Journal of Machine Learning Research 26 (2025) 1-30 Submitted 8/24; Revised 10/25; Published 10/25

Efficient Online Prediction for High-Dimensional Time Series via Joint Tensor Tucker Decomposition

Zhenting Luan ZHENTING.LUAN@POLYU.EDU.HK
Defeng Sun DEFENG.SUN@POLYU.EDU.HK
Department of Applied Mathematics
The Hong Kong Polytechnic University, Hong Kong, China

Haoning Wang WHN22@MAILS.TSINGHUA.EDU.CN
Liping Zhang* LIPINGZHANG@TSINGHUA.EDU.CN
Department of Mathematical Sciences
Tsinghua University, Beijing, China

Editor: Krishnakumar Balasubramanian

Abstract

Real-time prediction plays a vital role in various control systems, such as traffic congestion control and wireless channel resource allocation. In these scenarios, the predictor usually needs to track the evolution of the latent statistical patterns in the modern high-dimensional streaming time series continuously and quickly, which presents new challenges for traditional prediction methods. This paper is the first to propose a novel online algorithm (TOPA) based on tensor factorization to predict streaming tensor time series. The proposed algorithm TOPA updates the predictor in a low-complexity online manner to adapt to the time-evolving data. Additionally, an automatically adaptive version of the algorithm (TOPA-AAW) is presented to mitigate the negative impact of stale data. Simulation results demonstrate that our proposed methods achieve prediction accuracy similar to that of conventional offline tensor prediction methods, while being much faster than them during long-term online prediction. Therefore, TOPA-AAW is an effective and efficient solution method for online prediction of streaming tensor time series.

Keywords: high-dimensional time series, streaming data, online prediction, Tucker decomposition, alternating minimization method, autoregression

1. Introduction

Time series is a sequence of discrete data points observed over time, which is generated naturally in various areas such as economics, climate, and traffic (Kwon et al., 2021; Shih et al., 2019). The prediction of time series data is crucial in numerous real-world scenarios such as stock market analysis (Sharma et al., 2017) and traffic flow (Lv et al., 2014). For example, traffic prediction is the core component of intelligent transportation systems, including route planning, traffic control and car dispatching (Li and Shahabi, 2018). Conventional methods for time series prediction usually depend on autoregression (AR) models (Lewis and Reinsel, 1985), e.g., autoregressive integrated moving average (ARIMA) model

∗. Liping Zhang is the corresponding author (lipingzhang@tsinghua.edu.cn).

(Hamilton, 2020) and vector autoregressive (VAR) model (Zivot and Wang, 2006), to handle scalar or vector batch data.

However, with the emergence of large-scale time series data generated in modern production activities, classical prediction methods face challenges posed by modern sensing technologies. The modern time series data are usually super large-scale and detected from numerous sensors. For instance, the freeway traffic Performance Measurement System of the California Department of Transportation is a sensor network of more than 26000 individual-lane inductive loops installed throughout California freeways, with more than 35000 detectors that send traffic measures to a road control unit per 30 seconds (Pascale and Nicoli, 2011; Chen and Sun, 2021). Classical methods are not efficient and reliable for handling such large-scale time series. Moreover, modern time series data are often multi-dimensional with different characteristics shown in different dimensions, such as the channel state information in time, spatial, and frequency domains detected by periodic sounding reference signals (Luo et al., 2018).

With the rapid development of sensing technologies, *streaming time series* data are collected continuously from various types of sensors, such as traffic flow measurement in the traffic congestion control system (Maarala et al., 2015) and the periodic channel state information (CSI) in wireless channel resource allocation (Letaief and Zhang, 2006). Accurate real-time prediction is beneficial for the system to make timely decisions in such scenarios. However, conventional prediction methods must re-exploit the entire data set, including the newly observed data, to rebuild the prediction models in each process of predicting the upcoming data, which is unsuitable for high-frequency streaming time series. For example, in mobile scenarios, wireless channels change rapidly. So we're going to see a rapid mismatch of the external channels that rely on pilot estimation. As a result, the performance of interference suppression algorithms is severely degraded. It is crucial to use the prediction algorithm to obtain the channel of the channel at non-pilot time and improve the difference between the estimated value and the actual channel, improve the system performance (Kim et al., 2020). Additionally, due to the evolution of statistical patterns in streaming time series data, predicting the upcoming data over an extended period is improper without prediction model updating. Therefore, online prediction is essential to help forecast upcoming data and make immediate decisions continuously and rapidly.

