given as

$$\begin{aligned} \min_{\boldsymbol{w}_{\boldsymbol{s}_t^E} \in \mathbb{R}^M} \quad & \ell\big(y_t, \sum_{i=1}^M w_{\boldsymbol{s}_t^E}^{(i)} \hat{y}_t^{(i)}\big) \\ \text{subject to} \quad & \boldsymbol{w}_{\boldsymbol{s}_t^E}^T \mathbf{1} = 1, \end{aligned} \tag{A.2}$$

where the feasible region is now an $M-1$ dimensional hyperplane in $\mathbb{R}^M$. In fact, this problem could be cast as an $M-1$-dimensional unconstrained optimization over $\tilde{\boldsymbol{w}}_{\boldsymbol{s}_t} \in \mathbb{R}^{M-1}$ such that the M^{th} component of the weight vector complements the sum of the entire vector to be 1, i.e.,

$$\begin{aligned} \min_{\tilde{\boldsymbol{w}}_{\boldsymbol{s}_t^E} \in \mathbb{R}^{M-1}} \quad & \ell\big(y_t, \sum_{i=1}^{M-1} w_{\boldsymbol{s}_t^E}^{(i)} \hat{y}_t^{(i)}\big) \\ \text{subject to} \quad & w_{\boldsymbol{s}_t^E}^{(M)} = 1 - \tilde{\boldsymbol{w}}_{\boldsymbol{s}_t^E}^T \mathbf{1}. \end{aligned} \tag{A.3}$$

With the transformation of the constrained problem from (A.2), which is a linearly constrained quadratic optimization problem under the squared loss, i.e., $\ell(y_t, \hat{y}_t) = \mathbb{E}[(y_t - \hat{y}_t)^2 | \boldsymbol{s}_t^E]$, to (A.3), which is an unconstrained problem, the optimal affine ensembling weight vector admits a closed-form solution given as

$$\boldsymbol{w}_{\boldsymbol{s}_t^E,\text{aff}}^* = \boldsymbol{C}(\boldsymbol{s}_t)^{-1}\boldsymbol{a}(\boldsymbol{s}_t) - \frac{\mathbf{1}^T \boldsymbol{C}(\boldsymbol{s}_t)^{-1}\boldsymbol{a}(\boldsymbol{s}_t) - 1}{\mathbf{1}^T \boldsymbol{C}(\boldsymbol{s}_t)^{-1}\mathbf{1}} \boldsymbol{C}(\boldsymbol{s}_t)^{-1}\mathbf{1}.$$

We note that the affine combination preserves the unbiasedness of the base predictors. If each base predictor provides unbiased predictions of y_t, then we have

$$\begin{aligned}\mathbb{E}[\boldsymbol{w}_{\boldsymbol{s}_t^E}^T \hat{\boldsymbol{y}}_{\boldsymbol{t}} |\, \boldsymbol{s}_t^E] &= \sum_{i=1}^{M} w_{\boldsymbol{s}_t^E}^{(i)} \mathbb{E}[\hat{y}_t^{(i)}] = \sum_{i=1}^{M} w_{\boldsymbol{s}_t^E}^{(i)} \mathbb{E}[y_t] \\ &= \mathbb{E}[y_t] \sum_{i=1}^{M} w_{\boldsymbol{s}_t^E}^{(i)} = \mathbb{E}[y_t].\end{aligned}$$

Hence, for the expected squared error loss, the optimal affine-constrained weights are given as

$$L_{t,\text{aff}}^* := \sigma(\boldsymbol{s}_t)^2 - \boldsymbol{a}(\boldsymbol{s}_t)^T \boldsymbol{C}(\boldsymbol{s}_t)^{-1} \boldsymbol{a}(\boldsymbol{s}_t) + \frac{(\mathbf{1}^T \boldsymbol{C}(\boldsymbol{s}_t)^{-1} \boldsymbol{a}(\boldsymbol{s}_t) - 1)^2}{\mathbf{1}^T \boldsymbol{C}(\boldsymbol{s}_t)^{-1} \mathbf{1}}.$$

