Digital Signal Processing 132 (2022) 103802

Contents lists available at ScienceDirect

Digital Signal Processing

journal homepage: www.elsevier.com/locate/dsp

Joint optimization of linear and nonlinear models for sequential regression

Arda Fazla *, Mustafa E. Aydin, Suleyman S. Kozat

Department of Electrical and Electronics Engineering, Bilkent University, Ankara, Turkey

ARTICLE INFO

Article history:
Available online 28 October 2022

Keywords:
Seasonal auto-regressive integrated moving average with eXogenous factors (SARIMAX)
Soft gradient boosting decision tree (Soft GBDT)
Regression
Stochastic gradient descent (SGD)
Online learning

ABSTRACT

We investigate nonlinear regression and introduce a novel approach based on the joint optimization of linear and nonlinear models. In order to capture both the nonlinear and linear characteristics in sequential data, we model the underlying data as a combination of linear and nonlinear models, where we optimize the models jointly to minimize the final regression error. As the nonlinear model, we employ a differentiable version of the boosted decision trees. As the linear model, we use the well-known SARIMAX model. Our approach is generic so that any differentiable nonlinear or linear model can be readily employed provided that they are differentiable. By this joint optimization, we alleviate the well-known underfitting and overfitting problems in modeling sequential data. Through our experiments on synthetic and real-life data, we demonstrate significant improvements over individual components as well as the combination/mixture methods in the literature.

1. Introduction

1.1. Background

We study online regression/prediction, where we observe a sequential time series signal with related side-information sequence and predict the next sample. Online regression is extensively studied in the signal processing [1,2] and machine learning [3] literatures since it has a wide range of application areas including transportation [4], education [5] and object tracking [6]. In such regression problems, linear modeling approaches are commonly used because of their low computational complexity and robustness with limited data [7]. However, in most real-life applications, the data display nonlinear characteristics [8], where linear models may be inadequate [9]. In these situations, nonlinear models such as decision trees [10] and neural networks [11] are commonly used. We emphasize that the linear models are a subclass of nonlinear models, nonlinear models can theoretically model both the linear and nonlinear data. However, most nonlinear approaches have well-known shortcomings as they are hard to optimize due to high computational complexity [12] and tend to overfit [13].

As a remedy, ensemble or mixture models are heavily investigated in the signal processing and machine learning literatures [14] as they provide diversity and reliability by combining several different models including linear and nonlinear models [15]. Although ensemble models combine the predictions of different base models, each base model in the combination is independently optimized for the underlying data. Hence, the issues of underfitting and overfitting persist for linear models and nonlinear models, respectively. Another model mixture approach that tries to alleviate these problems is to combine linear and nonlinear models by fitting one of the models to the residuals of the other model [16]. This approach is called direct two-stage modeling and is extensively studied in the signal processing and machine learning literatures [16,17]. By fitting one model to the residuals of the other, the direct approach tries to capture both the underlying linear and nonlinear data components in real-life scenarios [8]. However, direct two-stage models are also insufficient in modeling the real-life sequential data as there is no joint optimization in this approach, i.e., one model is first fitted to the data independent of the other model and then the second model is fitted to the residuals of the first one. Hence, the models are not trained jointly to minimize the final error yielding sub-optimal performance [18].

To remedy these problems, we model the underlying data as an ensemble or combination of linear and nonlinear models, however, unlike previous approaches [14,16] we optimize the parameters jointly to minimize the final regression error. Hence, with joint optimization, we leverage both models. As the linear model, we use the well-known SARIMAX (Seasonal Auto-Regressive Integrated Moving Average with eXogenous factors) [19] and as the nonlinear model, we use the boosted soft decision trees (Soft GBDT) [20]. We

* Corresponding author.
E-mail addresses: arda@ee.bilkent.edu.tr (A. Fazla), enesa@ee.bilkent.edu.tr (M.E. Aydin), kozat@ee.bilkent.edu.tr (S.S. Kozat).

https://doi.org/10.1016/j.dsp.2022.103802
1051-2004/

provide the related gradient calculations for both architectures. We emphasize that our framework is generic so that any differentiable nonlinear or linear can be used instead of SARIMAX or Soft GBDT by changing the corresponding gradient equations, as shown in our paper.

1.2. Prior art

Real-life time series often contain both linear and nonlinear patterns [8], where neither linear models such as SARIMAX [19] nor nonlinear models such as decision trees [10] are adequate. As a remedy, different models are used together to capture different patterns in real-life data [21], the most common approach being ensemble or mixture models. Ensemble models combine the predictions of several models and are heavily investigated in different fields such as signal processing [22,14], speech recognition [23], computational intelligence, statistics, and machine learning [15,17]. There are several methods for designing ensembles; one can train individual models independently from each other and then combine them [14], or train individual models in a sequential manner where each model focuses on correcting the errors made by the previous ones [16].

Ensembling approaches are prone to suffer from overfitting and underfitting issues that are present in the individual base models [18]. To remedy these issues, several approaches are investigated, such as adding independent noise to the training data of each model, using different preprocessing techniques for each model, subsampling the training data of each model, and using an additional model to combine the predictions of the base models [21]. These approaches all introduce sub-optimal solutions, as the base models are still independently optimized to the data; hence, the training is independent of the model combination stage [18]. However, as we show in our simulations, linear and nonlinear models must be optimized jointly as joint optimization benefits all models. Therefore, we introduce an algorithm that optimizes the parameters of a linear and a nonlinear model jointly to minimize the final error.

There have been previous attempts to jointly optimize the base models in ensembling. The negative correlation learning method [18] attempts to train and combine individual models in the same learning process by negatively correlating the error of each model with the rest of the ensemble at each sample. Even though this 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. In addition, the negative correlation learning model [18] also requires a predetermined λ term, which is a parameter used to adjust the strength of the penalty given in the loss function for the correlation between the errors of the individual models, and can take any value between 0 and 1. Therefore, the value of the λ term directly changes the loss of each individual model at each sample, thus affecting the learning procedure of the ensemble algorithm. This causes the ensemble structure to become biased. Note that various approaches to model the λ term have been proposed [24,25], but they do not eliminate the bias in the structure but rather reduce it while increasing the complexity of the optimization. Our approach differs from the joint optimization methods in the literature [18,24,25] as the linear and nonlinear models in our approach learn to produce a single joint prediction for the data samples. Thus, the correlation between the models is not predetermined but learned during joint optimization. Hence, we significantly improve the regression performance as illustrated in our simulations.

1.3. Contributions

Our contributions are as follows:

- We introduce a two-stage model composed of the linear SARIMAX model and the nonlinear Soft GBDT model for sequential time series data regression that can be jointly optimized using any gradient based optimization algorithm.
- Our proposed structure is generic as the linear and nonlinear models in our approach can be readily replaced by any other differentiable regression model.
- With various experiments containing real-life data, we illustrate significant performance improvements with respect to the mixture methods in the literature.

2. Preliminaries

We first present a literature review on the two-stage approaches that focus on combining linear and nonlinear models. We then illustrate our problem description where we explain our notations to be employed for the rest of the paper.

2.1. Literature review

In the literature of time series forecasting, there have been various approaches to combine linear and nonlinear models. These approaches rely on the assumptions that real-life time series data often contain both linear and nonlinear patterns [26]. Therefore, linear models are not solely adequate in modeling complex real-life data, as they can not capture the nonlinear patterns in the data. On the other hand, even though nonlinear models can theoretically model both the linear and nonlinear data component, nonlinear models such as decision trees and artificial neural networks are hard to optimize due to high computational complexity [12] and tend to overfit [13].

