Markovian RNN: An Adaptive Time Series Prediction Network With HMM-Based Switching for Nonstationary Environments

Fatih Ilhan, Oguzhan Karaahmetoglu, Ismail Balaban, and Suleyman Serdar Kozat, *Senior Member, IEEE*

***Abstract*— We investigate nonlinear regression for nonstationary sequential data. In most real-life applications such as business domains including finance, retail, energy, and economy, time series data exhibit nonstationarity due to the temporally varying dynamics of the underlying system. We introduce a novel recurrent neural network (RNN) architecture, which adaptively switches between internal regimes in a Markovian way to model the nonstationary nature of the given data. Our model, Markovian RNN employs a hidden Markov model (HMM) for regime transitions, where each regime controls hidden state transitions of the recurrent cell independently. We jointly optimize the whole network in an end-to-end fashion. We demonstrate the significant performance gains compared to conventional methods such as Markov Switching ARIMA, RNN variants and recent statistical and deep learning-based methods through an extensive set of experiments with synthetic and real-life datasets. We also interpret the inferred parameters and regime belief values to analyze the underlying dynamics of the given sequences.**

***Index Terms*— Hidden Markov models (HMMs), nonlinear regression, nonstationarity, recurrent neural networks (RNNs), regime switching, time series prediction.**

I. Introduction

A. Preliminaries

WE STUDY nonlinear time series prediction with recurrent neural networks (RNNs) in nonstationary environments. In particular, we receive a sequential data and predict the next samples of the given sequence based on the knowledge about history, which includes the previous values of target variables and side information (exogenous variables). Time series prediction task is extensively studied for various applications in the machine learning [1], [2], ensemble learning [3], signal processing [4], and online learning theory [5] literature works. In most real-life scenarios such as finance and business applications, time series data may not be an output of a single stochastic process since the environment can possess nonstationary behavior. In particular, the dynamics of the underlying system, which generates the given sequence can exhibit temporally varying statistics. Moreover, the behavior of the system can even be chaotic or adversarial [6], [7]. Therefore, successfully modeling the nonstationarity of the data carries importance while performing prediction.

Although linear models have been popular, partly since they have been integrated into most statistics and econometrics software packages, neural network-based methods are becoming widely preferred for the time series prediction task thanks to their ability to approximate highly nonlinear and complex functions [8]. In particular, deep neural networks (DNNs) with multiple layers have been successful in resolving overfitting and generalization-related issues. Although their success in some fields such as computer vision has been demonstrated in numerous studies [9], [10], this multilayered structure is not suitable for sequential tasks since it cannot capture the temporal relations in time series data properly [11]. To this end, RNNs are used in sequential tasks thanks to their ability to exploit temporal behavior. RNNs contain a temporal memory called hidden state to store the past information, which helps them to model time series more successfully in several different sequential tasks [12]–[14]. Hence, we consider nonlinear regression with RNN-based networks to perform time series prediction.

To address the difficulties raised by nonstationarity, several methods mostly based on the various ideas of experts [15] are proposed due to their ability to represent nonstationary or piecewise sequential data. A group of experts models rely on the principle of divide and conquer, which aims to find the solution by partitioning the problem into smaller parts. These models usually consist of three main components: separate regressors called experts, a gate that separates the input space into regions, and a probabilistic model that combines the experts [3]. The fact that a group of experts simplifies complex problems by allowing each expert to focus on specific parts of the problem with soft partitioning provides an important advantage while modeling nonstationary sequences [3]. However, these methods require training multiple experts separately, which disallows joint optimization and end-to-end training. In addition, these methods rely on enough diversity among experts such that each expert makes errors at different regions in the feature space. Otherwise, their performance

Manuscript received 17 June 2020; revised 26 January 2021 and 1 June 2021; accepted 17 July 2021. Date of publication 9 August 2021; date of current version 6 February 2023. This work was supported in part by Tubitak under Project 117E153. *(Corresponding author: Fatih Ilhan.)*

Fatih Ilhan, Oguzhan Karaahmetoglu, and Suleyman Serdar Kozat are with the Department of Electrical and Electronics Engineering, Bilkent University, 06800 Ankara, Turkey, and also with DataBoss A. S., ODTU Teknokent, 06800 Ankara, Turkey (e-mail: filhan@ee.bilkent.edu.tr; koguzhan@ee.bilkent.edu.tr; kozat@ee.bilkent.edu.tr).

Ismail Balaban is with DataBoss A. S., ODTU Teknokent, 06800 Ankara, Turkey (e-mail: ismail.balaban@data-boss.com.tr).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3100528.

Digital Object Identifier 10.1109/TNNLS.2021.3100528

compared to single expert models becomes negligibly better or even worse, if none of the experts can fit the data well enough [16].

In certain studies, simple linear regressors such as autoregressive integrated moving average (ARIMA) models are considered as experts, where each expert specializes in a very small part of the problem [15]. To perform gating operation between experts, several adaptive switching approaches such as Markovian switching and transition diagrams are widely preferred [17]–[19]. Markov switching models and their variants were first introduced for sequential modeling tasks in the econometrics literature to handle nonstationarity [20]–[23] and also applied in other fields such as control systems in nonstationary environments [24]–[26]. Initial methods employing Markovian switching time series prediction use multiple linear regressors (or classifiers), where each regressor is responsible for characterizing the behavior of the time series in a different regime. The switching mechanism between these regimes is controlled by a hidden Markov model (HMM). Markov switching model can capture more complex dynamic patterns and nonstationarity, especially when the assumption of the existence of different regimes with Markovian transitions hold. This model and its variants are applied in analyzing and forecasting business, economic, and financial time series [18], [27]. For instance, these models are used to identify business cycles, which consist of several regimes such as expansion and recession states [21], [27]. However, none of these methods consider nonlinear regression with RNNs, which limits their capability to capture complex temporal patterns. Our model, Markovian RNN can be interpreted as a temporally adaptive mixture of experts model, where the regime-specific hidden state transition weights inside the RNN cell are employed as experts and HMM-based Markovian switching performs the gating operation between regimes. In this way, Markovian RNN can detect different regimes and focus on each of them separately through learning separate weights, which enables our model to adapt nonstationarity while making predictions.

Although there exists a significant amount of prior work on the time series prediction task in nonstationary environments, we combine the benefits of RNNs and HMM-based switching for nonlinear regression of nonstationary sequential data. In this study, we introduce a novel time series prediction network, Markovian RNN, which combines the advantages of RNNs and Markov switching. Our model employs an RNN with multiple hidden state transition weights, where each weight corresponds to a different regime. We control the transitions between these regimes with a HMM, which models the regimes as part of a Markov process. In this way, our model can capture the complex sequential patterns thanks to RNNs, and handle nonstationary with Markovian switching. We also optimize the whole network jointly at single stage. Our model can also be extended using different RNN structures such as long short-term memory (LSTM) [28] and gated residual unit (GRU) [29] networks as remarked in Section III-B. Furthermore, we demonstrate the performance gains of Markovian RNN in the time series prediction task with respect to the conventional forecasting methods such as ARIMA with Markov switching [20], RNN variants [28]–[30], and recent statistical [31] and deep learning-based methods [32]. We perform extensive experiments over both synthetic and real datasets. We also investigate the inferred regime beliefs and transitions, as well as analyzing forecasting error.

B. Prior Art and Comparisons

A considerable amount of research has been conducted in machine learning, signal processing, and econometrics literature works to perform time series prediction [2], [32], [33]. Although there are widely embraced linear methods such as autoregression (AR), moving average (MA), ARIMA and their variants in the conventional time series prediction framework, these methods fail to capture complex temporal patterns, since they cannot fully capture nonlinearity [33]. There exists a wide range of nonlinear approaches to perform regression in the machine learning and signal processing literatures. However, these earlier methods suffer from practical disadvantages such as memory and can perform badly due to stability and overfitting issues [34].

To overcome these limitations, neural network-based methods have been increasingly popular thanks to the developments in optimization and neural network literatures [10]. Most of the recent studies adopt RNNs and its variants such as GRU and LSTM networks for sequential tasks. Certain studies have successfully applied RNNs and their variants for sequential tasks [29], [35]. In this study, we employ RNNs considering their power to capture complex nonlinear temporal patterns and generalization capability over unseen data. In addition to RNNs, certain recent studies employ statistical methods based on decomposition [31] or hybrid models that combine statistical methods such as AR and exponential smoothing with RNN-based deep architectures [36], [37]. In another study, Oreshkin *et al.* [32] use fully-connected layer stacks with residual connections. However, these methods are not designed to perform under nonstationary environments in which the underlying dynamics of the sequence might evolve through time. To increase generalization, authors employ certain ensembling procedures. In [32], a three-level ensembling procedure is applied through training multiple models on different metrics, with various temporal scales and different random initializations, and considering the median of predictions as final. In Smyl [37], train models with different random initializations, number of epochs and data subsets and then perform rank-based selection over validation set.

In order to improve generalization capability and handle nonstationarity for time series prediction, several studies adopt a set of expert-based approaches instead of straightforward ensembling procedures used in [32] and [37]. A group of ARIMA experts is considered in [15] to obtain a universal approximator for prediction in stationary sequences. In the work, the authors interpret the mixture of experts as a form of neural network. Various studies have developed universal sequential decision algorithms by dividing sequences into independent segments and performing switching between linear regressors [15], [19]. However, these works utilize linear regressors as experts, which may perform poorly in challenging scenarios. Another study also employs nonlinear regressors as experts for stock price prediction task [38]. However, the nonlinear regressors employed in these studies have multilayered perceptron architectures without any temporally recurrent structure. Instead, we employ RNNs to handle the temporal patterns in time series data. In addition, we jointly optimize the whole network at a single stage whereas a group of expert models requires separate training sessions for each expert.

Designing the gating model in a group of experts approach is as crucial as choosing the expert models. For instance, Kozat and Singer employed randomized switching mechanism based on transition diagrams in [19]. In another study, authors use a gating procedure based on a fuzzy inference system in [38]. However, most studies, especially in the business domain and finance literature, prefer Markovian switching-based approaches since they express financial cycles more accurately [21], [23]. Certain earlier variants of this approach such as Hamilton model [20], Kim–Nelson–Startz (KNS) model [21], and Filardo model [22] have been specifically designed and preferred for the tasks in business and finance applications [27]. In addition, these approaches have been integrated into popular analysis, forecasting, and software packages [39], [40]. However, these statistical methods are not flexible in terms of modeling, since they employ linear models with certain assumptions such as sequences with varying mean and variance. Similar approaches have been applied for anomaly detection tasks as well. For instance, Cao *et al.* [41] developed an adaptive HMM with an anomaly state to detect price manipulations. Although Markovian switching-based methods are commonly used for sequential tasks in nonstationary environments, few of them consider nonlinear models, which are mostly simple multilayer networks. In addition, they usually require multiple training sessions and cannot be optimized jointly. However, we introduce a jointly optimizable framework, which can utilize the benefits of nonlinear modeling capability of RNNs and adaptive Markovian switching with HMMs in an effective way.

C. Contributions

Our main contributions are as follows:

1) We introduce a novel time series prediction model, Markovian RNN, based on RNNs and regime switching controlled by HMM. This approach enables us to combine the modeling capabilities of RNNs and adaptivity obtained by HMM-based switching to handle nonstationarity.
2) We use gradient descent-based optimization to jointly learn the parameters of the proposed model. The proposed sequential learning algorithm for Markovian RNN enables us to train the whole network end-to-end at the single stage with a clean pipeline.
3) Our model can prevent oscillations caused by frequent regime transitions by detecting and ignoring outlier data. The sensitivity of the introduced model can readily be tuned to detect regime switches depending on the requirements of desired applications, or with cross-validation.
4) Through an extensive set of experiments with synthetic and real-life datasets, we investigate the capability of Markovian RNN to handle nonstationary sequences with temporally varying statistics.
5) We compare the prediction performance of the introduced model with conventional switching methods, RNN variants, recent decomposition-based statistical methods and deep learning-based models in terms of root mean squared error (RMSE), mean absolute error (MAE).
6) We analyze the effect of a number of regimes and illustrate the inferred regime beliefs and switches to interpret our model.