As for the convex constraint case, we not only require that the components of the weight vector sum up to 1 but also they should be nonnegative, i.e., the optimization problem becomes

$$\begin{aligned} \min_{\boldsymbol{w}_{\boldsymbol{s}_t^E} \in \mathbb{R}^M} \quad & \ell\big(y_t, \sum_{i=1}^{M} w_{\boldsymbol{s}_t^E}^{(i)} \hat{y}_t^{(i)}\big) \\ \text{subject to} \quad & \boldsymbol{w}_{\boldsymbol{s}_t^E}^T \mathbf{1} = 1 \;\&\; w_{\boldsymbol{s}_t^E}^{(i)} \geq 0, \quad i = 1, \ldots, M. \end{aligned} \tag{A.4}$$

Under a convex loss ℓ, problem (A.4) is a convex quadratic minimization problem, and the feasible region is the M-dimensional unit simplex. Unlike the previous two constraints, (A.4) does not admit a closed-form solution. We can, however, project (unconstrained) weights to the unit simplex iteratively to find the optimal weight vector. For brevity, we let $M = 2$, which is a common case in practice, and derive the procedure under squared error loss. To this end, we begin by noting that the squared loss at time t as a function of the ensembling weight vector $\boldsymbol{w}_{\boldsymbol{s}_t^E}$ is

$$\begin{aligned} L_t(\boldsymbol{w}_{\boldsymbol{s}_t^E}) &= \sigma(\boldsymbol{s}_t)^2 - \boldsymbol{a}(\boldsymbol{s}_t)^T \boldsymbol{C}(\boldsymbol{s}_t)^{-1} \boldsymbol{a}(\boldsymbol{s}_t) \\ &\quad + (\boldsymbol{w}_{\boldsymbol{s}_t^E} - \boldsymbol{C}(\boldsymbol{s}_t)^{-1} \boldsymbol{a}(\boldsymbol{s}_t))^T \boldsymbol{C}(\boldsymbol{s}_t) (\boldsymbol{w}_{\boldsymbol{s}_t^E} - \boldsymbol{C}(\boldsymbol{s}_t)^{-1} \boldsymbol{a}(\boldsymbol{s}_t)) \\ &= L_{t,unc}^* + (\boldsymbol{w}_{\boldsymbol{s}_t^E} - \boldsymbol{w}_{\boldsymbol{s}_t^E,\text{unc}}^*)^T \boldsymbol{C}(\boldsymbol{s}_t) (\boldsymbol{w}_{\boldsymbol{s}_t^E} - \boldsymbol{w}_{\boldsymbol{s}_t^E,\text{unc}}^*). \end{aligned} \tag{A.5}$$

We can rewrite (A.5) instead of the optimal affine weights $\boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{aff}}$ as

$$\begin{aligned}
L_t(\boldsymbol{w}_{\boldsymbol{s}_t^E}) &= L^*_{t,\text{unc}} + ((\boldsymbol{w}_{\boldsymbol{s}_t^E} - \boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{aff}}) + (\boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{aff}} - \boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{unc}}))^T \boldsymbol{C}(\boldsymbol{s}_t)((\boldsymbol{w}_{\boldsymbol{s}_t^E} - \boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{aff}}) \\
&\quad + (\boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{aff}} - \boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{unc}})) \\
&= L^*_{t,unc} + (\boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{aff}} - \boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{unc}})^T \boldsymbol{C}(\boldsymbol{s}_t)(\boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{aff}} - \boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{unc}}) \\
&\quad + (\boldsymbol{w}_{\boldsymbol{s}_t^E} - \boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{aff}})^T \boldsymbol{C}(\boldsymbol{s}_t)(\boldsymbol{w}_{\boldsymbol{s}_t^E} - \boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{aff}}) \\
&\quad - 2(\boldsymbol{w}_{\boldsymbol{s}_t^E} - \boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{aff}})^T \boldsymbol{C}(\boldsymbol{s}_t)(\boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{aff}} - \boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{unc}}) \\
&= L^*_{t,\text{aff}} + (\boldsymbol{w}_{\boldsymbol{s}_t^E} - \boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{aff}})^T \boldsymbol{C}(\boldsymbol{s}_t)(\boldsymbol{w}_{\boldsymbol{s}_t^E} - \boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{aff}}) \\
&\quad - 2\frac{(\mathbf{1}^T \boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{unc}} - 1)}{\mathbf{1}^T \boldsymbol{C}(\boldsymbol{s}_t)^{-1}\mathbf{1}}(\boldsymbol{w}_{\boldsymbol{s}_t^E} - \boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{aff}})^T \mathbf{1} \\
&= L^*_{t,\text{aff}} + (\boldsymbol{w}_{\boldsymbol{s}_t^E} - \boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{aff}})^T \boldsymbol{C}(\boldsymbol{s}_t)(\boldsymbol{w}_{\boldsymbol{s}_t^E} - \boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{aff}}),
\end{aligned}$$