One of the former well-known model mixture/ensemble approaches is the two-stage method of fitting one of the models on the residuals, the error series, of another model. The final forecasts are then obtained by summing the forecasts of the two models [16,27–29]. The assumption in this approach is, the real-life time series data can be modeled as the aggregation of its distinct linear and nonlinear components in some manner [26]. This approach has inspired many researchers to employ different models to forecast real-life time series in various fields. G. Peter Zhang [16] proposed fitting a neural network on the residuals of a linear ARIMA (Auto-Regressive Integrated Moving Average) model, and showed significant performance improvements compared to the base models on different datasets. Qiang Wang et al. [27] proposed using the linear ARIMA (Auto-Regressive Integrated Moving Average) and the nonlinear BPNN (Back Propagation Neural Network) to form two separate ARIMA-BPNN and BPNN-ARIMA models, to simulate the carbon emissions of India, China, U.S and EU. However, by independently fitting the models to the data and then the residuals of the former model, it is hard to ensure that the linear and nonlinear patterns present in the data are captured solely by the linear and nonlinear models, respectively [26]. In addition, the relation between the forecasts of the former model in the approach, and its residuals, or even the relation between the forecasts of two models is unknown [26]. Hence, the two models do not fully benefit from each other.

As a remedy, several researchers have proposed an ensembling addition to this mixture approach [26,30–33]. After the latter model is fit on the residuals of the former model, an ensemble algorithm is introduced to the scheme, which combines the forecasts of the two models in a linear or nonlinear fashion. Paulo S.G. de Mattos Neto et al. [26] proposed combining the forecasts of two

different models using an MLP (Multi-Layer Perceptron) for time series forecasting related to public health. Domingos S. de O. Santos Júnior et al. [30] proposed two different models, one which an SVR (Support Vector Regression) is employed to combine the forecasts of an ARIMA-MLP model, where the MLP is fit on the residuals of the ARIMA, and another one in which an MLP is employed to combine the forecasts of an ARIMA-SVR model, where the SVR model is fit on the residuals of the ARIMA model. Paulo S.G. de Mattos Neto et al. [31] proposed fitting an LSTM (Long Short-Term Memory) on the residuals of several linear statistical models, and then combining the two model forecasts using SVR for wind speed forecasting. Diogo M. F. Izidio et al. [32] used a similar method for energy consumption forecasting, where a nonlinear model of choice is fit on the residuals of a seasonal ARIMA model, and a separate nonlinear model is used for combination of the forecasts of the two models. João Fausto Lorenzato de Oliveira et al. [33] investigated different techniques to combine the forecasts of different linear and nonlinear models using PSO (Particle Swarm Optimization). However, even though this approach has been widely used by various researchers in different fields, the two base models are still independently optimized to the data, hence, the training is independent of the model combination stage [18]. Therefore, the addition of an ensemble algorithm to model the relation between the forecasts of the former model and the latter residual model, is naturally sub-optimal. As we show in our simulations, linear and nonlinear models must be optimized jointly during training, as joint optimization benefits all models.

Another approach combining linear and nonlinear models, that employs joint optimization among the models, is the negative correlation learning [18], 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 [24,25], 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 that the real-life time series data can be modeled as the aggregation of its complementing linear and nonlinear components [26].

2.2. Problem description

All the vectors presented in this paper are in bold lowercase letters. Matrices are denoted by uppercase bold letters. For instance, $\boldsymbol{x}$ represents a vector while $\boldsymbol{X}$ represents a matrix. x_k denotes the k^{th} element of the vector $\boldsymbol{x}$, and $x_{t,k}$ represents the k^{th} element of the vector $\boldsymbol{x}_t$ at time instance t. $X_{i,j}$ represents the entry at the i^{th} row and j^{th} column of the matrix $\boldsymbol{X}$. Finally, $\boldsymbol{x}^T$ and $\boldsymbol{X}^T$ represent the transpose of the vector $\boldsymbol{x}$ and the matrix $\boldsymbol{X}$, respectively.

In this paper, we study the nonlinear prediction of sequential time series data. We observe the sequence $\{y_t\}_{t\geq 1}$, along with the 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, based on $\{y_1, ..., y_t\}$ and $\{\boldsymbol{s}_1, ..., \boldsymbol{s}_t\}$, we predict y_{t+1} in a purely online manner

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

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

$$L = \frac{1}{T}\sum_{t=1}^{T} l(y_t, \hat{y}_t), \tag{2}$$

for any T, where l is a given differential loss function, where we employ the squared error loss.

We next present our proposed approach in Section 3.

3. Proposed approach

We first propose our joint optimization approach in Section 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 3.2 and Section 3.3, respectively. Finally, training for the joint optimization of the two models is given in Section 3.4.

3.1. Joint optimization method

Fig. 1 illustrates the online training architecture of our proposed approach. At any sample time t, the linear and nonlinear models in our approach predict $\hat{y}_{t+1}^{\{n\}}$ and $\hat{y}_{t+1}^{\{m\}}$, respectively, while employing the past samples of the observed data $\{y_t\}_{t\geq 1}$ and the past side information vectors $\{\boldsymbol{s}_t\}_{t\geq 1}$. Hence, our joint model prediction is given by $\hat{y}_{t+1} = \hat{y}_{t+1}^{\{n\}} + \hat{y}_{t+1}^{\{m\}}$. After the predicted sequence is observed for the sample time $t+1$, i.e., y_{t+1} is made available, we optimize the linear and nonlinear models, where the loss to be minimized is given as $l(y_{t+1}, \hat{y}_{t+1}) = l(y_{t+1}, \hat{y}_{t+1}^{\{m\}} + \hat{y}_{t+1}^{\{n\}})$. We then update/optimize the learnable parameters of each model by the related gradients through the calculated differentiable loss function using a gradient based optimization.

We emphasize that the optimization of the linear and nonlinear models is based on residual fitting. As we employ the squared error loss function as l (which can be directly interchanged with error metrics such as root mean squared error (RMSE) or mean absolute error (MAE)), 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 $l(y_{t+1}, \hat{y}_{t+1}^{\{m\}} + \hat{y}_{t+1}^{\{n\}}) = l(y_{t+1} - \hat{y}_{t+1}^{\{m\}}, \hat{y}_{t+1}^{\{n\}})$, since the linear and nonlinear model predictions are produced solely by the linear and nonlinear model parameters, respectively. Thus, the prediction of the linear (nonlinear) model acts as a constant for the gradient calculations of the nonlinear (linear) model, which we illustrate during the gradient calculations in Section 3.4.

Hence, at each sample time t, our model makes the joint prediction $\hat{y}_{t+1}$, and once the y_{t+1} is observed, we optimize our linear and nonlinear models by their gradients through a differentiable loss function of choice, where the individual model predictions directly effect the optimization process of each other, hence performing joint optimization. Once the optimization to the single sample loss is done, we move on to predicting the next sample, thus, our predictions are produced in a purely online manner.