D. Organization

The organization of the article is as follows. We define the time series prediction task and describe the framework that uses RNNs and HMMs in Section II. Then we provide the introduced model, switching mechanism, and sequential learning algorithm for Markovian RNN in Section III. In Section IV, we demonstrate the performance improvements of the introduced model over an extensive set of experiments with synthetic and real datasets and investigate the inferred regimes and transitions. Finally, we conclude the article in Section VI with several remarks.

II. Problem Description

In this article, all vectors are column vectors and denoted by boldface lower case letters. Matrices are denoted by boldface upper case letters. $\boldsymbol{x}^T$ and $\mathbf{X}^T$ are the corresponding transposes of $\boldsymbol{x}$ and $\mathbf{X}$. $\|\boldsymbol{x}\|$ is the ℓ^2-norm of $\boldsymbol{x}$. $\odot$ and $\oslash$ denotes the Hadamard product and division, i.e., element-wise multiplication and division, operations, respectively. $|\mathbf{X}|$ is the determinant of $\mathbf{X}$. For any vector $\boldsymbol{x}$, x_i is the ith element of the vector. x_{ij} is the element that belongs to $\mathbf{X}$ at the ith row and the jth column. sum$(\cdot)$ is the operation that sums the elements of a given vector or matrix. δ_{ij} is the Kronecker delta, which is equal to one if $i = j$ and zero otherwise. E-notation is used to express very large or small values such that mEn $= m \times 10^n$.

We study nonlinear prediction of nonstationary time series. We observe a vector sequence $\boldsymbol{x}_{1:T} \triangleq \{\boldsymbol{x}_t\}_{t=1}^T$, where T is the length of the given sequence, and $\boldsymbol{x}_t \in \mathbb{R}^{n_x}$ is the input vector for the tth time step. Input vector can contain target variables (endogenous variables) as well as side information (exogenous variables). The target output signal corresponding to $\boldsymbol{x}_{1:T}$ is given by $\boldsymbol{y}_{1:T} = \{\boldsymbol{y}_t\}_{t=1}^T$, where $\boldsymbol{y}_t \in \mathbb{R}^{n_y}$ is the desired output vector at the tth time step. Our goal is to estimate $\boldsymbol{y}_t$ using the inputs until the tth time step by

$$\hat{\boldsymbol{y}}_t = f(\boldsymbol{x}_{1:t}; \boldsymbol{\theta})$$

where f is a nonlinear function parameterized with $\boldsymbol{\theta}$. After observing the target value $\boldsymbol{y}_t$, we suffer the loss $\ell(\boldsymbol{y}_t, \hat{\boldsymbol{y}}_t)$, and optimize the network with respect to this loss. We evaluate the performance of the network by the mean squared error (MSE) obtained over the sequence with

$$L_{\text{MSE}} = \frac{1}{T}\sum_{t=1}^{T} \ell_{\text{MSE}}(\boldsymbol{y}_t, \hat{\boldsymbol{y}}_t) \tag{1}$$

where

$$\ell_{\mathrm{MSE}}(\boldsymbol{y}_t, \hat{\boldsymbol{y}}_t) = \boldsymbol{e}_t^T \boldsymbol{e}_t$$

and $\boldsymbol{e}_t \triangleq \boldsymbol{y}_t - \hat{\boldsymbol{y}}_t$ is the error vector at the tth time step. Other extensions are also possible such as MAE as remarked in Section III-B.

A. Recurrent Neural Networks

We particularly study time series prediction with RNNs. For this task, we use the following form:

$$\begin{aligned} \boldsymbol{h}_t &= f_h(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t; \boldsymbol{\theta}_h) \\ &= f_h(\mathbf{W}_{hh}\boldsymbol{h}_{t-1} + \mathbf{W}_{xh}\boldsymbol{x}_t) \quad (2) \\ \hat{\boldsymbol{y}}_t &= f_y(\boldsymbol{h}_t; \boldsymbol{\theta}_y) \\ &= f_y(\mathbf{W}_{hy}\boldsymbol{h}_t) \quad (3) \end{aligned}$$

where $f_h(x) = \sigma_{\tanh}(x)$ is the element-wise tanh function such that $\sigma_{\tanh}(x) = ((e^x - e^{-x})/(e^x + e^{-x}))$, and $f_y(x) = x$. Here, $\boldsymbol{h}_t \in \mathbb{R}^{n_h}$ is the hidden state vector at time step t. $\mathbf{W}_{hh} \in \mathbb{R}^{n_h \times n_h}$, $\mathbf{W}_{xh} \in \mathbb{R}^{n_x \times n_h}$ and $\mathbf{W}_{hy} \in \mathbb{R}^{n_h \times n_y}$ are the weight matrices. We use $\boldsymbol{\theta}_h = \{\mathbf{W}_{hh}, \mathbf{W}_{xh}\}$ and $\boldsymbol{\theta}_y = \{\mathbf{W}_{hy}\}$ to denote the state transition and state-to-observation parameters, respectively.

We note that the introduced framework can be applied to any RNN structure. Hence, it can be extended to various RNN-based networks such as Gated Recurrent Units (GRU) [29] and LSTM Units [28]. We provide the equations for possible extensions in Remark 1 in Section III-A. Here, we consider RNNs due to their simplicity and practicality in real-life applications. We also do not state bias terms explicitly since they can be augmented into input vector such that $\boldsymbol{x}_t \leftarrow [\boldsymbol{x}_t;\ 1]$.

B. Hidden Markov Models

We utilize HMMs to model the switching mechanism of RNNs, as will be described in Section III. HMM is a statistical model, which consists of a discrete-time discrete-state Markov chain with unobservable hidden states $k_t \in \{1, \ldots, K\}$, where K is the number of states. The joint distribution has the following form:

$$p(k_{1:T}, \boldsymbol{y}_{1:T}) = \prod_{t=1}^{T} p(k_t | k_{t-1}; \Psi) p(\boldsymbol{y}_t | k_t; \boldsymbol{\theta}) \quad (4)$$

where $p(k_1|k_0) = \pi(k_1)$ is the initial state distribution. $p(k_t|k_{t-1}; \Psi)$ is the transmission model defined by a transmission matrix $\Psi \in \mathbb{R}^{K \times K}$ such that $\Psi_{ij} \triangleq p(k_t = j | k_{t-1} = i)$. The observation model (emission model) is sometimes modeled as a Gaussian such that $p(\boldsymbol{y}_t | k_t = k; \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{y}_t | \boldsymbol{\mu}_k, \Sigma_k)$.

The state posterior $p(k_t | \boldsymbol{y}_{1:T})$ is also called the filtered belief state and can be estimated recursively by the forward algorithm by

$$\begin{aligned} p(k_t | \boldsymbol{y}_t) &= \frac{p(\boldsymbol{y}_t | k_t) p(k_t | \boldsymbol{y}_{1:t-1})}{p(\boldsymbol{y}_t | \boldsymbol{y}_{1:t-1})} \\ &\propto p(\boldsymbol{y}_t | k_t) \sum_{k_t=1}^{K} p(k_t | k_{t-1}) p(k_{t-1} | \boldsymbol{y}_{t-1}). \quad (5) \end{aligned}$$

Let $\alpha_{t,k} \triangleq p(k_t = k | \boldsymbol{y}_{1:T})$ denote the belief for the kth state, define $\boldsymbol{\alpha}_t = [\ldots, \alpha_{t,k}, \ldots]^T$ as the K-dimensional belief state vector, and $\boldsymbol{\phi}_t = [\ldots, p(\boldsymbol{y}_t | k_t = k), \ldots]^T$ as the K-dimensional likelihood vector, respectively. Then (5) can be expressed as

$$\boldsymbol{\alpha}_t \propto \boldsymbol{\phi}_t \odot (\Psi^T \boldsymbol{\alpha}_{t-1}). \quad (6)$$

The filtered belief state vector can be obtained after normalizing the expression in (6) through dividing by the sum of values. We note that we call HMM states as regimes from now on to prevent terminological ambiguity with the hidden states of RNN.

In Section III, we introduce the Markovian RNN architecture with HMM-based switching between regimes. We also provide the equations and the sequential learning algorithm of our framework.

III. Novel RNN Structure

In this section, we introduce our novel contributions for sequential learning with RNNs. We provide the structure of the Markovian RNN, by describing the modified network with multiple internal regimes in Section III-A and HMM-based switching mechanism in Section III-B. We present the sequential learning algorithm for the introduced framework in Section III-C. The detailed schematic of the overall structure of our model is given in Fig. 1.

A. RNNs With Multiple Internal Regimes

Here, we describe the introduced Markovian RNN structure with multiple regimes, where each regime controls hidden state transition independently. To this end, we modify the conventional form given in (2) and (3) as

$$\begin{aligned} \boldsymbol{h}_t^{(k)} &= f_h\left(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t; \boldsymbol{\theta}_h^{(k)}\right) \\ &= f_h\left(\mathbf{W}_{hh}^{(k)}\boldsymbol{h}_{t-1} + \mathbf{W}_{xh}^{(k)}\boldsymbol{x}_t\right) \quad (7) \\ \hat{\boldsymbol{y}}_t^{(k)} &= f_y\left(\boldsymbol{h}_t^{(k)}; \boldsymbol{\theta}_y\right) \\ &= f_y\left(\mathbf{W}_{hy}\boldsymbol{h}_t^{(k)}\right) \quad (8) \end{aligned}$$

where $k \in \{1, \ldots, K\}$ is the regime index, and K is the number of regimes. We also illustrate the modified RNN cell with multiple regimes in the left-hand side of Fig. 1. Here, the hidden state vector is independently propagated to the next time step at each node with different weights $\boldsymbol{\theta}_h^{(k)}$. We highlight that the state-to-observation model is same for all regimes. However, the resulting predictions $\hat{\boldsymbol{y}}_t^{(k)}$ are still different for each regime because of different hidden states $\boldsymbol{h}_t^{(k)}$ obtained for the tth time step.

We obtain the final estimate of the hidden state at time step t by the weighted average of hidden states of each regime as

$$\boldsymbol{h}_t = \sum_{k=1}^{K} w_{t,k} \boldsymbol{h}_t^{(k)} \quad (9)$$

where $w_{t,k}$ is the weight for the kth regime. Finally, we estimate the output using (3). Here, the weights $w_{t,k}$ are determined by the switching mechanism described in Section III-B.