where the last line follows from $(\boldsymbol{w}_{\boldsymbol{s}_t^E} - \boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{aff}})^T \mathbf{1} = 0$ due to the unit simplex constraint. Further ignoring the constant $L^*_{t,\text{aff}}$, the optimization problem reduces to minimizing the $\boldsymbol{C}(\boldsymbol{s}_t)$-weighted ℓ_2-norm of $\boldsymbol{w}_{\boldsymbol{s}_t^E} - \boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{aff}}$ subject to $\boldsymbol{w}_{\boldsymbol{s}_t^E}$ being on the unit simplex. For the assumed $M = 2$, we let Δ_2 be the unit simplex in $\mathbb{R}^2$. There are 2 cases to consider for this problem:

- $\boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{aff}} \in \Delta_2$: The optimal affine combination is within the unit simplex set and hence a closed-form solution can be written for the optimal weight vector: $\boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{con}} = \boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{aff}}$. Accordingly, $L^*_{t,\text{con}} := L^*_{t,\text{aff}}$.
- $\boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{aff}} \notin \Delta_2$: This case is when combination weights are of opposite signs. For simplicity, we assume the weight that is assigned to the first base predictor is negative and the other one is positive. Then, we can write $\boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{con}} = \boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{aff}} + \alpha[1 \;\; -1]^T$ for some $\alpha \in \mathbb{R}$. Consequently, the cost function to minimize becomes the $\boldsymbol{C}(\boldsymbol{s}_t)$-weighted ℓ_2-norm of $\alpha[-1 \;\; 1]^T$. This implies that the cost is proportional to the magnitude of α. The optimal choice would be the smallest value that keeps $\boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{con}}$ within Δ_2, which is $-w^{(1)}_{\boldsymbol{s}_t^E,\text{aff}}$. In this case, $\boldsymbol{w}^*_{\boldsymbol{s}_t^E,\text{con}}$ is the corner point of Δ_2: $[0 \;\; 1]^T$. This leads to $L^*_{t,\text{con}} := L^*_{t,\text{aff}} + [-1 \;\; 1]\, \boldsymbol{C}(\boldsymbol{s}_t)\, [-1 \;\; 1]^T (w^{(1)}_{\boldsymbol{s}_t^E,\text{aff}})^2$.

Remark. *From the inclusion order of the feasible regions of the constrained problems as well as the derived optimal squared error losses as the special cases, we*

naturally have

$$L^*_{t,unc} \leq L^*_{t,aff} \leq L^*_{t,con} \tag{A.6}$$

for all times t. This inequality holds in theory when the correlation statistics of the target signal and the base models are known at all times. In practice, however, those statistics are rarely known and therefore, there is a learning cost associated with each constraint, which renders the relation in (A.6) useless. Hence, even though the unconstrained case seems to promise the least error, the associated learning procedure might lead to overfitting, i.e., in essence, affine and convex constrained cases perform as "regulators" in that, they offer a trade-off between bias and variance of the ensembling model. In fact, as shown in our simulations in Section 2.5, the unconstrained ensembling scheme does not always achieve the lowest error.

Remark. *Since $\boldsymbol{C}(\boldsymbol{s}_t)$ and $\boldsymbol{a}(\boldsymbol{s}_t)$ are rarely known in practice, we present novel and generic ensemble algorithms to find the optimal ensembling weight vectors under three constraint schemes, in Section 2.4.*

Appendix B

Ensemble Learning Procedure

Here, we provide a detailed view of our ensemble learning procedure as illustrated in Fig. 2.2.