Therefore, we model f in (1) as the process of producing a joint prediction by the linear and nonlinear models, where both models

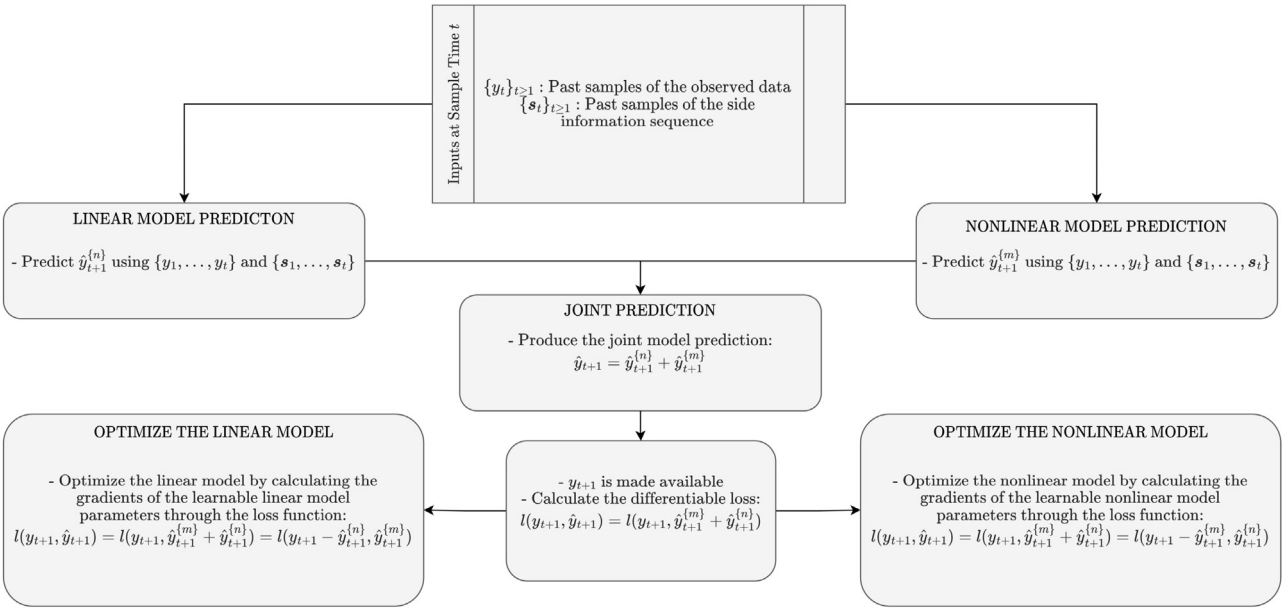

Fig. 1. Online training of our proposed approach based on the joint optimization of linear and nonlinear models.

are optimized together to minimize the final loss in an online manner, updating f at each time sample/iteration accordingly. Hence, the corresponding relation is given as:

$$\hat{y}_{t+1} = f\big(\{\ldots, y_{t-1}, y_t\}, \{\ldots, \boldsymbol{s}_{t-1}, \boldsymbol{s}_t\}\big) = \hat{y}_{t+1}^{\{n\}} + \hat{y}_{t+1}^{\{m\}}, \tag{3}$$

where f indicates the proposed joint optimization approach explained in Fig. 1, $\hat{y}_{t+1}^{\{m\}}$ is the forecast of the linear model and $\hat{y}_{t+1}^{\{n\}}$ is the forecast of the nonlinear model, for the time sample $t+1$.

Our proposed approach is based on the widely-known residual fitting methods in the literature. Many researchers have proposed different methods as explained in Section 2.1, with the main aspect being residual fitting. Our method distinguishes from the state-of-the-art residual fitting methods in the literature as our model learns the co-relation between the model predictions directly from joint-optimization. For example, the method of fitting one of the models on the residuals, the error series, of another model that is directly fit to the observed sequence [16,27–29], does not adequately provide the learning process between the relation between the forecasts of the two models. In addition, even though the latter model is fit on the residuals of the former model, the former model does not benefit from the learning process of the latter model. On the other hand, the introduction of an ensemble algorithm to the scheme, which models the relation between the residual fit model and the former model [26,30–33] also has certain drawbacks. Even though there is a model that learns the relation between the two model predictions, in our proposition, we show that the learning between the two models can be directly achieved with joint optimization. Thus, the introduction of an ensemble algorithm is not necessary that also increases the computational complexity, as we demonstrate in our simulations. Even though there are a few approaches in the literature that involve some level of interaction between the models during the learning procedure, such as the negative correlation learning [18], none of these approaches satisfy unbiased interaction between the models during learning, which is satisfied by our joint optimization approach. However, the disadvantage of using our proposed approach, compared to the similar studies is that the base models in our approach must be fully differentiable. Hence, model structures that are not differentiable, such as the hard decision trees [34], can not be directly employed.

We next introduce the models employed in our approach and then provide the corresponding gradient equations for the proposed joint optimization.

3.2. Linear model

As the linear model we employ the widely used model SARIMAX (Seasonal Auto-Regressive Integrated Moving Average with eXogenous factors) developed by Box and Jenkins [19]. The SARIMAX model is a statistical approach and its main purpose is to forecast future values of the given sequential time series data by using linear relations of the previously observed values of the sequential data, as well as the side-information and the error terms. The SARIMAX is often expressed by the orders of its parameters, such as SARIMAX $(p,d,q)(P,D,Q)s$, and is given as

$$\phi(B)\Phi(B^s)\Delta^d\Delta_s^D y_t = c + \theta(B)\Theta(B^s)\epsilon_t + \sum_{i=1}^{m}\beta_i s_{t,i}, \tag{4}$$

where c is constant, ϵ_t is the white noise process that can be evaluated as the error sequence, B is the back-shift operator defined as $B^i y_t = y_{t-i}$ and s is the seasonality parameter of the model. For example, $s=7$ implies weekly seasonality in daily data, while $s=365$ implies yearly seasonality. Δ term indicates the differencing applied to the sequential data to diminish the effect of the trend to make the data stationary, where $\Delta^d = (1-B)^d$ is the non-seasonal 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 $\theta(B)$ are the non-seasonal auto-regressive and moving average parts of the model in the orders of p and q, respectively, whereas $\Phi(B^s)$ and $\Theta(B^s)$ are the seasonal auto-regressive and moving average parts of the model in the orders of P and Q, respectively. The $\phi(B)$, $\theta(B)$, $\Phi(B^s)$ and $\Theta(B^s)$ indicate the lag operations given as

$$\begin{aligned}
\phi(B) &= 1 - \phi_1 B - \cdots - \phi_p B^p \\
\theta(B) &= 1 - \theta_1 B - \cdots - \theta_q B^q \\
\Phi(B^s) &= 1 - \Phi_1 B^s - \cdots - \Phi_P B^{sP} \\
\Theta(B^s) &= 1 - \Theta_1 B^s - \cdots - \Theta_Q B^{sQ}.
\end{aligned} \tag{5}$$

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

Seasonal AR (P=1, s=7); Nonseasonal AR (p=2); Nonseasonal Difference (d=1); Observed Sequence; Seasonal MA (Q=1, s=7); Nonseasonal MA (p=1); Error Sequence; Constant; eXogenous Term

The summation term $\sum_{i=1}^{m}\beta_i s_{t,i}$ on the right side of (4) is the eXogenous term, which models the relationship between the observed sequence y_t and the side information vector $\boldsymbol{s}_t$. To clarify the model structure, an example of the SARIMAX $(2,1,1)(1,0,1)$ [7] model can be expressed 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 side-information coefficients β_i are learnt during our joint optimization approach explained in Section 3.1. The orders of these parameters indicated as $(p,d,q)(P,D,Q)s$ are the hyperparameters of the model to be selected before optimizing the linear model and play a critical role in modeling the observed data accurately.