Recent advances in tensor research have led to the development of prediction models (Ahn et al., 2021; Wang et al., 2024) for tensor time series (TTS) data with high dimensions based on tensor factorization such as Tucker decomposition and CP decomposition (De Lathauwer et al., 2000; Kolda and Bader, 2009; Jing et al., 2018). These methods explore the joint low-rank structure of all observed data, where the statistical patterns or the periodicity of data are used for generating predictors. The tensor factorization strategy (Yu et al., 2016) is a generalization of matrix factorization, which has been shown to be very efficient in large-scale multi-variable time series. Jing et al. (2018) developed a multi-linear prediction model, MOAR, based on Tucker decomposition, which projected the original tensors into several subspaces and built temporal connections among the core tensors by the traditional autoregressive model. The constrained version of MOAR, called MCAR, regularizes the residual of the joint tensor decomposition to find a more stable solution. The projection matrices, representing the feature subspaces in the Tucker decomposition, capture the low-rank structure of tensors, which can be further assumed on the Stiefel

manifold (Jing et al., 2018) or following some semi-parametric model (Chen et al., 2024) if the mode-wise structure can be partially explained by some relevant covariate. Some statistical methods are also presented for the projection matrices inference under Gaussian or sub-Gaussian noises (Xu et al., 2025; Xia et al., 2022). Shi et al. (2020) proposed a prediction algorithm that combined tensor decomposition and ARIMA model for the block-Hankelized TTS data, which exploited the low-rank structure of TTS and captured the intrinsic correlations among it. Both of these approaches demonstrate strong performance for single-time predictions by constructing predictors from scratch with numerous iterative iterations. However, they are not suitable for direct application in streaming TTS online prediction due to the necessity of repeating the process of constructing predictors. Tan et al. (2016) presented a short-term traffic flow prediction approach to predicting streaming traffic data based on dynamic tensor completion (DTC). This approach regarded the data to be predicted as some missing data in the corner of a multi-dimensional tensor. It used DTC to complete the tensor to obtain the missing data for prediction. However, this process depends on the multi-mode periodicity of traffic flow data, and hence is unsuitable for generic TTS data without periodicity.

From the above mentioned, using traditional forecasting methods to deal with such scenarios often faces the dual challenges of computational complexity and multi-dimensional feature mining. A natural question is how to design an accurate and fast online prediction algorithm for such high-dimensional tensor time series.

In this paper, we design an algorithm — Tucker-decomposition based Online Prediction Algorithm (TOPA) — for predicting generic streaming TTS data in an online manner, which allows us to quickly update the predictors from the previous model and the latest observations rather than rebuild it from scratch. At each sampling time during the streaming observation, we employ the Tucker decomposition for dimension reduction and joint feature subspace extraction of the streaming TTS, and find the autoregression pattern in the dimension-reduced TTS (the core tensor series). Then, the prediction results are obtained from the predicted core tensor together with the inverse Tucker decomposition. We design a regularized optimization problem with a least-squared form to search for the proper Tucker decomposition and regression model. In order to quickly solve the optimization and update the previous predictor, we introduce an online updating scheme for the optimization, of which the initialization is exactly the solution to the previous optimization problem formulated at the last time point. Moreover, for streaming data in which the statistical pattern evolves over a long period, we propose an automatically-adaptive-weight (AAW) strategy for processing past data during online prediction, which can reduce the cumulative error of the predictor from stale data and automatically decrease the weights of very noisy data. The improved version of TOPA is called TOPA-AAW.