Algorithm 1 Ensemble Learning Procedure

- **Offline Phase:** Training the Base and Ensemble Algorithms
  inputs:
  - $\{y_t\} \triangleq \{y_t\}_{t=1}^{T}$: the training data that consists of T sequential samples from the time series sequence y_t.
  - M base models.
  - $\{\boldsymbol{s}_t^{(i)}\} \triangleq \{\boldsymbol{s}_t^{(i)}\}_{t=1}^{T}$: time series features for the i^{th} base predictor at time T.

  outputs:
  - ensemble learner, which produces the weight vector $w_{\boldsymbol{s}_t^E}$ for combining base predictions.
  - $\{\boldsymbol{s}_t^E\} \triangleq \{\boldsymbol{s}_t^E\}_{t=1}^{T}$: time series features for the ensemble learner.

  prepare ensemble data:

- split $\{y_t\} \triangleq \{y_t\}_{t=1}^{T}$ into training and test periods for the base algorithms, where $1 \leq t \leq T_1$ is the training period and $T_1 + 1 \leq t \leq T$ is the test period.
- train all of the base algorithms over the training period $1 \leq t \leq T_1$.
- generate forecasts for the test period for all of the base models, i.e., form $\{\hat{y}_t^{(i)}\}_{t=T_1+1}^{T},\ 1 \leq i \leq M$.
- construct the ensemble side-information vector $\boldsymbol{s}_t^E$, which is usually the union of the base models' side-information vectors.

train the ensemble learner:

- train the ensemble learner using $\boldsymbol{s}_t^E$ to minimize the loss

$$\arg\min_{\boldsymbol{w}_{\boldsymbol{s}_t^E}^T} \sum_{t=T_1+1}^{T} \ell(y_t, \hat{y}_t^E) \tag{B.1}$$

  where $\hat{y}_t^E = \sum_{i=1}^{M} w_{\boldsymbol{s}_t^E}^{(i)} \hat{y}_t^{(i)}$, ℓ is a given differentiable loss function such as the squared error loss, $\boldsymbol{w}_{\boldsymbol{s}_t^E} = [\, w_{\boldsymbol{s}_t^E}^{(1)}, w_{\boldsymbol{s}_t^E}^{(2)}, \ldots, w_{\boldsymbol{s}_t^E}^{(M)} \,]^T$ is the weight vector for the linear combination of the base learners, and $w_{\boldsymbol{s}_t^E}^{(i)}$ is the combination weight assigned to the i^{th} base learner at time t.

- train all of the base algorithms over the entire training data $1 \leq t \leq T$.

- **Online Phase:** Predicting the Test Data
  inputs:
  - $\{y_t\}_{t=T+1}^{T_2}$: the test data that consists of sequential data samples of the time series sequence y_t, from $T+1$ to T_2.
  - the trained ensemble learner from the offline phase.
  - M base models, trained on the training data.
  - $\{\boldsymbol{s}_t^{(i)}\}_{t=T+1}^{T_2}$: time series features for the i^{th} base predictor at time T.
  - $\{\boldsymbol{s}_t^E\}_{t=T+1}^{T_2}$: time series features for the ensemble learner.

  outputs:

– combination weights $w^{(i)}_{\boldsymbol{s}^E_t}$ for each base learner $1 \leq i \leq M$, for $T+1 \leq t \leq T_2$.

– ensemble predictions $\hat{y}^E_t$ for $T+1 \leq t \leq T_2$.

generate combination weights:

– generate forecasts for the test data from all of the base models, i.e., generate $\{\hat{y}^{(i)}_t\}_{t=T+1}^{T_2},\ 1 \leq i \leq M$.

– construct the ensemble side-information vector $\boldsymbol{s}^E_t$ for the test dataset.

– feed the side-information vector $\boldsymbol{s}^E_t$ to the ensemble learner to generate the weight vector $\boldsymbol{w}^T_{\boldsymbol{s}^E_t} = [\, w^{(1)}_{\boldsymbol{s}^E_t}, w^{(2)}_{\boldsymbol{s}^E_t}, \ldots, w^{(M)}_{\boldsymbol{s}^E_t}\,]^T$, for $T+1 \leq t \leq T_2$.