Remark. Although we use SARIMAX as our linear model, other linear models can also be directly used. We would like to emphasize that the Box & Jenkins models family [19] guarantees the proper linear modeling of the observed data sequence, as the error series of the models (difference between the model forecasts and the predicted sequence) do not present relevant linear patterns with respect to the predicted sequence. Therefore, the models from the Box & Jenkins family can be directly interchanged with the SARIMAX model in our approach. Note that, this is not a premise guaranteed by other models such as decision trees [10] or neural networks [11]. One only needs to change the gradient calculations in (10) and (11) to incorporate different linear models.

3.3. Nonlinear model

As the nonlinear model, we use regression decision trees [35]. Commonly, hard decision trees are employed in the literature [34], which work by strictly partitioning the input space using hard decision boundaries [36]. The input vector $\boldsymbol{x}_t$, which in our case is given as $[\ldots, y_{t-1}, y_t, \ldots, s_{t-1}, s_t]^T$, enters the tree from the root node, and is then subjected to a decision rule, rooting it either to the left or the right child, called the internal nodes. This decision-making process is repeated at each internal node, finally reaching a leaf node. A leaf node is a node where no further nodes are connected to itself. At the leaf node, a scalar output value is produced, which in time series regression is the prediction of the model, through which the input space is partitioned.

Hard decision trees have well-known shortcomings as they are prone to overfitting [13] and the decision rule applied at each node is strict, in the sense that only one of the leaf nodes contributes to the output. In addition, the hard decisions made by the hard decision tree model cause it to inherently lack differentiability. However, in order to jointly optimize the outputs of the linear and nonlinear models in our joint optimization approach with respect to Fig. 1, we need to have a fully differentiable and learnable model structure. Therefore, as the nonlinear model in our approach, we employ the *soft* version of the decision trees, which is introduced in [20].

Soft decision trees differ from the hard decision trees as there is no strict rooting at the nodes, but certain rooting probabilities for each child node are generated, resulting in all of the leaves contributing to the final output. At each internal node $n \in N$ in the tree, we apply a decision process that produces a Bernoulli random variable p_n. The value of p_n corresponds to the "probability" of rooting the input to the left child node, given by

$$p_n = \sigma(\boldsymbol{w_n}^T \boldsymbol{x}_t + b_n), \tag{6}$$

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 given by $\sigma(x) = 1/(1+e^{-x})$. Similarly, the probability of rooting the input to the right child node is given by $1-p_n$. The sigmoid function maps the outputs of the internal nodes to the range of $[0,1]$ so that the output values produced by each internal node can be expressed as probabilities.

Fig. 2 illustrates the prediction mechanism of the soft decision tree, e.g., we have 4 leaf nodes and 3 internal nodes. N_1, N_2, N_3 represent the first, second and third internal nodes of the tree, respectively. All other nodes are the leaf nodes. The input vector passes through all the internal nodes and produces probabilities. For each leaf node, there exists a "path probability" formed by taking the product of all the probabilities produced by the internal nodes leading to the leaf node, starting from the root node. Therefore, unlike hard decision trees, we do not end up with an output value produced by a single leaf, but all the leaves contribute to the output on the scale of their path probabilities. Hence, the final output of a single decision tree is given by

$$\begin{aligned} o(\boldsymbol{x}_t) &= \sum_{\ell=1}^{\mathcal{L}} \mathrm{P}_\ell \boldsymbol{\phi}_\ell \\ &= \sum_{\ell=1}^{\mathcal{L}} \Big(\prod_{n \in \mathrm{N}(\ell)} p_i \Big) \boldsymbol{\phi}_\ell \\ &= \sum_{\ell=1}^{\mathcal{L}} \Big(\prod_{n \in \mathrm{N}(\ell)} \sigma(\boldsymbol{w}_n^T \boldsymbol{x}_t + b_n) \Big) \boldsymbol{\phi}_\ell, \end{aligned} \tag{7}$$

where $\mathcal{L}$ represents all the leaves of the tree, P_ℓ is the path probability of the ℓ^{th} leaf node, the nodes leading to the ℓ^{th} leaf node are given by $\mathrm{N}(\ell)$ and ϕ_ℓ is the scalar output value produced at the ℓ^{th} leaf node. The probabilities at each internal node $n \in N$ in the tree, given by p_n, allow us to form a fully 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. Therefore, as illustrated in Fig. 2, all internal nodes have learnable parameters $\boldsymbol{w_n}$ and b_n, and produce probabilities given by p_n. As shown by the bold arrows, all leaves contribute to the final output of the soft decision tree by producing the corresponding leaf output ϕ_l, for any given input vector $\boldsymbol{x}_t$. Note that ϕ_l are also learnable.

Many of these soft decision trees are used together in order to form soft gradient boosting decision trees (Soft GBDT). Each consecutive tree fits to the residuals of the previous tree and produces output values with respect to (7). Hence, the final output value produced by the Soft GBDT is given as

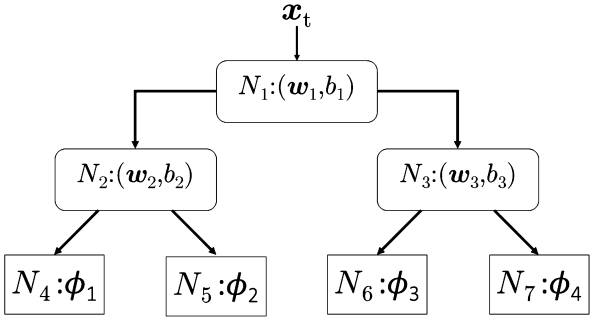

Fig. 2. Soft decision tree.

$$\hat{y}_{t+1}^{\{n\}} = \sum_{k=1}^{K} \nu o^{(k)}(\boldsymbol{x}_t) + c, \tag{8}$$

where K is 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 prediction, produced by the Soft GBDT, $\boldsymbol{x}_t$ is the input vector, ν is the regularization parameter that weighs the predictions of each tree in order to prevent overfitting and c is a constant value independent of the input vector that can be selected as the mean of the sequential time series data y_t, until the current prediction time $t+1$. Note that each soft decision tree $k \in K$ has its own learnable parameters $\boldsymbol{w}_n^{(k)}$, $b_n^{(k)}$ and $\phi_\ell^{(k)}$.

Therefore, we have constructed a fully differentiable gradient boosting tree, with learnable parameters $\boldsymbol{w}_n^{(k)}$ and $b_n^{(k)}$ for each internal node $n \in N$ and ϕ_ℓ for each leaf node $\ell \in \mathcal{L}$ for each tree $k \in K$. Note that, the depth of the soft decision trees, the regularization parameter ν and number of trees in the Soft GBDT are the hyperparameters of the model to be selected before the optimization stage, and play a critical role in modeling the observed data accurately.

Remark. The training of a soft decision tree differs from the hard decision tree as there are learnable parameters (weight vectors) in each node. These learnable parameters are initially set randomly regarding a normal distribution, and are then optimized at each iteration/sample of the online learning stage, in order for the overall Soft GBDT to produce more accurate predictions of the observed sequence.

Remark. Although we use Soft GBDT as our nonlinear model, other nonlinear models such as Artificial Neural Networks (ANN), Polynomial Auto-Regressive (PAR) model etc. can be directly used. One only needs to change the gradient calculations in (12), (13) and (14) to incorporate different nonlinear models.