In summary, the main contribution of this paper is to propose an online prediction algorithm for streaming TTS based on tensor factorization, which includes the following specific aspects.

- The gradually varied joint tensor factorization structure of streaming TTS is updated by the proposed online updating strategy for tracking the changing underlying dynamic statistical characteristics. Compared with conventional joint tensor factorization methods, this strategy takes much less time owing to only a few alternative iterations with an inherited initialization.

- We design an online updating algorithm TOPA with inherited solutions to quickly update the joint tensor factorization structure once observing new data. The prediction accuracy of this algorithm is close to offline methods (e.g. Jing et al., 2018; Shi et al., 2020), while the time complexity is significantly lower. The performance of TOPA is validated in numerical simulations, including synthetic and real-world data sets. We also analyze the convergence of our proposed algorithms.
- To reduce the interference from stale data to the latest statistical characteristics modeling, we propose an automatically adaptive regularization strategy TOPA-AAW for the introduced least-squared optimization problem. The TOPA-AAW can gradually decrease the weights of historical and abnormal data.

The paper is organized as follows. In Section 2, we define the online prediction problem, give a real-world example for the online prediction problem for tensor time series,and review the joint tensor factorization strategy for TTS. In Section 3, we propose the TOPA and introduce the online manner for predicting streaming TTS data. In order to address the problem of data staleness, we add automatically adaptive weights for regularizing the joint tensor factorization and propose TOPA-AAW in Section 4. In Section 5, we evaluate the performance of our proposed algorithms in different scenarios and compare it with some tensor offline prediction methods and neural-network-based methods. Finally, Section 6 concludes the paper and suggests possible future directions.

2. Preliminaries

In this section, we first give notation and tensor operators used in this paper. We next to define the online prediction problem and introduce the joint tensor factorization strategy for TTS.

2.1 Notation and Tensor Operators

We employ bold italic lower-case letters, bold italic capital letters, and bold calligraphic letters to denote vectors, matrices, and tensors, respectively (e.g., $\boldsymbol{x}, \boldsymbol{X}$ and $\boldsymbol{\mathcal{X}}$). For a positive integer n, denote the set $[n] = \{1, 2, \ldots, n\}$. We use uppercase letters with script typestyle to denote the set of data (e.g., $\mathscr{G}$). Furthermore, we represent the time series with given length T by uppercase letters with script typestyle equipped with a subscript $[T]$ (e.g., $\mathscr{G}_{[T]}$). Denote $\boldsymbol{I}_k$ as the $k \times k$ identity matrix. We use $(\cdot)^H$ to denote the conjugate transpose of a complex matrix. The set of unitary matrices with given scale $I \times R (I > R)$ is denoted as

$$\mathbb{U}^{I\times R} = \{\boldsymbol{U} \in \mathbb{C}^{I\times R} | \boldsymbol{U}^H\boldsymbol{U} = \boldsymbol{I}_R\}.$$

Each dimension of a tensor is called a *mode*. The *m-mode product* $\times_m$ between a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{C}^{I_1\times\cdots\times I_M}$ and a matrix $\boldsymbol{U}_m \in \mathbb{C}^{J_m\times I_m}$ is defined as

$$(\boldsymbol{\mathcal{X}} \times_m \boldsymbol{U}_m)_{i_1\ldots i_{m-1}ji_{m+1}\ldots i_M} = \sum_{i_m=1}^{I_m} \boldsymbol{\mathcal{X}}_{i_1\ldots i_m\ldots i_M} \boldsymbol{U}_{ji_m}$$

for $\forall j \in [J_m], \forall i_l \in [I_l], \forall l \in [M]\backslash\{m\}$. The *m-mode unfolding matrix* of a tensor is recorded as $\boldsymbol{\mathcal{X}}_{(k)} \in \mathbb{C}^{I_m\times(\prod_{l\neq m} I_l)}$. Denote $R(\boldsymbol{\mathcal{X}}) = (rank(\boldsymbol{\mathcal{X}}_{(1)}), \cdots, rank(\boldsymbol{\mathcal{X}}_{(M)}))$ as the *Tucker-*