– linearly combine the weights given to each base learner with corresponding predictions to predict $\hat{y}^E_t = \sum_{i=1}^{M} w^{(i)}_{\boldsymbol{s}^E_t} \hat{y}^{(i)}_t$, for $T+1 \leq t \leq T_2$.

Appendix C

Ensemble Derivations

C.1 LightGBM Ensemble Derivations

C.1.1 Unconstrained Case

For the unconstrained case, there is no restriction on the value of $w^{(i)}_{\boldsymbol{s}^E_t}$. Therefore, we can simply write the transformation relation as $w^{(i)}_{\boldsymbol{s}^E_t} = p^{(i)}_{\boldsymbol{s}^E_t}$. The gradient and hessian equations in (2.3) for the base learner i, at time t become

$$\begin{aligned}
G_i &= 2(y_t - \hat{y}^E_t)(-\frac{\partial \hat{y}^E_t}{\partial p^{(i)}_{\boldsymbol{s}^E_t}}) \\
&= 2(y_t - \hat{y}^E_t)(-\sum_{m=1}^{M} \frac{\partial w^{(m)}_{\boldsymbol{s}^E_t}}{\partial p^{(i)}_{\boldsymbol{s}^E_t}} \hat{y}^{(m)}_t) \\
&= 2\hat{y}^{(i)}_t(\hat{y}^E_t - y_t) \\
H_i &= 2\hat{y}^{(i)}_t(\frac{\partial \hat{y}^E_t}{\partial p^{(i)}_{\boldsymbol{s}^E_t}}) \\
&= 2(\hat{y}^{(i)}_t)^2.
\end{aligned}$$

C.1.2 Affine Constrained Case

For the affine constrained case, the restriction is the summation of the combination weights should be equal to one, i.e., $\sum_{i=1}^{M} w_{\boldsymbol{s}_t^E}^{(i)} = 1$. In order to satisfy this relation, we apply the given normalization operation $w_{\boldsymbol{s}_t^E}^{(i)} = \frac{p_{\boldsymbol{s}_t^E}^{(i)}}{\sum_{m=1}^{M} p_{\boldsymbol{s}_t^E}^{(m)}}$. Hence, the gradient and hessian equations in (2.3) for the base learner i, at time t become

$$
\begin{aligned}
G_i &= 2(y_t - \hat{y}_t^E)(-\frac{\partial \hat{y}_t^E}{\partial p_{\boldsymbol{s}_t^E}^{(i)}}) \\
&= 2(y_t - \hat{y}_t^E)(-\sum_{m=1}^{M} \frac{\partial w_{\boldsymbol{s}_t^E}^{(m)}}{\partial p_{\boldsymbol{s}_t^E}^{(i)}} \hat{y}_t^{(m)}) \\
&= 2(y_t - \hat{y}_t^E)(\frac{1}{\sum_{m=1}^{M} p_{\boldsymbol{s}_t^E}^{(m)}})(\hat{y}_t^E - \hat{y}_t^{(i)}) \\
H_i &= 2(\frac{1}{\sum_{m=1}^{M} p_{\boldsymbol{s}_t^E}^{(m)}})^2(\hat{y}_t^E - \hat{y}_t^{(i)})(3\hat{y}_t^E - 2y_t - \hat{y}_t^{(i)})
\end{aligned}
$$

C.1.3 Convex Constrained Case

For the convex constrained case, the restriction is the summation of the combination weights should be equal to one, such that $\sum_{i=1}^{M} w_{\boldsymbol{s}_t^E}^{(i)} = 1$. In addition, $0 \leq w_{\boldsymbol{s}_t^E}^{(i)} \leq 1$ should also be satisfied. Therefore, we apply the softmax transformation given as $w_{\boldsymbol{s}_t^E}^{(i)} = \frac{e^{p_{\boldsymbol{s}_t^E}^{(i)}}}{\sum_{m=1}^{M} e^{p_{\boldsymbol{s}_t^E}^{(m)}}}$. Therefore, the gradient and hessian equations in (2.3) for the base learner i, at time t become