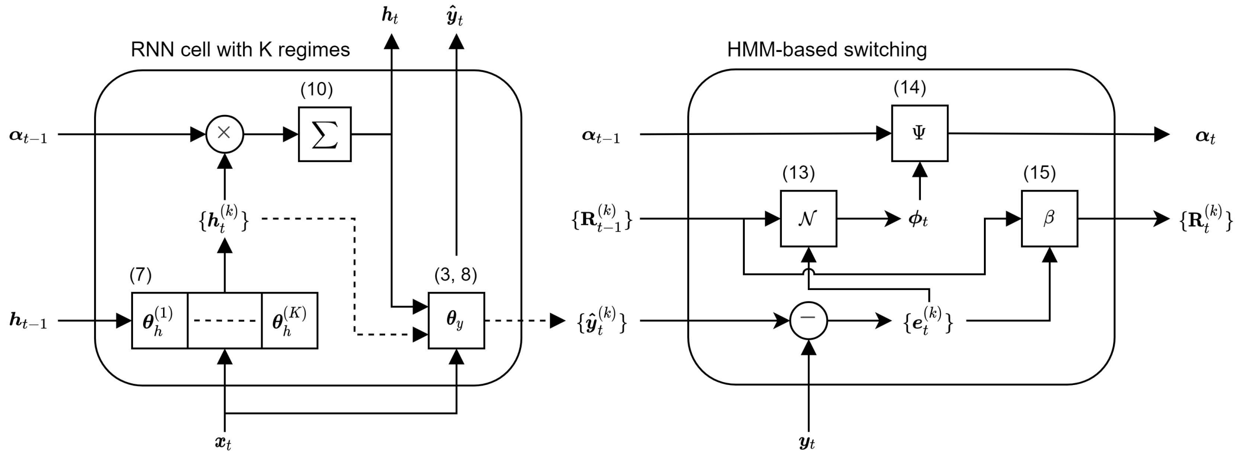


Fig. 1. Detailed schematic of Markovian RNN cell. Here, $\boldsymbol{x}_t$, $\boldsymbol{y}_t$, and $\hat{\boldsymbol{y}}_t$ are the input, target, and prediction vectors for the tth time step. $\boldsymbol{\alpha}_t$ and $\boldsymbol{h}_t$ are the belief state and hidden state vectors, respectively. $\mathbf{R}_t^{(k)}$ is the error covariance matrix for the kth regime.

The number of regimes, K, is considered as a hyperparameter and can be selected using cross-validation.

Remark 1: Our model can also be extended with different RNN structures such as LSTM [28] and GRU [29]. For instance, for LSTM, all gating operations and cell/hidden state updates can be performed for each internal regime with the following equations:

$$\boldsymbol{c}_t^{(k)} = \boldsymbol{c}_{t-1}^{(k)} \odot \boldsymbol{f}_t^{(k)} + \tilde{\boldsymbol{c}}_t^{(k)} \odot \boldsymbol{i}_t^{(k)}$$
$$\boldsymbol{h}_t^{(k)} = f_h\left(\boldsymbol{c}_t^{(k)}\right) \odot \boldsymbol{o}_t^{(k)}$$
$$\hat{\boldsymbol{y}}_t^{(k)} = f_y\left(\mathbf{W}_{hy}\boldsymbol{h}_t^{(k)}\right)$$

where $\boldsymbol{f}_t^{(k)}$, $\boldsymbol{i}_t^{(k)}$, and $\boldsymbol{o}_t^{(k)}$ are the forget, input, and output gates, and $\tilde{\boldsymbol{c}}_t^{(k)}$ is the candidate cell state at time step t for the kth regime such that

$$\boldsymbol{f}_t^{(k)} = \sigma\left(\mathbf{W}_{xf}^{(k)}\boldsymbol{x}_t + \mathbf{W}_{fh}^{(k)}\boldsymbol{h}_{t-1}^{(k)}\right)$$
$$\boldsymbol{i}_t^{(k)} = \sigma\left(\mathbf{W}_{xi}^{(k)}\boldsymbol{x}_t + \mathbf{W}_{ih}^{(k)}\boldsymbol{h}_{t-1}^{(k)}\right)$$
$$\tilde{\boldsymbol{c}}_t^{(k)} = f_h\left(\mathbf{W}_{xg}^{(k)}\boldsymbol{x}_t + \mathbf{W}_{gh}^{(k)}\boldsymbol{h}_{t-1}^{(k)}\right)$$
$$\boldsymbol{o}_t^{(k)} = \sigma\left(\mathbf{W}_{xo}^{(k)}\boldsymbol{x}_t + \mathbf{W}_{oh}^{(k)}\boldsymbol{h}_{t-1}^{(k)}\right)$$

where σ and f_h are nonlinear element-wise activation functions. For the final estimates of the hidden state, we can apply (9). For the cell state, the same form is applicable as well

$$\boldsymbol{c}_t = \sum_{k=1}^{K} w_{t,k}\boldsymbol{c}_t^{(k)}.$$

The final output estimate $\hat{\boldsymbol{y}}_t$ can be calculated with (3).

B. HMM-Based Switching Mechanism

We employ an HMM to control the switching mechanism between internal regimes. In particular, we perform soft switching, where the weights given in (9) are represented using the belief values of the HMM as follows:

$$\boldsymbol{h}_t = \sum_{k=1}^{K} \alpha_{t-1,k}\boldsymbol{h}_t^{(k)} \tag{10}$$

where $\alpha_{t-1,k} \triangleq w_{t,k}$ denote the belief for the kth regime. To perform belief update as given in (6), we need to calculate the likelihood values of ϕ_t for the tth time step after observing $\boldsymbol{y}_t$. To this end, for MSE loss, we consider the error model with Gaussian distribution such that

$$\boldsymbol{y}_t = \hat{\boldsymbol{y}}_t^{(k)} + \boldsymbol{e}_t^{(k)} \tag{11}$$
$$\boldsymbol{e}_t^{(k)} \sim \mathcal{N}\left(0, \mathbf{R}_{t-1}^{(k)}\right) \tag{12}$$

where $\boldsymbol{e}_t^{(k)}$ is the error vector and $\mathbf{R}_{t-1}^{(k)}$ is the error covariance matrix for the kth regime, which stores the errors of the corresponding regime until the tth time step, excluding the last step. Then we compute the likelihood by

$$p\left(\boldsymbol{y}_t|k_t = k\right) = \frac{1}{\sqrt{(2\pi)^{n_y}|\mathbf{R}_{t-1}^{(k)}|}}\exp\left(-\frac{1}{2}\boldsymbol{e}_t^{(k)^T}\mathbf{R}_{t-1}^{(k)^{-1}}\boldsymbol{e}_t^{(k)}\right). \tag{13}$$

Once we obtain the likelihoods, we update the regime belief vector using (6) as

$$\tilde{\boldsymbol{\alpha}}_t = \boldsymbol{\phi}_t \odot \left(\Psi^T\boldsymbol{\alpha}_{t-1}\right)$$
$$\boldsymbol{\alpha}_t = \tilde{\boldsymbol{\alpha}}_t/\text{sum}(\tilde{\boldsymbol{\alpha}}_t) \tag{14}$$

where we calculate $\boldsymbol{\phi}_t = [\ldots, p(\boldsymbol{y}_t|k_t = k), \ldots]^T$ with (13). We finally update the error covariance matrix using exponential smoothing by

$$\mathbf{R}_t^{(k)} = (1-\beta)\mathbf{R}_{t-1}^{(k)} + \beta\boldsymbol{e}_t^{(k)}\boldsymbol{e}_t^{(k)^T} \tag{15}$$

where $\beta \in [0, 1)$ controls the smoothing effect, which can be selected using cross validation. For instance, $\beta = 0.95$ would result in high sensitivity to errors, which can cause outlier data to bring frequent oscillations between regimes, whereas very small values for β might prevent the system to capture fast switches. The second part of the schematic in Fig. 1 illustrates the operations of HMM-based switching module.

Remark 2: Our frameworks can also be used with different loss functions such as MAE loss. In this case, we can model the distribution of the error vector $\boldsymbol{e}_t$ with the multivariate Laplacian distribution. The computation of regime likelihoods given in (13) can be modified for the multivariate Laplacian distribution as

$$p\left(\boldsymbol{y}_t|k_t = k\right) = \frac{2}{\sqrt{(2\pi)^{n_y}|\mathbf{R}_{t-1}^{(k)}|}} K_\upsilon\left(\sqrt{2}\rho\right)\left(-\frac{\rho}{2}\right)^{n_y/2}$$

where $\rho = \boldsymbol{e}_t^{(k)^T}\mathbf{R}_{t-1}^{(k)^{-1}}\boldsymbol{e}_t^{(k)}$, $\upsilon = 1 - n_y/2$ and K_υ is the modified Bessel function of the second kind [42]. For 1-D case ($n_y = 1$), considering scalars instead of vectors, the likelihood equation reduces to

$$p(y_t|k_t = k) = \frac{1}{2r_{t-1}^{(k)}}\exp\left(-\frac{|e_t^{(k)}|}{r_{t-1}^{(k)}}\right)$$

where e_t^k and r_t^k are the error value and error variance at the tth time step for the kth regime.

Remark 3: HMM-based switching inherently prevents instability due to the frequent oscillations between regimes or possible saturations at well-fitted regimes. One might argue that certain regimes that have been explored heavily during the early stages of the training would dominate other regimes and cause the system to degenerate into quite a few regimes. However, since the error covariance matrix penalizes errors for well-fit regimes more harshly than the regimes that are not explored yet, the model will tend to switch to other regimes if the predictions made by the dominating regimes start to produce high errors. Here, the choice of the smoothing parameter β can be interpreted as an adjuster of the tolerance for the errors made in different regimes. As β increases, the switching mechanism will have greater sensitivity to errors, which can cause instability and high deviations in the regime belief vector. Likewise, as β approaches toward zero, the system will not be able to capture switches due to the smoothing effect. This can eventually lead to saturations at well-fitted regimes. Thus, the choice of β directly affects the behavior of our model and we can readily tune it depending on the needs of the specific application or select it with cross-validation. We further discuss and illustrate the effect of this parameter in Section IV-C.

C. Sequential Learning Algorithm for Markovian RNN

In this section, we describe the learning algorithm of the introduced framework. During training, at each time step t, our model predicts the hidden state $\boldsymbol{h}_t$ and output $\hat{\boldsymbol{y}}_t$. We receive the loss given in (1) after observing the target output $\boldsymbol{y}_t$. We denote the set of weights of our model as $\boldsymbol{\theta} = \left\{\{\boldsymbol{\theta}_h^{(k)}\}_{k=1}^K, \boldsymbol{\theta}_y, \Psi\right\}$. We use the gradient descent algorithm to jointly optimize the weights during the training.