3.4. Model training

Hence, we have differentiable structures for both the linear and nonlinear model to be employed in our proposed joint optimization approach. For joint optimization, we choose the stochastic gradient descent algorithm since it is fast, generalizes better than other gradient based optimization algorithms [37] and is suitable for online learning where a single sample is used for optimization at each iteration. However, different optimization algorithms that use the gradient information such as Adam, AdaGrad etc. can also be directly employed. Note that, both of the models are optimized to the data sample-by-sample, as we conduct the training based on online learning, as explained in Section 3.1 and illustrated in Fig. 1. We now provide the related gradient calculations for both of 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 to be minimized by our joint model at time t is given as

$$l(y_t, \hat{y}_t), \tag{9}$$

where $\hat{y}_t = \hat{y}_t^{\{m\}} + \hat{y}_t^{\{n\}}$. Here, $\hat{y}_t^{\{m\}}$ and $\hat{y}_t^{\{n\}}$ are the linear and nonlinear predictions of the joint model, respectively. In the case of the squared error loss, which is the error metric we employ in our simulations, the loss equation (9) takes the form

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

Since our model has two components, a linear model and a nonlinear one, while jointly optimizing both models, the parameters of both models are updated based on the joint prediction error made by the two models. The structure for the linear model in our approach is given as (4), where the parameters to be estimated are

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

where ϕ_i are the coefficients of the autoregressive terms in the order p, Φ_i are the coefficients of the seasonal autoregressive terms in the order P, θ_i are the coefficients of the moving average terms in the order q, Θ_i are the coefficients of the seasonal moving average terms in the order Q, s is the seasonality and m is the length of the eXogenous terms. The sizes of these parameters depend on the given time series data, and are the hyperparameters of the model, which should be adjusted in order to accurately predict the data. Here, $\boldsymbol{w}$ is the weight vector, which contains all the learnable parameters of the model to be updated. The gradients for the parameters in $\boldsymbol{w}$ are given as

$$\begin{aligned} \frac{\partial l(y_t, \hat{y}_t)}{\partial w_j} &= \frac{\partial l(y_t, \hat{y}_t)}{\partial \hat{y}_t^{\{m\}}} \frac{\partial \hat{y}_t^{\{m\}}}{\partial w_j} \\ &= (\hat{y}_t - y_t) \frac{\partial \hat{y}_t^{\{m\}}}{\partial w_j} \\ &= (\hat{y}_t^{\{n\}} + \hat{y}_t^{\{m\}} - y_t) \frac{\partial \hat{y}_t^{\{m\}}}{\partial w_j}, \end{aligned} \tag{10}$$

where w_j is the j_{th} term in $\boldsymbol{w}$ to be estimated. The $\frac{\partial \hat{y}_t^{\{m\}}}{\partial w_j}$ term in the equation is simply the term that w_j is multiplied with in (4). For example, in the case of no differencing nor seasonal differencing in the equations, the gradients for $\boldsymbol{w}$ are given as

$$\frac{\partial \hat{y}_t^{\{m\}}}{\partial \boldsymbol{w}} = \big[y_{t-1}, \ldots, y_{t-sP}, \epsilon_{t-1}, \ldots, \epsilon_{t-sQ}, s_{t,1}, \ldots, s_{t,m}\big]. \tag{11}$$

The output of the nonlinear model in our approach is given as (8) where the equation for the output of each soft decision tree is given by (7). Differing from the linear model in our approach, the nonlinear model consists of many individual weak-learners, contributing to an ensemble result. Therefore, we need to consider the loss with respect to every soft decision tree in the model. This loss is given by

$$E_{total} = \frac{1}{2} \sum_{k=1}^{K} (r^{(k)} - \nu o^{(k)})^2, \tag{12}$$

where $o^{(k)}$ is the output of k^{th} tree, ν is the regularization parameter and $r^{(k)}$ is the residual coming to the k^{th} tree, which is formed

by adding the residual terms from the previous trees up to the k^{th} tree. $r^{(k)}$ can be expressed for any k^{th} tree 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 (12). The learnable parameters for the k^{th} soft decision tree are the leaf node outputs $\phi_\ell^{(k)}$ for each leaf node $\ell \in \mathcal{L}$ where $\mathcal{L}$ is the set of leaves of the tree, the weight term $\boldsymbol{w_n}^{(k)}$ and the bias term $b_n{}^{(k)}$ 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 given by

$$\frac{\partial E_{total}}{\partial \phi_\ell^{(k)}} = \mathrm{P}_\ell^{(k)} \sum_{i=k}^{K} (\nu o^{(i)} - r^{(i)}) \tag{13}$$

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

$$\frac{\partial E_{total}}{\partial b_n^{(k)}} = \Bigg(\sum_{\ell^{(k)} \in \mathrm{D}(n^{(k)})} \phi_\ell^{(k)} \frac{\mathrm{P}_\ell^{(k)} \mathrm{p}_n^{(k)} (1 - \mathrm{p}_n^{(k)})}{\mathrm{p}_n^{(k)} - \rho(\ell^{(k)})} \sum_{i=k}^{K} (\nu o^{(i)} - r^{(i)}) \Bigg), \tag{14}$$

where K is the total number of soft decision trees in the model, $\mathrm{D}(n^{(k)})$ is the set of all the leaf nodes that pass down from the internal node $n^{(k)}$, $\boldsymbol{x}_t$ is the common input to 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 $\mathrm{LD}(n^{(k)})$ is the set of leaf nodes that pass down from the left branch of the internal node n, and $\mathrm{RD}(n^{(k)})$ is the set of leaf nodes that pass down from the right branch of the internal node n. Therefore, we have $\mathrm{LD}(n^{(k)}) + \mathrm{RD}(n^{(k)}) = \mathrm{D}(n^{(k)})$.

Finally, the updates for the parameters of both models are given as

$$\boldsymbol{w} = \boldsymbol{w} - \mu_m \frac{\partial l(y_t, \hat{y}_t)}{\partial \boldsymbol{w}}$$

$$\phi_\ell^{(k)} = \phi_\ell^{(k)} - \mu_n \frac{\partial E_{total}}{\partial \phi_\ell^{(k)}}$$

$$\boldsymbol{w}_n^{(k)} = \boldsymbol{w}_n^{(k)} - \mu_n \frac{\partial E_{total}}{\partial \boldsymbol{w}_n^{(k)}}$$

$$b_n^{(k)} = b_n^{(k)} - \mu_n \frac{\partial E_{total}}{\partial b_n^{(k)}},$$

where μ_m and μ_n are the hyperparameters of the stochastic gradient descent algorithm for the linear and nonlinear models that determine how quickly the models are adapted to the optimization problem. Hence, we now have the gradient equations for both the linear and nonlinear models in our joint optimization approach.

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 $\boldsymbol{w_n}^{(k)}$ 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 of a simple residual fitting operation, as both the hyperparameters of linear and nonlinear models should be set in both cases.

The following section illustrates the simulations of our jointly optimized model.

Table 1
The illustrations of the models used in our simulations.

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 3.1 and Fig. 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.

4. Simulations

In this section, we demonstrate the performance of our joint model under different scenarios with several examples, while comparing with other mixture approaches in literature focusing on combining linear and nonlinear models, based on residual fitting and/or ensemble learning. We use five different models, which are explained in Table 1. Note that, for each of the mixture methods, we employ the SARIMAX and the Soft GBDT models from our proposed approach for fairness. For the rest of the simulations, we refer to these models with their names given in Table 1 in order to reduce repetition.