$$
\begin{aligned}
G_i &= 2(y_t - \hat{y}_t^E)(-\frac{\partial \hat{y}_t^E}{\partial p_{\boldsymbol{s}_t^E}^{(i)}}) \\
&= 2(y_t - \hat{y}_t^E)(-\sum_{m=1}^{M} \frac{\partial w_{\boldsymbol{s}_t^E}^{(m)}}{\partial p_{\boldsymbol{s}_t^E}^{(i)}} \hat{y}_t^{(m)}) \\
&= 2(y_t - \hat{y}_t^E) w_{\boldsymbol{s}_t^E}^{(i)} (\hat{y}_t^E - \hat{y}_t^{(i)}) \\
H_i &= G_i(1 - 2w_{\boldsymbol{s}_t^E}^{(i)}) + 2(w_{\boldsymbol{s}_t^E}^{(i)})^2(\hat{y}_t^E - \hat{y}_t^{(i)})^2.
\end{aligned}
$$

C.2 MLP Ensemble Derivations

C.2.1 Unconstrained Case

For the unconstrained case, there is no restriction on the value of $w^{(i)}_{\boldsymbol{s}^E_t}$. Therefore, we can simply write the transformation relation as $w^{(i)}_{\boldsymbol{s}^E_t} = p^{(i)}_{\boldsymbol{s}^E_t}$. Hence, the backpropagation equations become

$$\frac{\partial L_t}{\partial U^{(1)}_{l,k}} = \sum_{i=1}^{M} 2(\hat{y}^E_t - y_t)\hat{y}^{(i)}_t U^{(2)}_{m,l} f'(v_l)x_k,$$
$$\frac{\partial L_t}{\partial U^{(2)}_{m,l}} = 2(\hat{y}^E_t - y_t)\hat{y}^{(m)}_t z_l,$$

where $f'(v_l)$ is the derivative of ReLU, which is the piece-wise function

$$f'(v_l) = \begin{cases} 0, & \text{if } v_l \leq 0 \\ v_l, & \text{if } v_l > 0. \end{cases}$$

C.2.2 Affine Constrained Case

For the affine constrained case, the restriction is the summation of the combination weights should be equal to one, such that $\sum_{i=1}^{M} w^{(i)}_{\boldsymbol{s}^E_t} = 1$. In order to satisfy this relation, we apply the given normalization operation $w^{(i)}_{\boldsymbol{s}^E_t} = \frac{p^{(i)}_{\boldsymbol{s}^E_t}}{\sum_{m=1}^{M} p^{(m)}_{\boldsymbol{s}^E_t}}$. Hence, the backpropagation equations become

$$\frac{\partial L_t}{\partial U^{(1)}_{l,k}} = \sum_{i=1}^{M} 2c(\hat{y}^E_t - y_t)(\hat{y}^{(m)}_t - \hat{y}^E_t)U^{(2)}_{m,l} f'(v_l)x_k,$$
$$\frac{\partial L_t}{\partial U^{(2)}_{m,l}} = 2c(\hat{y}^E_t - y_t)(\hat{y}^{(m)}_t - \hat{y}^E_t)z_l,$$

where c is the constant term given as $c = (\frac{1}{\sum_{m=1}^{M} p^{(m)}_{\boldsymbol{s}^E_t}})$.

C.2.3 Convex Constrained Case

In the convex constrained case, the restriction is the summation of the combination weights should be equal to one, such that $\sum_{i=1}^{M} w_{\boldsymbol{s}_t^E}^{(i)} = 1$. In addition, $0 \leq w_{\boldsymbol{s}_t^E}^{(i)} \leq 1$ should also be satisfied. Therefore, we apply the given softmax transformation $w_{\boldsymbol{s}_t^E}^{(i)} = \frac{e^{p_{\boldsymbol{s}_t^E}^{(i)}}}{\sum_{m=1}^{M} e^{p_{\boldsymbol{s}_t^E}^{(m)}}}$. Hence, the backpropagation equations become

$$\frac{\partial L_t}{\partial U_{l,k}^{(1)}} = \sum_{i=1}^{M} 2(\hat{y}_t^E - y_t) w_m (\hat{y}_t^{(m)} - \hat{y}_t^E) U_{m,l}^{(2)} f'(v_l) x_k,$$

$$\frac{\partial L_t}{\partial U_{m,l}^{(2)}} = 2(\hat{y}_t^E - y_t) w_m (\hat{y}_t^{(m)} - \hat{y}_t^E) z_l.$$