In Algorithm 1, we present the sequential learning algorithm for the introduced Markovian RNN. First, we initialize the model weights $\boldsymbol{\theta}$, hidden state $\boldsymbol{h}_1$, regime belief vector $\boldsymbol{\alpha}_1$ and error covariance matrices $\left\{\mathbf{R}_1^{(k)}\right\}_{k=1}^K$. For a given sequence with temporal length T, after receiving input $\boldsymbol{x}_t$ at each time step t, we compute hidden states for each internal regime using (7). Then, we predict the output with these hidden states for each regime using (8). After forward-pass of each internal regime, we generate $\boldsymbol{h}_t$ and output prediction $\hat{\boldsymbol{y}}_t$ using (10) and (3). After receiving the target output $\boldsymbol{y}_t$, we compute the loss using

Algorithm 1 Sequential Learning Algorithm for Markovian RNN

```
1: Input: Input and target time series: $\{\boldsymbol{x}_t\}_{t=1}^T$ and $\{\boldsymbol{y}_t\}_{t=1}^T$.
2: Parameters: Error covariance update term $\beta \in [0, 1)$, learning rate $\eta \in \mathbb{R}^+$, number of epochs $n \in \mathbb{N}^+$, early stop tolerance $n_{tolerance} \in \mathbb{N}$, training/validation set durations $T_{train}$ and $T_{val}$.
3: Initialize: $\boldsymbol{\theta}$ (weights).
4: Initialize: $\boldsymbol{\theta}_{best} = \boldsymbol{\theta}$ (best weights)
5: Initialize: $\upsilon = \infty$ (lowest validation loss)
6: Initialize: $j = 0$ (counter for early stop)
7: for epoch $e = 1$ to $n$ do
8:   Training Phase:
9:   for time step $t = 1$ to $T_{train}$ do
10:      Initialize: $\boldsymbol{h}_1$, $\boldsymbol{\alpha}_1$ and $\{\mathbf{R}_t^{(k)}\}_{k=1}^K$.
11:      RNN Cell Forward Pass:
12:      Receive $\boldsymbol{x}_t$
13:      for regime $k = 1$ to $K$ do
14:         $\boldsymbol{h}_t^{(k)} = f_h(\mathbf{W}_{hh}^{(k)}\boldsymbol{h}_{t-1} + \mathbf{W}_{xh}^{(k)}\boldsymbol{x}_t)$
15:         $\hat{\boldsymbol{y}}_t^{(k)} = f_y(\mathbf{W}_{hy}\boldsymbol{h}_t^{(k)})$
16:      end for
17:      $\boldsymbol{h}_t = \sum_{k=1}^K \alpha_{t-1,k}\boldsymbol{h}_t^{(k)}$
18:      $\hat{\boldsymbol{y}}_t = f_y(\mathbf{W}_{hy}\boldsymbol{h}_t)$
19:      Calculate Loss:
20:      Receive $\boldsymbol{y}_t$
21:      $\boldsymbol{e}_t = \boldsymbol{y}_t - \hat{\boldsymbol{y}}_t$
22:      $\ell_{\text{MSE}}(\boldsymbol{y}_t, \hat{\boldsymbol{y}}_t) = \boldsymbol{e}_t^T\boldsymbol{e}_t$
23:      Backward Pass:
24:      Update model weights via backpropagation using $\frac{\partial\ell}{\partial\boldsymbol{\theta}}$
25:      $\Psi_{ij} \leftarrow \exp(\Psi_{ij})/\sum_{j'=1}^K \exp(\Psi_{ij'})$
26:      HMM-based Switching:
27:      $\boldsymbol{\phi}_t = [\ldots, p(\boldsymbol{y}_t|k_t = k), \ldots,]^T$ from (13)
28:      $\tilde{\boldsymbol{\alpha}}_t = \boldsymbol{\phi}_t \odot (\Psi^T\boldsymbol{\alpha}_{t-1})$
29:      $\boldsymbol{\alpha}_t = \tilde{\boldsymbol{\alpha}}_t/\text{sum}(\tilde{\boldsymbol{\alpha}}_t)$
30:      $\boldsymbol{e}_t^{(k)} = \boldsymbol{y}_t - \hat{\boldsymbol{y}}_t^{(k)}$
31:      $\mathbf{R}_t^{(k)} = (1-\beta)\mathbf{R}_{t-1}^{(k)} + \beta\boldsymbol{e}^{(k)}\boldsymbol{e}^{(k)^T}$
32:   end for
33:   Validation Phase:
34:   $L_{val} = 0$ (validation loss)
35:   for time step $t = T_{train}$ to $T_{train} + T_{val}$ do
36:      Make predictions $\hat{\boldsymbol{y}}_t$ using (3)-(15).
37:      $L_{val} = L_{val} + \ell_{\text{MSE}}(\boldsymbol{y}_t, \hat{\boldsymbol{y}}_t)$
38:   end for
39:   $\bar{L}_{val} = \frac{L_{val}}{T_{val}}$
40:   if $\bar{L}_{val} < \upsilon$ then
41:      $\upsilon = \bar{L}_{val}$
42:      $\boldsymbol{\theta}_{best} = \boldsymbol{\theta}$
43:      $j = 0$
44:   else
45:      $j = j + 1$
46:   end if
47:   if $j > n_{tolerance}$ then
48:      return $\boldsymbol{\theta}_{best}$
49:   end if
50: end for
51: return $\boldsymbol{\theta}_{best}$
```

(1) and update the model weights through the backpropagation of the derivatives. We provide the derivatives of model weights in Appendix A. To satisfy the requirement that each row of Ψ should sum to one, we scale the values row-wise using softmax function such that $\Psi_{ij} \leftarrow \exp(\Psi_{ij})/\sum_{j'=1}^{K}\exp(\Psi_{ij'})$. Finally, we update the regime belief vector and error covariance matrices by (13)–(15).

Remark 4: Our model introduces more parameters depending on the number of regimes. In our approach, we use truncated backpropagation through time (TBPTT) algorithm, which results in $O(n_h^2)$ weights, $O(n_h\tau)$ space complexity and $O(n_h^2\tau)$ time complexity for vanilla RNN[1] [43]. In Markovian RNN, each regime has separate state transition parameters ($\{\boldsymbol{\theta}_h^{(k)}\}_{k=1}^{K}$). Therefore, we have $O(Kn_h^2)$ weights, $O(Kn_h\tau)$ space complexity and $O(Kn_h^2\tau)$ time complexity for our model, i.e., the computational complexity increases linearly with the number of regimes. Even though the computation of the likelihood in (13) can be computationally expensive due to determinant and matrix inversion operations, we do not suffer in practice since n_y is usually small or 1 as in our experiments in Section IV.

IV. Simulations

In this section, we demonstrate the performance of the introduced Markovian RNN model and its extensions both on real and synthetic datasets. We show the improvements achieved by our model by comparing the performance with vanilla RNN, GRU, LSTM, conventional methods such as ARIMA, Markovian switching ARIMA (MS-ARIMA) [20], KNS model [21], and Filardo model with time-varying transition probabilities (TVTPs) [22]. We also consider recent successful methods such as Prophet [31] that employ a decomposition-based statistical approach and NBeats [32] that has a deep architecture with stacked fully connected blocks with residuals. In the first part, we simulate three synthetic sequences with two regimes and analyze the inference and switching capacity of our model under different scenarios. Then, we investigate the effect of a number of regimes on performance improvement. In the second set of experiments, we demonstrate the performance enhancement obtained by our method in six real-life datasets and compare our results with the results of other methods. Also, we investigate the inferred regimes for given sequences by interpreting the temporal evolution of regime beliefs and the switching behavior of Markovian RNN.

For real dataset experiments, we report test errors in terms of MSE, MAE and mean absolute percentage error (MAPE). Here, MAPE provides a scale-invariant measure that enables comparisons across datasets. The expression for MAPE is given as $L_{\text{MAPE}} = (100/T)\sum_{t=1}^{T}((|y_t - \hat{y}_t|)/(y_t))$, where y_t is the target value and $\hat{y}_t$ is the predicted value at the tth time step.

For synthetic dataset experiments, using MAPE measure is not feasible since the generated sequences consist of real numbers and MAPE behaves very inconsistent due to the division operation for the series that contain values close to zero. Therefore, we report another scale-independent measure, mean absolute scaled error (MASE) that provides the accuracy of forecasts by comparing them with naive forecasts ($\hat{y}_t = y_{t-1}$). It is defined as follows:

$$L_{\text{MASE}} = \frac{\frac{1}{T}\sum_{t=1}^{T}|y_t - \hat{y}_t|}{\frac{1}{T-1}|y_t - y_{t-1}|}.$$

Here, if the MASE score is greater than or equal to 1, it means that the prediction model does not improve over naive forecasts.

A. Synthetic Dataset Experiments

In order to analyze the capability of Markovian RNN to detect different regimes, and to investigate the switching behavior between these regimes, we conduct initial experiments on synthetic data. We first describe the simulation setups for synthetic data generation and then, present the results obtained by all methods on the synthetic datasets.

1) Simulation Setups: In the synthetic data experiments, our goal is to predict the output y_t given the input data $x_{1:t}$ such that $x_t \in \mathbb{R}$ is a scalar. The output data are given by $y_t = x_{t+1}$. Here, the goal of these experiments is to conceptually show the effectiveness of our algorithm. In order to demonstrate the learning behavior of our algorithm with different patterns and switching scenarios, simulated sequences should have various regimes, where each regime possesses different temporal statistics. To this end, we conduct three experiments in which we simulate autoregressive processes with deterministic and Markovian switching, and a sinusoidal with Markovian switching.

a) Autoregressive process with deterministic switching: In the first synthetic dataset experiment, we aim to generate a sequence with sharp transitions and obvious distinctions between regimes. To this end, we generate an autoregressive (AR) process with deterministic switching, which is given with the following set of equations:

$$x_{t+1} = \begin{cases} x_t + \epsilon & \text{if} \quad \bmod(t, 1000) < 500 \\ -0.9\,x_t + \epsilon & \text{if} \quad \bmod(t, 1000) \geq 500 \end{cases} \tag{16}$$

where $x_t \in \mathbb{R}$ is the value of the time series at the tth time step, and $\epsilon \sim \mathcal{N}(0, 0.01)$ is the process noise. Here, (16) describes an AR process with two equal-duration regimes with temporal length 500, in which the system oscillates between. The first regime describes a random walk process, whereas the second regime gradually drifts toward white noise. The simulated system is deterministic in terms of switching mechanism between regimes since it strictly depends on the time step. Fig. 2(a) demonstrates the time series generated by this setup.

b) Autoregressive process with markovian switching: In this setup, we consider Markovian switching instead of deterministic switching. Here, the transition between regimes has Markovian property, therefore the regime of next time step only depends on the current regime. We consider third-order AR processes with the coefficients of $\{0.95, 0.5, -0.5\}$

[1]We use big O notation, i.e., $g(n_h) = O(f(n_h))$, to describe the limiting behavior as $n_h \gg n_y$, where n_y is the number of output dimensions.

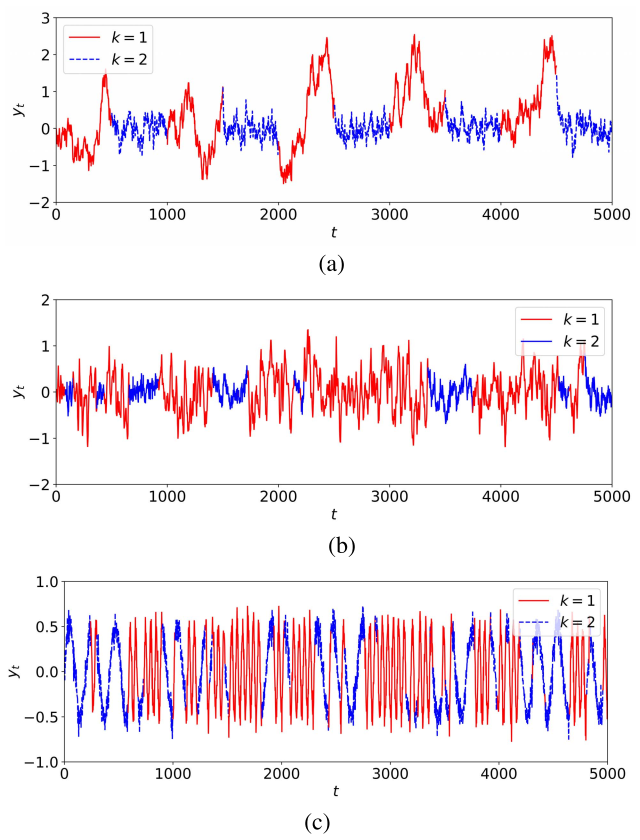


Fig. 2. Illustrations of simulated sequences for synthetic dataset experiments. Red color is used for the first regime and blue color is used for the second regime. (a) AR(1) process with two regimes and deterministic switching. (b) AR(3) process with two regimes and Markovian switching. (c) Sinusoidal process with two regimes and Markovian switching.

and $\{0.95, -0.5, 0.5\}$ for each regime, respectively, and $\epsilon \sim \mathcal{N}(0, 0.01)$. For the transition matrix, we consider

$$\Psi = \begin{bmatrix} 0.998 & 0.002 \\ 0.004 & 0.996 \end{bmatrix}.$$

Fig. 2(b) demonstrates the time series generated by this setup.

c) Sinusoidal process with markovian switching: In this experiment, we generate a noisy sinusoidal signal with two regimes, where every regime represents a different frequency. Here, the simulated signal has two regimes with the magnitude of 0.5 and periods of 50 and 200 for the generated sinusoidals. The whole sequence consists of 5000 time steps and Markovian switching is controlled by the transition matrix

$$\Psi = \begin{bmatrix} 0.99 & 0.01 \\ 0.01 & 0.99 \end{bmatrix}.$$

We also scale the magnitude to half and add Gaussian noise to the generated signal ($\epsilon \sim \mathcal{N}(0, 0.0025)$).