For our simulations, first, we generate synthetic data with strong linear and nonlinear components and verify the necessity of using linear and nonlinear models in a combined manner, rather than individually. In the second part, we illustrate the performance of our model in real-life scenarios, using the daily data of the total residential natural gas demand in Turkey between the years 2018-2020, and the M5 Forecasting Dataset [38] by Walmart. For

Table 2
The hyperparameter search pace for the models used in our simulations.

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}

all the scenarios, we only consider one-step-ahead forecasting and compare different models in terms of the total loss, where the loss function l is demonstrated for different error metrics given as Mean Squared Error (MSE), Mean Absolute Percentage Error (MAPE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE). We also illustrate the cumulative normalized total error with respect to time for all the models used in the experiments, in order to illustrate the online learning process of the model, given as

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

where y_k is the sample of the signal to be predicted and $\hat{y}_k$ is the prediction by the corresponding model.

During the training of the models in our simulations, we first search for the best possible hyperparameters while excluding the test dataset, which consists of the last 20% of the data. We select the best set of hyperparameters for each of the models used in the simulations, which minimizes the total loss in terms of the squared error loss on the first 80% of the data. (Note that the predictions are calculated according to Fig. 1, therefore, we eliminate any data leakage problem.) After the hyperparameters are calculated we re-train the models with the best set of hyperparameters found, on the entire dataset with online learning as explained in Section 3.1 and Fig. 1. We then illustrate and compare the performance of our models using multiple error metrics. The search space for the hyperparameters of the models to be used in our simulations is given Table 2.

Note that, for the models from the literature (reviewed in Section 2.1) that use an additional ensemble model to combine base model forecasts (literature_model_3, literature_model_4, literature_model_5, literature_model_6), the ensembling operation is not trained with online learning for fairness, as the ensemble models that we employ (MLP and Linear Regressor) are not trained with online learning in the given studies. Therefore, after the online learning of the base models, we train the ensemble models on the first 80% of the data and illustrate their performances on the last 20% of the data.

4.1. Synthetic data

The synthetic data consists of manually generated linear and nonlinear data components. Formulation for the linear data is given by

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

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

$$y_t^{\{n\}} = n_1 y_{t-1}^{\{n\}} + n_2 y_{t-2}^{\{n\}} + v_t^{\{n\}} \text{ , if } y_{t-1}^{\{n\}} > 0 \text{ and } y_{t-2}^{\{n\}} > 0$$
$$y_t^{\{n\}} = n_2 (y_{t-1}^{\{n\}})^2 - n_2 y_{t-2}^{\{n\}} + v_t^{\{n\}} \text{ , if } y_{t-1}^{\{n\}} > 0 \text{ and } y_{t-2}^{\{n\}} < 0$$

Table 3
The performance comparison for the models used in simulating the synthetic datasets, under various error metrics.

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**

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

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. Finally, we generate our synthetic data by

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

where the data is of length 1000 and the accuracy of the models is evaluated on the last 200 samples of the data. We generate 10 random instances of y_t (note that instances in case of divergence of the data have been excluded), and illustrate the performance of the models under various error metrics over 10 experiments.

Using the synthetic dataset, we fit the linear, nonlinear and joint models on the synthetically generated data, which contains both the linear and nonlinear data components. Fig. 3 illustrates the performance of the models in terms of the cumulative normalized squared error with respect to time, on the test data. We also illustrate the performance comparison of our joint approach and the based models under different error metrics in Table 3, for an average of 10 randomly generated synthetic datasets. We observe that our joint model outperforms all the other models, hence our joint optimization approach is able to learn the data components better than the base models. Hence, joint optimization is beneficial to both models.

We now illustrate the performance of our joint optimization approach under real-life datasets, while also comparing with state-of-the-art model mixture methods in the literature.

4.2. Total residential natural gas demand in Turkey

The data used in this section consists of the total residential natural gas demand in Turkey for 1000 days during the years 2018-2020. Each day is considered as a single sample, and the data is in natural logarithmic scale. The data shows both linear and nonlinear characteristics, therefore, it is a good candidate for our model experiments. We evaluate the model accuracy on the test set, consisting of the last 200 samples of the data.

We fit the linear and nonlinear base models, our jointly optimized model, and the mixture methods from the literature to the data. In Fig. 4, the comparison of the cumulative error for the models are given, where our jointly optimized model outperforms all other approaches. The error of the Soft GBDT is not provided on Fig. 4 since it had significantly worse error than other methods. Note that, due to the high error of the Soft GBDT, all of the model mixture that combine the two base methods perform worse than the linear SARIMAX model. However, our joint optimization approach does not have this issue and outperforms both of the base models and other mixture models, in terms of multiple error metrics as illustrated in Table 4.

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

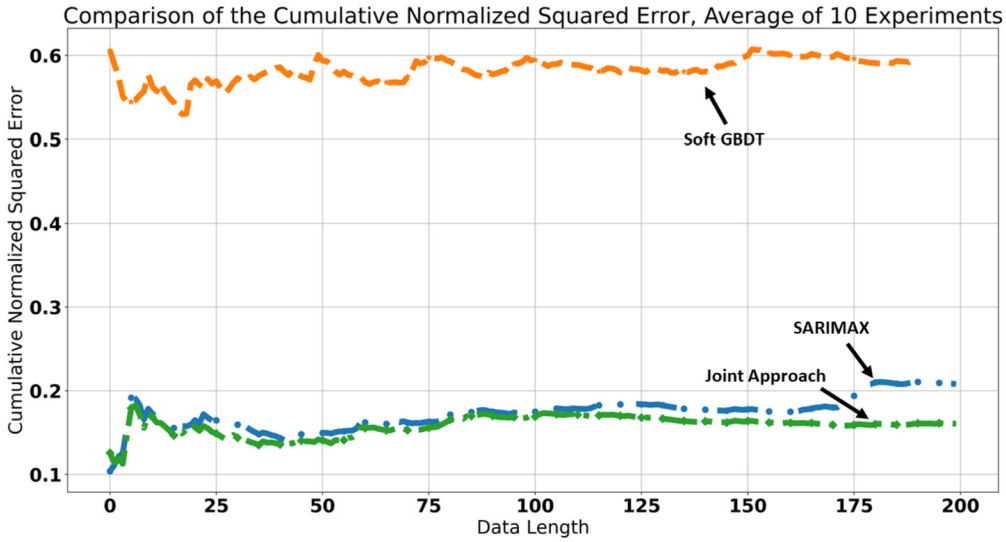

Fig. 3. Comparison of the cumulative normalized squared error for the synthetic data of our jointly optimized model, the base models.