2) Synthetic Dataset Performance: Here, we present the training procedure and the results of the methods in terms of RMSE, MAE, and MASE. In these experiments, each synthetic time series has 5000 time steps of temporal length. We split the data into three splits for training (60%), validation (20%), and test (20%). We perform training on the training set and choose the best configuration based on the performance in the validation set. Then, we compare the test results of the best configuration for each method. We also perform early stopping based on validation error such that we stop the training if the loss does not decrease for 20 consecutive epochs or the number of epochs reaches 200.

In Table I, we provide the test RMSE, MAE, and MASE obtained by each method on the synthetic datasets. We also provide the results for real dataset experiments in Table II and III, and discuss in Section IV-B in more detail. In all setups, our model performs significantly better than other methods. Regardless of the switching mechanism and process dynamics, our model brings considerable improvements. The performance KNS model is not competitive with other methods, since it relies on switching variance between regimes. TVTP also has the form of MS-ARIMA, but it also models the transition probabilities as temporally varying values. In our setups, the transition probabilities are fixed, hence, this method does not bring any improvement over MS-ARIMA. MS-ARIMA works significantly more accurately than standard ARIMA as expected. Likewise, Markovian RNN enjoys the benefits of adaptive HMM-based switching, which improves the predictions compared to the predictions of vanilla RNN. We also illustrate regime belief values of our model for AR process sequence with deterministic switching in Fig 3. Our model is able to detect and switch between the regimes properly. This behavior is further discussed in Section IV-C.

3) Effect of the Number of Regimes: In this part, we investigate the effect of number of regimes on the performance of our model. To this end, we focused on Markovian switching AR process simulations with 2, 5 and 10 regimes. All sequences have the same length $T = 5000$ and variance $\sigma^2 = 0.01$. We set the probability of staying at the same regime as 0.98 and probabilities of transition to other regimes to have the same value. We set AR coefficients as $((0.95, 0.5, -0.5), (0.95, -0.5, 0.5))$, $((0.95, 0.5, -0.5), (0.95, -0.5, 0.5), (1), (-0.9), (0.5)))$ and $((0.95, 0.5, -0.5), (0.95, -0.5, 0.5), (1), (-0.9), (0.5), (0.9), (-0.5), (-1), (0.75, 0.25), (0.25, 0.75))$, respectively.

To investigate the improvement for a different number of regimes, we have compared vanilla RNN with Markovian RNN. In addition, we have considered a different number of regimes for our model to see how it performs if the number of regimes is overestimated or underestimated, although we select it through cross-validation for other experiments. We also included the results obtained for the RNN ensemble that contains 10 RNNs trained with different random initializations and considered the mean of outputs as the final prediction.

As shown in Table II, the performance gap between our method and vanilla RNN increases as the number of regimes gets higher. For instance, MASE improvements were 0.04, 0.052 and 0.089 for $K = 2$, $K = 5$ and $K = 10$, respectively. In addition, we observe that significantly underestimating or overestimating the number of regimes can degrade the performance, however, our model configured with 5 regimes obtains very close scores to the best configurations for sequences with 2 and 10 regimes. Furthermore, the RNN ensemble cannot surpass our model although it gets

TABLE I

SYNTHETIC DATASET EXPERIMENT RESULTS FOR BASELINE METHODS AND THE INTRODUCED MARKOVIAN RNN IN TERMS OF RMSE, MAE, AND MASE

Simulations	AR (deterministic)			AR (markovian)			Sinusoidal (markovian)		
Methods	RMSE	MAE	MASE	RMSE	MAE	MASE	RMSE	MAE	MASE
ARIMA	0.333	0.228	0.539	0.183	0.148	0.913	0.136	0.108	0.866
MS-ARIMA [20]	0.206	0.145	0.477	0.148	0.120	**0.824**	0.128	0.103	0.839
KNS [21]	0.447	0.271	0.550	0.196	0.155	1.110	0.142	0.114	0.890
TVTP [22]	0.206	0.145	0.500	0.160	0.129	0.891	0.136	0.108	0.840
RNN [30]	0.193	0.134	0.474	0.146	0.113	0.844	0.126	0.099	**0.795**
Markov-RNN	**0.178**	**0.120**	**0.458**	**0.126**	**0.097**	0.836	**0.121**	**0.091**	0.801

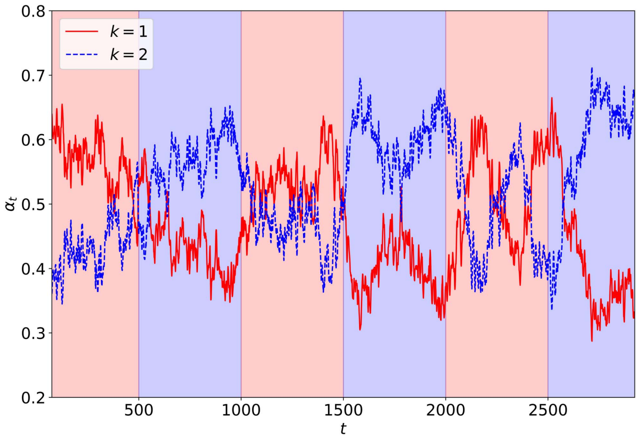


Fig. 3. Regime beliefs of Markovian RNN for AR process sequence with deterministic switching. Here, background colors represent the real regime value, where red color is used for the first regime and blue color is used for the second regime. Our model can properly distinguish between the two regimes except for a short-term undesired switch around $t = 2300$, thus the resulting regime belief values are consistent with the real regimes.

TABLE II

RESULTS FOR SIMULATIONS OF AR PROCESSES WITH DIFFERENT NUMBER OF REGIMES IN TERMS OF RMSE, MAE, AND MASE

Simulations	K=2			K=5			K=10		
Methods	RMSE	MAE	MASE	RMSE	MAE	MASE	RMSE	MAE	MASE
ARIMA	0.18	0.14	0.91	0.18	0.14	0.90	0.21	0.17	0.94
RNN	0.14	0.11	0.79	0.15	0.12	0.80	0.18	0.14	0.89
RNN Ensemble	0.14	0.11	0.78	0.14	0.11	0.76	0.16	0.14	0.83
Markov-RNN-2	**0.13**	0.10	0.76	0.14	0.11	0.76	0.17	0.14	0.85
Markov-RNN-5	0.13	**0.10**	**0.75**	**0.13**	**0.11**	**0.74**	0.15	**0.13**	0.80
Markov-RNN-10	0.16	0.12	0.79	0.14	0.11	0.77	0.**15**	0.13	**0.80**

significantly better results than a single RNN, which shows the effectiveness of Markovian switching employed in our model compared to straightforward ensembling procedures while adapting statistical changes in data.

B. Real Dataset Experiments

In this section, we provide the performance of our model and other methods in six real-life datasets. We also investigate the regime switching behavior of our model and the effect of error covariance smoothing parameter (β) introduced in (15) and mentioned in Remark 3. We consider the following datasets:

1) *USD/EUR*, *USD/GBP* and *USD/TRY* currency exchange ratio data [44]: We use the data from January 1, 2010 to January 24, 2020 with hourly resolution. We utilize the mean and variance of prices for every day and work in daily resolution. The goal is to predict the average currency rate for the next day.
2) *M5 Forecasting Competition Sales Dataset* [45]: The dataset is available for M5 Forecasting Competition in Kaggle and covers the daily item sales for Walmart stores in the USA. It contains the records from January 29, 2011, to May 22, 2016. We consider the aggregated time series in daily resolution.
3) *UCI Electricity Load Dataset* [46]: The dataset contains the electricity usage of 370 customers covering the period from January 1, 2011, and January 1, 2015. We consider the aggregated time series in daily resolution starting from 2012 since some records in 2011 were missing.
4) *UCI PEMS-SF Traffic Dataset* [47]: The dataset contains the occupancy rate of 440 freeways in the San Francisco Bay Area and spans the dates between January 1, 2008, and March 30, 2009. We consider the mean aggregated time series in hourly resolution.

For all experiments, we split the data into three splits for training (60%), validation (20%), and test (20%). We also repeat the experiments 10 times to reduce random effects and provide the standard deviation of MAPE scores in Tables III and IV. We take the first-order difference of the data to decrease the trend effect. Before testing, we calibrate the predictions by fitting a linear regressor on the validation set for all models. We also perform early stopping on validation error such that we stop the training if the loss does not decrease for 20 consecutive epochs or the number of epochs reaches 200. We provide the hyperparameters in Appendix B.

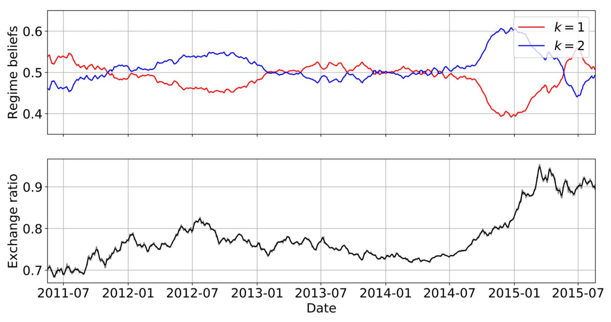
In Tables III and IV, we provide the test RMSE, MAE, and MAPE values obtained by each method on *USD/EUR*, *USD/TRY*, *USD/GBP*, *Sales*, *Electricity*, and *Traffic* datasets. For currency exchange ratio dataset experiments, all methods perform worst in *USD/TRY* due to the high oscillations in the sequence. We observe that the performance gap between our model and ARIMA-based models is greater in *Sales*, *Electricity*, and *Traffic* datasets probably due to the greater randomness effects in currency exchange ratio datasets. In all cases except the MAPE score in the *Electricity* dataset, Markovian RNN and its extensions perform significantly better than other methods. Only in the *Electricity* dataset, our model gives a slightly higher (+0.01%) MAPE than N-Beats [32]. Since our models are directly trained to minimize the MSE,

TABLE III
USD/EUR, USD/GBP, AND USD/TRY DATASET EXPERIMENT RESULTS IN TERMS OF RMSE, MAE, AND MAPE

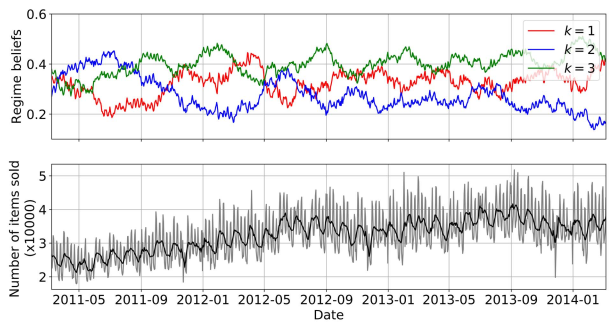
Datasets	*USD/EUR*			*USD/TRY*			*USD/GBP*		
Methods	RMSE	MAE	MAPE (%)	RMSE	MAE	MAPE (%)	RMSE	MAE	MAPE (%)
ARIMA	2.34E-3	1.78E-3	2.19 ± 0.00	5.42E-2	2.67E-2	5.57 ± 0.00	2.69E-3	1.98E-3	2.88 ± 0.00
MS-ARIMA [20]	2.28E-3	1.73E-3	1.94 ± 0.00	5.34E-2	2.63E-2	5.55 ± 0.00	2.53E-3	1.88E-3	2.49 ± 0.00
KNS [21]	2.14E-3	1.54E-3	1.73 ± 0.00	5.40E-2	2.57E-2	5.61 ± 0.00	2.61E-3	1.85E-3	2.61 ± 0.00
TVTP [22]	2.37E-3	1.81E-3	1.99 ± 0.00	5.29E-2	2.50E-2	5.19 ± 0.00	2.47E-3	1.79E-3	2.30 ± 0.00
RNN [30]	2.24E-3	1.68E-3	1.76 ± 0.03	5.36E-2	2.57E-2	4.80 ± 0.07	2.60E-3	1.84E-3	2.32 ± 0.01
GRU [29]	2.06E-3	1.58E-3	1.81 ± 0.02	5.20E-2	2.65E-2	4.85 ± 0.05	2.35E-3	1.75E-3	2.28 ± 0.02
LSTM [28]	2.04E-3	1.53E-3	1.77 ± 0.04	5.32E-2	2.65E-2	4.83 ± 0.09	2.37E-3	1.74E-3	2.28 ± 0.03
Prophet [31]	2.18E-3	1.60E-3	1.75 ± 0.00	5.26E-2	2.68E-2	4.78 ± 0.00	2.54E-3	1.82E-3	2.40 ± 0.00
N-Beats [32]	2.10E-3	1.59E-3	1.71 ± 0.03	5.24E-2	2.57E-2	4.68 ± 0.04	2.41E-3	1.76E-3	2.33 ± 0.01
Markov-RNN	2.02E-3	1.51E-3	**1.65 ± 0.03**	5.10E-2	**2.34E-2**	4.64 ± 0.05	2.34E-3	1.69E-3	2.18 ± 0.04
Markov-GRU	1.99E-3	1.49E-3	1.73 ± 0.03	5.16E-2	2.44E-2	4.68 ± 0.02	**2.15E-3**	**1.57E-3**	**2.07 ± 0.01**
Markov-LSTM	**1.95E-3**	**1.47E-3**	1.67 ± 0.04	**5.10E-2**	2.47E-2	**4.63 ± 0.07**	2.19E-3	1.68E-3	2.15 ± 0.03

TABLE IV
SALES, ELECTRICITY, AND TRAFFIC DATASET EXPERIMENT RESULTS IN TERMS OF RMSE, MAE, AND MAPE

Datasets	*Sales*			*Electricity*			*Traffic*		
Methods	RMSE	MAE	MAPE (%)	RMSE	MAE	MAPE (%)	RMSE	MAE	MAPE (%)
ARIMA	5639.8	4527.4	11.32 ± 0.00	4033.6	2914.9	5.34 ± 0.00	7.45E-3	6.33E-3	7.07 ± 0.00
MS-ARIMA [20]	4921.0	4032.3	9.91 ± 0.00	3940.9	2893.3	5.19 ± 0.00	7.73E-3	5.70E-3	6.44 ± 0.00
KNS [21]	4900.1	3914.5	10.04 ± 0.00	4172.1	2998.0	5.38 ± 0.00	7.82E-3	5.63E-3	6.38 ± 0.00
TVTP [22]	4765.3	3394.2	8.29 ± 0.00	4001.3	2794.6	5.19 ± 0.00	7.92E-3	5.95E-3	7.92 ± 0.00
RNN [30]	3508.2	2574.6	6.61 ± 0.03	4026.3	2906.6	5.27 ± 0.04	4.83E-3	3.52E-3	4.38 ± 0.02
GRU [29]	3456.1	2505.1	6.40 ± 0.02	3880.1	2746.9	5.25 ± 0.04	4.80E-3	3.48E-3	4.23 ± 0.02
LSTM [28]	3602.8	2666.4	6.65 ± 0.06	3892.7	2798.3	5.21 ± 0.06	4.73E-3	3.32E-3	4.14 ± 0.02
Prophet [31]	3416.8	2448.5	6.34 ± 0.00	4040.7	2901.9	5.29 ± 0.00	4.70E-3	3.38E-3	4.17 ± 0.00
N-Beats [32]	3435.7	2481.2	6.33 ± 0.02	3878.9	2777.5	**4.97 ± 0.02**	4.57E-3	3.35E-3	4.00 ± 0.01
Markov-RNN	**3348.3**	2458.1	6.15 ± 0.05	3871.0	2735.9	5.01 ± 0.04	4.63E-3	3.48E-3	4.36 ± 0.04
Markov-GRU	3400.8	2478.8	6.31 ± 0.02	**3790.6**	**2693.3**	4.98 ± 0.04	4.42E-3	3.25E-3	3.92 ± 0.02
Markov-LSTM	3371.8	**2331.5**	**6.10 ± 0.05**	3828.9	2771.7	5.10 ± 0.09	**4.33E-3**	**3.22E-3**	**3.85 ± 0.03**



(a)



(b)

Fig. 4. Filtered regime beliefs of Markovian RNN and data sequence for two experiments. Since the real regime values are not observable in real dataset experiments, consistency analysis is not possible. However, we still observe that our model switches between regimes in a stable way without any saturation. (a) We observe that there are certain periods in which the second regime dominates the predictions, especially when the market is in an uptrend. (b) Second regime seems comparably more significant during summers but gradually loses its dominance during 2013 and 2014. (a) Filtered regime beliefs of Markovian RNN and data sequence for USD/EUR. (b) Filtered regime beliefs of Markovian RNN and data sequence for Sales.

the performance improvement in terms of MAPE can be less significant. On the other hand, N-Beats relies on an ensembling approach that aggregates the predictions from submodels trained on different metrics, scales, and data subsets [32]. Therefore, it usually gives the most competitive and robust (in terms of metric) results among the comparison methods. Overall, our method obtains the lowest error values thanks to the efficient combination of nonlinear modeling capability of RNNs and adaptive switching controlled by HMM. Although the regimes that affect the sequential dynamics cannot be observed, our model can detect these regions and switch between them to adjust the predictions for nonstationarity. We further analyze the switching behavior of Markovian RNN in Section IV-C and investigate the inferred regime belief vectors.

C. Regime Switching Behavior of Markovian RNN

To investigate how our model detects the regimes and decides to switch from one to another, we illustrate the regime beliefs through time in Fig. 3 for AR process sequence with deterministic switching and in Fig. 4 for *USD/EUR* and *sales* datasets, respectively. In Fig. 3, we observe that our model

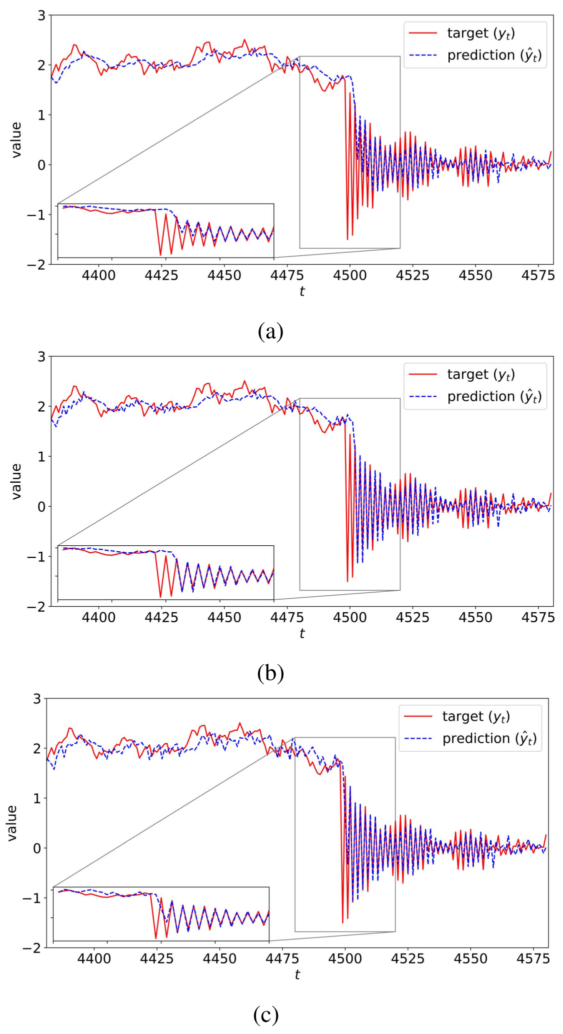


Fig. 5. Markovian RNN predictions on the test set of the AR process with deterministic switching, and the zoomed in plot at the regime switching region for different error covariance smoothing parameters. Here, our model can adaptively handle nonstationarity by switching between internal regimes during test. (a) $\beta = 0.5$. (b) $\beta = 0.7$. (c) $\beta = 0.9$.

can determine different regimes correctly and adjust the belief vector accordingly. During the analysis of real-life dataset experiments, checking the consistency between regime beliefs and real regimes is not possible, since they are not observable. In Fig. 4(a), we observe that there are certain periods in which the second regime dominates the predictions, especially when the market is in an uptrend. We can also observe how the model switches between three regimes for *Sales* dataset in Fig. 4(b). The second regime seems comparably more significant during summers but gradually loses its dominance during 2013 and 2014. Considering the switching behaviors in Fig. 4, we can interpret that our model does not suffer from rapid oscillations between regimes or does not saturate in a regime. The evolution of regime beliefs is stable but still responsive to nonstationarity.

In addition, we zoomed-in view to the predictions of Markovian RNN around switching regions in Fig. 5(a), and show that our model can successfully adapt to nonstationarity. As we mention in Remark 3, the error covariance smoothing parameter, β in (15) can be tuned with cross-validation to adjust the switching behavior of Markovian RNN. For instance, lower values of this parameter ($\beta = 0.5$) bring more toleration for prediction errors and may provide robustness against outliers with the expense of lagged transitions between regimes as shown in Fig. 5(a). On the contrary, higher values ($\beta = 0.9$) provide faster transitions but may cause oscillating predictions or lead to instability.

V. Future Directions

Our model can effectively combine Markovian switching with RNNs to improve robustness against nonstationarity and increase usability in real-life time series prediction applications. To further increase the practical value, improving the switching mechanism can be a promising research direction. Since we employ soft switching, the backward passes are executed for each regime during the training stage and forward passes are executed for each regime in both training and test stages. This approach introduces a computational cost that increases linearly with the number of regimes. To bring efficiency from this perspective, hard switching-based approaches using Bayesian optimization can also be considered. In addition, our method models the switching mechanism in a Markovian setting. Even though this approach is practically feasible for most real-life time series applications, it is also possible to design more generic switching architectures that can capture more complicated switching behaviors.

VI. Conclusion

We study nonlinear regression for time series prediction in nonstationary environments. We introduce a novel time series prediction network, Markovian RNN, which is an RNN with multiple internal regimes and, HMM-based switching. Each internal regime controls the hidden state transitions with different weights. We employ an HMM to control the switching mechanism between the internal regimes and jointly optimize the whole network in an end-to-end fashion. By combining the nonlinear representation capability of RNNs and the adaptivity obtained thanks to HMM-based switching, our model can capture nonlinear temporal patterns in highly nonstationary environments.