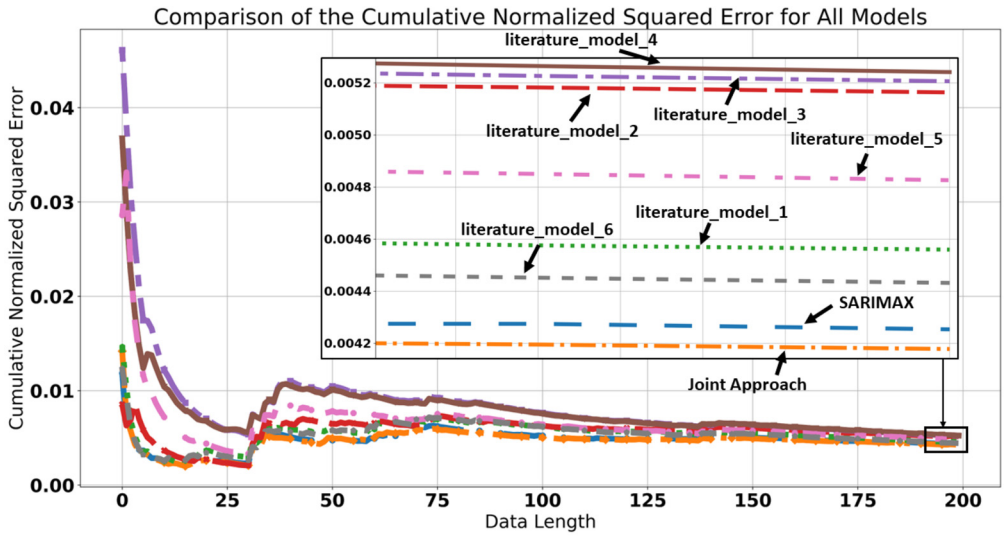

Fig. 4. Comparison of the cumulative error in the natural logarithmic scale for the total residential natural gas demand in Turkey. The performance of our jointly optimized model, the base models and the mixture methods are shown. Note that the error plot for the Soft GBDT model is not provided here, as the error produced by the nonlinear model is high compared to other methods. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

Table 4
The comparison of model performances used in simulating the total residential natural gas demand in Turkey.

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

Forecasting Centre (MOFC), widely cited in the prediction literature [38]. 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 in 3 different categories and 7 different departments. The products are sold in 10 different Walmart stores at 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 evaluated as a single sample. As illustrated in Fig. 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.

We fit the linear and nonlinear base models, our jointly optimized model, and the mixture methods to the data. The comparison of the cumulative error for these models is given in Fig. 6 and their errors on the test set under different error metrics are illustrated in Table 5. Note that, the comparisons for the error metric MAPE are not given as the data contains zero values. Our jointly optimized model significantly outperforms all of the models based on residual fitting from the literature, where only the MLP ensemble combination approach literature_model_6 outperforms our model.

5. Conclusions

We studied the nonlinear regression problem in an online manner and introduced a novel joint optimization approach consisting

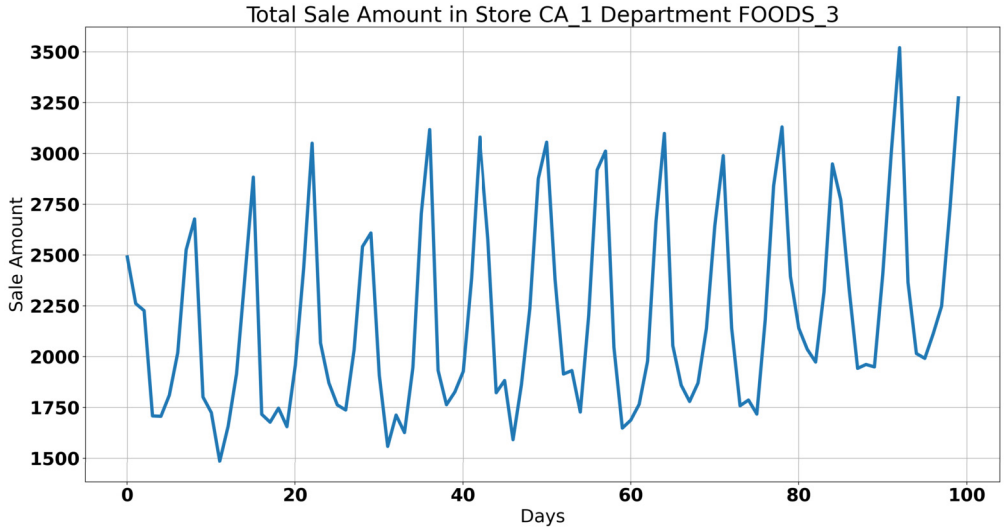

Fig. 5. Total number of sales for the test set in store CA_1 department FOODS_3 of Walmart, USA.

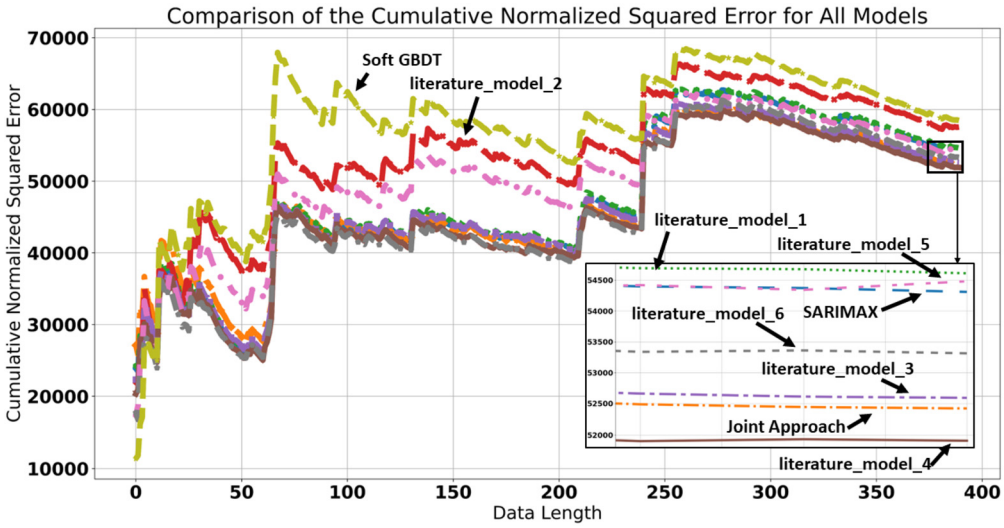

Fig. 6. Comparison of the cumulative error in the natural logarithmic scale for the M5 data. The performance of our jointly approach, the base models and the mixture methods from the literature are shown. The error plots start from the 10th sample for a clear visualization.

Table 5
The comparison of model performances used in simulating the total unit sales in store CA_1 department FOODS_3 of Walmart, USA.

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

of linear and nonlinear models. In order to capture both the linear and nonlinear properties of real-life sequential data, we model the underlying data as an ensemble/combination of linear and nonlinear models. However, unlike the previous state-of-the-art approaches, we jointly optimized the parameters of both models in order to minimize the final regression error. Thus, we leverage both models while avoiding the well-known overfitting and underfitting issues associated with nonlinear and linear models, respectively. We use the widely known SARIMAX for the linear model and employ the soft gradient boosted decision trees (Soft GBDT) for the nonlinear model. We then use the stochastic gradient descent algorithm to jointly optimize the two models and also provide the related gradient calculations. Our model is generic so that any differentiable linear and/or nonlinear model can be used instead of SARIMAX and/or Soft GBDT, or any gradient optimization algorithm can be used instead of stochastic gradient descent. We first verify the necessity of our joint optimization approach, and then showcase performance improvements with respect to both the individual base models and the model mixture methods in the literature over the well-known real life datasets.

CRediT authorship contribution statement

Arda Fazla: Conceptualization, Methodology, Software, Writing – original draft. **Mustafa E. Aydin:** Conceptualization, Software, Writing – original draft. **Suleyman S. Kozat:** Conceptualization, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The code will be made available on GitHub.