Through an extensive set of synthetic and real-life dataset experiments, we demonstrate the performance gains compared to the conventional econometric methods such as MS-ARIMA [20] and Filardo model [22], and recent successful approaches including Prophet [31] and N-Beats [32]. We show that the introduced model performs significantly better than other methods in terms of prediction RMSE, MAE, and MAPE thanks to the joint optimization and the efficient combination of nonlinear regression with RNNs, and HMM-based regime switching. Markovian RNN can properly determine the regimes and switch between them to make more accurate predictions. We also analyze the effect of the error covariance smoothing parameter and number of regimes on our model. As the experimental results and our analysis indicate, our model can capture nonlinear temporal patterns while successfully adapting nonstationarity without any instability or saturation issues.

APPENDIX A

In this part, we provide the derivatives of the model weights ($\boldsymbol{\theta} = \{\{\mathbf{W}_{xh}\}_{k=1}^{K}, \{\{\mathbf{W}_{hh}\}_{k=1}^{K}, \mathbf{W}_{hy}, \Psi\}$) of Markovian RNN. The equations of the basic derivatives are as follows:

$$\frac{\partial \ell_t}{\partial \hat{\boldsymbol{y}}_t} = -2\boldsymbol{e}_t^T \tag{17}$$

$$\frac{\partial \hat{\boldsymbol{y}}_t}{\partial \boldsymbol{h}_t} = \mathbf{W}_{hy} \tag{18}$$

$$\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_t^{(k)}} = \alpha_{t-1,k} \tag{19}$$

$$\frac{\partial \boldsymbol{h}_t^{(k)}}{\partial \boldsymbol{h}_{t-1}} = \mathbf{W}_{hh}^{(k)} \odot \operatorname{diag}\left(f_h'\left(\boldsymbol{z}_t^{(k)}\right)\right) \tag{20}$$

$$\frac{\partial \boldsymbol{h}_t}{\partial \alpha_{t-1,k}} = \boldsymbol{h}_t^{(k)} \tag{21}$$

$$\frac{\partial \alpha_{t,k}}{\partial \tilde{\alpha}_{t,k'}} = \frac{\delta_{kk'}\operatorname{sum}(\tilde{\boldsymbol{\alpha}}_t) - \tilde{\alpha}_{t,k}}{\operatorname{sum}(\tilde{\boldsymbol{\alpha}}_t)^2} \tag{22}$$

where $\boldsymbol{z}_t^{(k)} = \mathbf{W}_{hh}^{(k)}\boldsymbol{h}_{t-1} + \mathbf{W}_{xh}^{(k)}\boldsymbol{x}_t$. We use (19) and (20) to obtain the following:

$$\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t-\tau}} = \prod_{t'=t-\tau+1}^{t} \frac{\partial \boldsymbol{h}_t'}{\partial \boldsymbol{h}_{t'-1}} \tag{23}$$

where $\partial \boldsymbol{h}_t / \partial \boldsymbol{h}_{t-1} = \sum_{k=1}^{K} \alpha_{t-1,k}\mathbf{W}_{hh}^{(k)} \odot \operatorname{diag}\left(f_h'\left(\boldsymbol{z}_t^{(k)}\right)\right)$. Then, we can use (17)–(20) and (23) to obtain $\partial \ell_t / \partial \mathbf{W}_{xh}^{(k)}$ and $\partial \ell_t / \partial \mathbf{W}_{hh}^{(k)}$ as follows:

$$\frac{\partial \ell_t}{\partial \mathbf{W}_{xh}^{(k)}} = \sum_{t'=t-\tau}^{t} \frac{\partial \ell_t}{\partial \hat{\boldsymbol{y}}_t}\frac{\partial \hat{\boldsymbol{y}}_t}{\partial \boldsymbol{h}_t}\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t'}}\frac{\partial \boldsymbol{h}_{t'}}{\partial \boldsymbol{h}_{t'}^{(k)}}\frac{\partial \boldsymbol{h}_{t'}^{(k)}}{\partial \mathbf{W}_{xh}^{(k)}} \tag{24}$$

$$\frac{\partial \ell_t}{\partial \mathbf{W}_{hh}^{(k)}} = \sum_{t'=t-\tau}^{t} \frac{\partial \ell_t}{\partial \hat{\boldsymbol{y}}_t}\frac{\partial \hat{\boldsymbol{y}}_t}{\partial \boldsymbol{h}_t}\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t'}}\frac{\partial \boldsymbol{h}_{t'}}{\partial \boldsymbol{h}_{t'}^{(k)}}\frac{\partial \boldsymbol{h}_{t'}^{(k)}}{\partial \mathbf{W}_{hh}^{(k)}} \tag{25}$$

where $\partial \boldsymbol{h}_{t'}^{(k)} / \partial \mathbf{W}_{xh}^{(k)} = \left[\partial \boldsymbol{h}_{t'}^{(k)} / \partial W_{xh,ij}^{(k)}\right]$ such that $\partial \boldsymbol{h}_{t'}^{(k)} / \partial W_{xh,ij}^{(k)} = x_{t',j}\boldsymbol{d}_i \odot f_h'(\boldsymbol{z}_{t'}^{(k)})$, $\partial \boldsymbol{h}_{t'}^{(k)} / \partial \mathbf{W}_{hh}^{(k)} = \left[\partial \boldsymbol{h}_{t'}^{(k)} / \partial W_{hh,ij}^{(k)}\right]$ such that $\partial \boldsymbol{h}_{t'}^{(k)} / \partial W_{hh,ij}^{(k)} = h_{t'-1,j}\boldsymbol{d}_i \odot f_h'(\boldsymbol{z}_{t'}^{(k)})$, and τ is the truncation length. Here, $\boldsymbol{d}$ is a vector such that $d_{i'} = \delta_{ii'}$.

Using (17), we can calculate $\partial \ell_t / \partial \mathbf{W}_{hy}$ as

$$\frac{\partial \ell_t}{\partial \mathbf{W}_{hy}} = \frac{\partial \ell_t}{\partial \hat{\boldsymbol{y}}_t}\frac{\partial \hat{\boldsymbol{y}}_t}{\partial \mathbf{W}_{hy}} = -2\boldsymbol{e}_t\boldsymbol{h}_t^T. \tag{26}$$

Finally, we can compute the derivative of the transition matrix Ψ using (17), (18), (21) and (23) as follows:

$$\frac{\partial \ell_t}{\partial \Psi} = \sum_{t'=t-\tau}^{t} \frac{\partial \ell_t}{\partial \hat{\boldsymbol{y}}_t}\frac{\partial \hat{\boldsymbol{y}}_t}{\partial \boldsymbol{h}_t}\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t'}}\frac{\partial \boldsymbol{h}_{t'}}{\partial \Psi} \tag{27}$$

where $\partial \boldsymbol{h}_{t'} / \partial \Psi = \sum_{k=1}^{K} \boldsymbol{h}_{t'}^{(k)} \partial \alpha_{t'-1,k} / \partial \Psi$. We can express the derivative terms in this summation as

$$\frac{\partial \alpha_{t'-1,k}}{\partial \Psi} = \sum_{k'=1}^{K} \frac{\partial \alpha_{t'-1,k}}{\partial \tilde{\alpha}_{t'-1,k'}}\frac{\partial \tilde{\alpha}_{t'-1,k'}}{\partial \Psi} \tag{28}$$

where $\partial \tilde{\alpha}_{t'-1,k'} / \partial \Psi = \left[\partial \tilde{\alpha}_{t'-1,k'} / \partial \Psi_{ij}\right]$ such that $\partial \tilde{\alpha}_{t'-1,k'} / \partial \Psi_{ij} = \delta_{i,k'}\phi_{t-1,k'}\alpha_{t'-2,j}$, and $\partial \alpha_{t'-1,k} / \partial \tilde{\alpha}_{t'-1,k'}$ is given in (22).

TABLE V

HYPERPARAMETER SETTINGS THAT RESULTED IN THE BEST VALIDATION PERFORMANCE (n_h: NUMBER OF HIDDEN DIMENSIONS, ρ_0: CONCENTRATION PARAMETER, τ: TRUNCATION LENGTH, K: NUMBER OF REGIMES, η: LEARNING RATE). (A) MARKOVIAN RNN. (B) MARKOVIAN GRU. (C) MARKOVIAN LSTM.

Datasets	Hyperparameters
AR-Det.	$n_h = 16$, $\rho_0 = 0.5$, $\beta = 0.7$, $\tau = 4$, $K = 2$, $\eta = 0.0003$
AR-Markov	$n_h = 8$, $\rho_0 = 0.6$, $\beta = 0.5$, $\tau = 8$, $K = 2$, $\eta = 0.0001$
Sin-Markov	$n_h = 16$, $\rho_0 = 0.7$, $\beta = 0.9$, $\tau = 8$, $K = 2$, $\eta = 0.003$
USDEUR	$n_h = 32$, $\rho_0 = 0.7$, $\beta = 0.9$, $\tau = 32$, $K = 2$, $\eta = 0.003$
USDTRY	$n_h = 32$, $\rho_0 = 0.5$, $\beta = 0.5$, $\tau = 16$, $K = 3$, $\eta = 0.001$
USDGBP	$n_h = 64$, $\rho_0 = 0.7$, $\beta = 0.7$, $\tau = 64$, $K = 3$, $\eta = 0.0003$
Sales	$n_h = 16$, $\rho_0 = 0.7$, $\beta = 0.6$, $\tau = 16$, $K = 3$, $\eta = 0.003$
Electricity	$n_h = 32$, $\rho_0 = 0.7$, $\beta = 0.8$, $\tau = 8$, $K = 4$, $\eta = 0.003$
Traffic	$n_h = 16$, $\rho_0 = 0.5$, $\beta = 0.3$, $\tau = 16$, $K = 3$, $\eta = 0.001$

(a)

Datasets	Hyperparameters
USDEUR	$n_h = 16$, $\rho_0 = 0.7$, $\beta = 0.9$, $\tau = 32$, $K = 2$, $\eta = 0.003$
USDTRY	$n_h = 32$, $\rho_0 = 0.7$, $\beta = 0.6$, $\tau = 64$, $K = 3$, $\eta = 0.0002$
USDGBP	$n_h = 32$, $\rho_0 = 0.8$, $\beta = 0.7$, $\tau = 64$, $K = 3$, $\eta = 0.0006$
Sales	$n_h = 16$, $\rho_0 = 0.7$, $\beta = 0.6$, $\tau = 16$, $K = 3$, $\eta = 0.003$
Electricity	$n_h = 16$, $\rho_0 = 0.9$, $\beta = 0.9$, $\tau = 8$, $K = 4$, $\eta = 0.001$
Traffic	$n_h = 16$, $\rho_0 = 0.25$, $\beta = 0.1$, $\tau = 32$, $K = 3$, $\eta = 0.0006$

(b)

Datasets	Hyperparameters
USDEUR	$n_h = 16$, $\rho_0 = 0.95$, $\beta = 0.5$, $\tau = 64$, $K = 2$, $\eta = 0.003$
USDTRY	$n_h = 32$, $\rho_0 = 0.9$, $\beta = 0.5$, $\tau = 64$, $K = 3$, $\eta = 0.002$
USDGBP	$n_h = 32$, $\rho_0 = 0.8$, $\beta = 0.3$, $\tau = 64$, $K = 3$, $\eta = 0.0006$
Sales	$n_h = 16$, $\rho_0 = 0.9$, $\beta = 0.5$, $\tau = 100$, $K = 3$, $\eta = 0.001$
Electricity	$n_h = 16$, $\rho_0 = 0.9$, $\beta = 0.3$, $\tau = 32$, $K = 4$, $\eta = 0.002$
Traffic	$n_h = 8$, $\rho_0 = 0.5$, $\beta = 0.2$, $\tau = 64$, $K = 3$, $\eta = 0.0008$

(c)

APPENDIX B

Here, we provide the hyperparameter settings for our model in real dataset experiments. For the parameter search of RNN/GRU/LSTM and their Markovian variants, we perform grid search in the range of $n_h \in \{4, \ldots, 128\}$, $\tau \in \{2, \ldots, 100\}$, and $\eta = [0.00001, 0.03]$. For the number of internal regimes in our models, we search in $K \in \{2, \ldots, 5\}$ since we could not observe any improvement from increasing the number of regimes after $K = 5$. We employ Xavier initialization for weight initialization and set initial regime belief values to $1/K$. We initialize the rows of the transition matrix (Ψ) using the Dirichlet distribution with the concentration vector $\boldsymbol{\rho}^{(k)}$ such that $\rho_i^{(k)} = \rho_0$ if $i = k$ and $1 - \rho_0/K - 1$ otherwise, where k is the regime index. Here, ρ_0 determines the concentration over the diagonal elements of the initialized transition matrix. We search ρ_0 and β on the interval of $[0.1, 0.95]$. We provide the hyperparameters that resulted in the best validation performance for Markovian RNN, Markovian GRU, and Markovian LSTM in Table V for synthetic and real datasets. For ARIMA, MS-ARIMA [20], KNS [21], and TVTP [22], we determine the component order ranges through analyzing autocorrelation and partial autocorrelation plots, and use the parameters that result in the best validation performance. For these methods, we use the statsmodels package in Python [40]. For Prophet [31], we directly use the code provided by the authors and enable all decomposition options depending on the resolution of the dataset. For other models, we use the PyTorch framework.

REFERENCES

[1] T. Ergen and S. S. Kozat, "Efficient online learning algorithms based on LSTM neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 8, pp. 3772–3783, Aug. 2018.

[2] K. Greff, R. K. Srivastava, J. Koutnìk, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space Odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.

[3] S. E. Yuksel, J. N. Wilson, and P. D. Gader, "Twenty years of mixture of experts," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 8, pp. 1177–1193, Aug. 2012.

[4] L. Zhang, Y. Zhu, and W. X. Zheng, "Energy-to-peak state estimation for Markov jump RNNs with time-varying delays via nonsynchronous filter with nonstationary mode transitions," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 10, pp. 2346–2356, Oct. 2015.

[5] N. D. Vanli, M. O. Sayin, I. Delibalta, and S. S. Kozat, "Sequential nonlinear learning for distributed multiagent systems via extreme learning machines," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 546–558, Mar. 2017.

[6] M. Raginsky, R. M. Willett, C. Horn, J. Silva, and R. F. Marcia, "Sequential anomaly detection in the presence of noise and limited feedback," *IEEE Trans. Inf. Theory*, vol. 58, no. 8, pp. 5544–5562, Aug. 2012.

[7] A. Miranian and M. Abdollahzade, "Developing a local least-squares support vector machines-based neuro-fuzzy model for nonlinear and chaotic time series prediction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 2, pp. 207–218, Feb. 2013.

[8] L. Shao, D. Wu, and X. Li, "Learning deep and wide: A spectral method for learning deep networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 12, pp. 2303–2308, Dec. 2014.

[9] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 11, pp. 3212–3232, Nov. 2019.

[10] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[11] M. Hermans and B. Schrauwen, "Training and analysing deep recurrent neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 190–198.

[12] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," in *Proc. IJCAI*, 2016, pp. 2873–2879.

[13] N. Laptev, J. Yosinski, L. E. Li, and S. Smyl, "Time-series extreme event forecasting with neural networks at Uber," in *Proc. Int. Conf. Mach. Learn.*, no. 34, 2017, pp. 1–5.

[14] W. De Mulder, S. Bethard, and M.-F. Moens, "A survey on the application of recurrent neural networks to statistical language modeling," *Comput. Speech Lang.*, vol. 30, no. 1, pp. 61–98, 2015.

[15] A. Zeevi, R. Meir, and R. Adler, "Time series prediction using mixtures of experts," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 9, 1997, pp. 309–318.

[16] L. I. Kuncheva, *Diversity in Classifier Ensembles*. Hoboken, NJ, USA: Wiley, 2014, ch. 8, pp. 247–289.

[17] Y. Ephraim and N. Merhav, "Hidden Markov processes," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1518–1569, Jun. 2002.

[18] P. Nystrup, H. Madsen, and E. Lindström, "Long memory of financial time series and hidden Markov models with time-varying parameters," *J. Forecasting*, vol. 36, no. 8, pp. 989–1002, Dec. 2017.

[19] S. S. Kozat and A. C. Singer, "Universal randomized switching," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1922–1927, Mar. 2010.

[20] J. D. Hamilton, "A new approach to the economic analysis of nonstationary time series and the business cycle," *Econometrica, J. Econ. Soc.*, vol. 57, no. 2, pp. 357–384, 1989.

[21] C.-J. Kim, C. R. Nelson, and R. Startz, "Testing for mean reversion in heteroskedastic data based on Gibbs-sampling-augmented randomization," *J. Empirical Finance*, vol. 5, no. 2, pp. 131–154, Jun. 1998.

[22] A. J. Filardo, "Business-cycle phases and their transitional dynamics," *J. Bus. Econ. Stat.*, vol. 12, no. 3, pp. 299–308, 1994.

[23] M. Wang, Y.-H. Lin, and I. Mikhelson, "Regime-switching factor investing with hidden Markov models," *J. Risk Financial Manage.*, vol. 13, no. 12, p. 311, Dec. 2020.

[24] J. Cheng, J. H. Park, X. Zhao, H. R. Karimi, and J. Cao, "Quantized nonstationary filtering of networked Markov switching RSNSs: A multiple hierarchical structure strategy," *IEEE Trans. Autom. Control*, vol. 65, no. 11, pp. 4816–4823, Nov. 2020.

[25] X. Zhou, J. Cheng, J. Cao, and M. Ragulskis, "Asynchronous dissipative filtering for nonhomogeneous Markov switching neural networks with variable packet dropouts," *Neural Netw.*, vol. 130, pp. 229–237, Oct. 2020.

[26] R. Lu, J. Tao, P. Shi, H. Su, Z.-G. Wu, and Y. Xu, "Dissipativity-based resilient filtering of periodic Markovian jump neural networks with quantized measurements," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 5, pp. 1888–1899, May 2018.

[27] C.-J. Kim and C. Nelson, *State-Space Models With Regime Switching: Classical and Gibbs-Sampling Approaches With Applications*. Cambridge, MA, USA: MIT Press, 2017.

[28] S. Hochreiter and J. J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 80–1735, 1997.

[29] K. Cho *et al.*, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1724–1734.

[30] J. L. Elman, "Finding structure in time," *Cognit. Sci.*, vol. 14, no. 2, pp. 179–211, Mar. 1990.

[31] S. J. Taylor and B. Letham, "Forecasting at scale," *Amer. Statistician*, vol. 72, no. 1, pp. 37–45, 2018.

[32] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting," in *Proc. 8th Int. Conf. Learn. Represent.*, 2020, pp. 1–31. [Online]. Available: https://openreview.net/pdf?id=r1ecqn4YwB

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

[34] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1998.

[35] T. Ergen and S. S. Kozat, "Unsupervised anomaly detection with LSTM neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 8, pp. 3127–3141, Aug. 2010.

[36] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "DeepAR: Probabilistic forecasting with autoregressive recurrent networks," *Int. J. Forecasting*, vol. 36, no. 3, pp. 1181–1191, Jul. 2020.

[37] S. Smyl, "A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting," *Int. J. Forecasting*, vol. 36, no. 1, pp. 75–85, Jan. 2020.

[38] R. Ebrahimpour, H. Nikoo, S. Masoudnia, M. R. Yousefi, and M. S. Ghaemi, "Mixture of MLP-experts for trend forecasting of time series: A case study of the Tehran stock exchange," *Int. J. Forecasting*, vol. 27, no. 3, pp. 804–816, Jul. 2011.

[39] SAS. *SAS Documentation*. Accessed: Jan. 20, 2021. [Online]. Available: https://support.sas.com/en/documentation.html

[40] S. Seabold and J. Perktold, "Statsmodels: Econometric and statistical modeling with Python," in *Proc. 9th Python Sci. Conf.*, 2010, p. 61.

[41] Y. Cao, Y. Li, S. Coleman, A. Belatreche, and T. M. McGinnity, "Adaptive hidden Markov model with anomaly states for price manipulation detection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 2, pp. 318–330, Feb. 2015.

[42] S. Kotz, T. Kozubowski, and K. Podgorski, *The Laplace Distribution and Generalizations*. Boston, MA, USA: Springer, 2001.

[43] R. J. Williams and D. Zipser, *Gradient-Based Learning Algorithms for Recurrent Networks and Their Computational Complexity*. Mahwah, NJ, USA: Lawrence Erlbaum Associates, 1995, pp. 433–486.

[44] *Historical Data Feed*. Accessed: Jan. 24, 2020. [Online]. Available: https://www.dukascopy.com/swiss/english/marketwatch/historical/

[45] *M5 Forecasting—Accuracy*. Accessed: Dec. 12, 2020. [Online]. Available: https://www.kaggle.com/c/m5-forecasting-accuracy/data

[46] *UCI Machine Learning Repository: Electricity Load Diagrams Data Set*. Accessed: Dec. 15, 2020. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014

[47] *UCI Machine Learning Repository: PEMS-SF Data Set*. Accessed: Dec. 15, 2020. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/PEMS-SF

Fatih Ilhan received the B.S. degree in electrical and electronics engineering from Bilkent University, Ankara, Turkey, in 2019, where he is currently pursuing the M.S. degree with the Department of Electrical and Electronics Engineering.

His research interests include machine learning, convex/nonconvex optimization, online learning, and their applications on time-series prediction, spatio-temporal forecasting, and anomaly detection.

Oguzhan Karaahmetoglu received the B.S. degree in electrical and electronics engineering from Middle East Technical University, Ankara, Turkey, in 2019. He is currently pursuing the M.S. degree in the Department of Electrical and Electronics Engineering, Bilkent University, Ankara, Turkey.

His current research interests include spatio-temporal prediction, online learning, point process modeling, adaptive filtering, and big data.

Ismail Balaban received the B.S. degree in computer engineering from Middle East Technical University, Ankara, Turkey, in 2018.

He is working as a Machine Learning Engineer at DataBoss Security & Analytics, Ankara. His current research interests are machine learning and deep learning. He focuses on forecasting and natural language processing.

Suleyman Serdar Kozat (Senior Member, IEEE) received the B.S. (Hons.) degree from Bilkent University, Ankara, Turkey, in 1998, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Illinois at Urbana-Champaign, Urbana, IL, USA, in 2001 and 2004, respectively.

He joined the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA, as a Research Staff Member and later became a Project Leader with the Pervasive Speech Technologies Group, where he focused on problems related to statistical signal processing and machine learning. He was a Research Associate with the Cryptography and Anti-Piracy Group, Microsoft Research, Redmond, WA, USA. He is currently a Professor with the Department of Electrical and Electronics Engineering, Bilkent University. He has authored or coauthored more than 120 articles in refereed high impact journals and conference proceedings and holds several patent inventions (currently used in several different Microsoft and IBM products, such as MSN and ViaVoice). He holds several patent inventions due to his research accomplishments with the IBM Thomas J. Watson Research Center and Microsoft Research. His current research interests include cybersecurity, anomaly detection, big data, data intelligence, adaptive filtering, and machine learning algorithms for signal processing.

Dr. Kozat was a recipient of many international and national awards. He is the elected President of the IEEE Signal Processing Society, Turkey Chapter.