References

[1] A.H. Mirza, M. Kerpicci, S.S. Kozat, Efficient online learning with improved LSTM neural networks, Digit. Signal Process. 102 (2020) 102742.
[2] M. Ntemi, C. Kotropoulos, Online hotel rating prediction through a dynamic weighted ordered probit model, Digit. Signal Process. 120 (2022) 103310.
[3] D.F. Specht, et al., A general regression neural network, IEEE Trans. Neural Netw. 2 (6) (1991) 568–576.
[4] J. Van Lint, Online learning solutions for freeway travel time prediction, IEEE Trans. Intell. Transp. Syst. 9 (1) (2008) 38–47.
[5] A.A. Mubarak, H. Cao, W. Zhang, Prediction of students' early dropout based on their interaction logs in online learning environment, Interact. Learn. Environ. (2020) 1–20.
[6] A. Hadjaz, G. Marceau, P. Savéant, M. Schoenauer, Online learning for ground trajectory prediction, arXiv preprint, arXiv:1212.3998, 2012.
[7] D.C. Montgomery, E.A. Peck, G.G. Vining, Introduction to Linear Regression Analysis, John Wiley & Sons, 2021.
[8] E.E. Kuruoğlu, Nonlinear least lp-norm filters for nonlinear autoregressive α-stable processes, Digit. Signal Process. 12 (1) (2002) 119–142.
[9] A.C. Singer, G.W. Wornell, A.V. Oppenheim, Nonlinear autoregressive modeling and estimation in the presence of noise, Digit. Signal Process. 4 (4) (1994) 207–221.
[10] S. Thomassey, A. Fiordaliso, A hybrid sales forecasting system based on clustering and decision trees, Decis. Support Syst. 42 (1) (2006) 408–421.
[11] G. Montavon, W. Samek, K.-R. Müller, Methods for interpreting and understanding deep neural networks, Digit. Signal Process. 73 (2018) 1–15.
[12] J.D. Pintér, How difficult is nonlinear optimization? A practical solver tuning approach, with illustrative results, Ann. Oper. Res. 265 (1) (2018) 119–141.
[13] X. Ying, An overview of overfitting and its solutions, J. Phys. Conf. Ser. 1168 (2019) 022022.
[14] Y. Ren, L. Zhang, P.N. Suganthan, Ensemble classification and regression-recent developments, applications and future directions, IEEE Comput. Intell. Mag. 11 (1) (2016) 41–53.
[15] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, G.E. Hinton, Adaptive mixtures of local experts, Neural Comput. 3 (1) (1991) 79–87.
[16] G.P. Zhang, Time series forecasting using a hybrid ARIMA and neural network model, Neurocomputing 50 (2003) 159–175.
[17] H. Xu, R. Ma, L. Yan, Z. Ma, Two-stage prediction of machinery fault trend based on deep learning for time series analysis, Digit. Signal Process. 117 (2021) 103150.
[18] Y. Liu, X. Yao, Ensemble learning via negative correlation, Neural Netw. 12 (10) (1999) 1399–1404.
[19] G.E. Box, G.M. Jenkins, G.C. Reinsel, G.M. Ljung, Time Series Analysis: Forecasting and Control, John Wiley & Sons, 2015.
[20] J. Feng, Y. Xu, Y. Jiang, Z. Zhou, Soft gradient boosting machine, CoRR, arXiv: 2006.04059 [abs], 2020, arXiv:2006.04059.
[21] G.P. Zhang, A neural network ensemble method with jittered training data for time series forecasting, Inf. Sci. 177 (23) (2007) 5329–5346.
[22] D. Kari, A.H. Mirza, F. Khan, H. Ozkan, S.S. Kozat, Boosted adaptive filters, Digit. Signal Process. 81 (2018) 61–78.
[23] M. Lee, J. Lee, J.-H. Chang, Ensemble of jointly trained deep neural network-based acoustic models for reverberant speech recognition, Digit. Signal Process. 85 (2019) 1–9.
[24] H. Chen, X. Yao, Regularized negative correlation learning for neural network ensembles, IEEE Trans. Neural Netw. 20 (12) (2009) 1962–1979.
[25] H. Chen, X. Yao, Multiobjective neural network ensembles based on regularized negative correlation learning, IEEE Trans. Knowl. Data Eng. 22 (12) (2010) 1738–1751.
[26] P.S. de Mattos Neto, G.D. Cavalcanti, F. Madeiro, Nonlinear combination method of forecasters applied to pm time series, Pattern Recognit. Lett. 95 (2017) 65–72.
[27] Q. Wang, S. Li, R. Li, F. Jiang, Underestimated impact of the covid-19 on carbon emission reduction in developing countries–a novel assessment based on scenario analysis, Environ. Res. 204 (2022) 111990.
[28] Q. Wang, S. Li, R. Li, Forecasting energy demand in China and India: using single-linear, hybrid-linear, and non-linear time series forecast techniques, Energy 161 (2018) 821–831.
[29] Q. Wang, S. Li, 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, J. Clean. Prod. 313 (2021) 127897.
[30] S.d.O. Domingos, J.F. de Oliveira, P.S. de Mattos Neto, An intelligent hybridization of ARIMA with machine learning models for time series forecasting, Knowl.-Based Syst. 175 (2019) 72–86.
[31] P.S. de Mattos Neto, J.F. de Oliveira, S.d.O. Domingos, H.V. Siqueira, M.H. Marinho, F. Madeiro, An adaptive hybrid system using deep learning for wind speed forecasting, Inf. Sci. 581 (2021) 495–514.
[32] D.M. Izidio, P.S. de Mattos Neto, L. Barbosa, J.F. de Oliveira, M.H.d.N. Marinho, G.F. Rissi, Evolutionary hybrid system for energy consumption forecasting for smart meters, Energies 14 (7) (2021) 1794.
[33] J.F.L. de Oliveira, L.D.S. Pacífico, P.S.G. de Mattos Neto, E.F.S. Barreiros, C.M. de Oliveira Rodrigues, A.T. de Almeida Filho, A hybrid optimized error correction system for time series forecasting, Appl. Soft Comput. 87 (2020) 105970.
[34] Q. Ding, Long-term load forecast using decision tree method, in: 2006 IEEE PES Power Systems Conference and Exposition, IEEE, 2006, pp. 1541–1543.
[35] W.-Y. Loh, Classification and regression trees, Wiley Interdiscip. Rev. Data Min. Knowl. Discov. 1 (1) (2011) 14–23.
[36] Y. Bengio, O. Delalleau, C. Simard, Decision trees do not generalize to new variations, Comput. Intell. 26 (4) (2010) 449–467.
[37] 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, Adv. Neural Inf. Process. Syst. 33 (2020) 21285–21296.
[38] S. Makridakis, E. Spiliotis, V. Assimakopoulos, M5 accuracy competition: results, findings, and conclusions, Int. J. Forecast. (2022).

Arda Fazla received the B.S. degree in Electrical and Electronics Engineering from Middle East Technical University, Ankara, Turkey in 2021. He is currently pursuing the M.S. degree in the Department of Electrical and Electronics Engineering at Bilkent University. His research interests include machine learning, time series prediction, signal processing and online learning.

Mustafa Enes Aydin received the B.S. degree in Electrical and Electronics Engineering from Bilkent University, Ankara, Turkey in 2020. He is currently pursuing the M.S. degree in the Department of Electrical and Electronics Engineering at Bilkent University. His research interests include statistical signal processing, time series prediction and machine learning.

Suleyman Serdar Kozat received the B.S. (Hons.) degree from Bilkent University, Ankara, Turkey, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Illinois at Urbana-Champaign, Urbana, IL, USA. He is currently a Professor at the Electrical and Electronics Engineering Department at Bilkent University. He has co-authored over 120 papers in refereed high impact journals and conference proceedings and holds several patent inventions. His current research interests include cyber security, anomaly detection, big data, data intelligence, adaptive filtering and machine learning algorithms for signal processing.