rank of an Mth-order tensor $\boldsymbol{\mathcal{X}}$. The inner product between two tensors is defined as $\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \rangle = \sum_{i_1,\cdots,i_M} \boldsymbol{\mathcal{X}}_{i_1\ldots i_M} \boldsymbol{\mathcal{Y}}_{i_1\ldots i_M}$. The Frobenius norm of a tensor $\boldsymbol{\mathcal{X}}$ is defined as $\|\boldsymbol{\mathcal{X}}\|_F = \sqrt{\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{X}} \rangle}$. Moreover, the Frobenius norm of a tuple of tensor $\mathcal{X} = (\boldsymbol{\mathcal{X}}_1, \ldots, \boldsymbol{\mathcal{X}}_n)$ is defined as $\|\mathcal{X}\|_F = \sqrt{\sum_{i=1}^n \|\boldsymbol{\mathcal{X}}_i\|_F^2}$. The tensor product between two tensors is denoted as $(\boldsymbol{\mathcal{X}}_{i_1,\cdots,i_M}) \otimes (\boldsymbol{\mathcal{Z}}_{j_1,\cdots,j_N}) = (\boldsymbol{\mathcal{X}}_{i_1,\cdots,i_M} \boldsymbol{\mathcal{Z}}_{j_1,\cdots,j_N})$.

2.2 Problem Formulation

Denote $\boldsymbol{\mathcal{X}}_t \in \mathbb{C}^{I_1 \times \cdots \times I_M}$ as the observation at sampling time t and $\mathscr{X}_{[T]} := \{\boldsymbol{\mathcal{X}}_1, \cdots, \boldsymbol{\mathcal{X}}_T\}$ as the TTS until sampling time T. Given $\mathscr{X}_{[T]}$, the prediction of $\boldsymbol{\mathcal{X}}_{T+1}$ is to find a predictor $h_T(\cdot)$ that minimizes the mean squared error (MSE) as

$$\min \quad \|\boldsymbol{\mathcal{X}}_{T+1} - h_T(\mathscr{X}_{[T]})\|_F, \tag{1}$$

where $\hat{\boldsymbol{\mathcal{X}}}_{T+1} := h_T(\mathscr{X}_{[T]})$ is the prediction value of $\boldsymbol{\mathcal{X}}_{T+1}$.

In this paper, we consider the online prediction problem (1) for streaming TTS, in which the prediction function $h_T(\cdot)$ changes as streaming observations $\boldsymbol{\mathcal{X}}_T$ arrive sequentially. Therefore, the previous predictor should be updated with the latest observation for the next prediction at each sampling time. This dynamic process is referred to as *online prediction*.

For high-dimensional streaming TTS data, the curse of dimensionality makes data processing challenging. To mitigate this challenge and explore the multi-aspect temporal continuity of TTS, we first introduce a real-world example for TTS in the next subsection, then review the *joint tensor factorization* strategy which motivates us to develop low-complexity algorithms for such TTS online prediction.

2.3 Real-World Example for TTS: Wireless Communication Channel Prediction

In wireless communication systems, channel state information (CSI) between user equipment (UE) and base station constantly changes with the movement of terminal devices, environmental changes, channel interference, and multipath propagation. Since the CSI estimated by the guide symbols is relatively outdated due to feedback delay, channel prediction can help optimize various wireless communication system modules, such as channel resource preallocation, adaptive coding and decoding, and beamforming (Liu et al., 2013). By predicting changes in CSI, scheduling strategies can be adjusted accordingly to ensure reliable and efficient data transmission in wireless communications.

As a real-world instance for TTS, we consider a 3-dimensional multi-input multi-output (MIMO) wireless channel model, a simplified version of the model introduced by Ademaj et al. (2016) without polarization and antenna pattern. Specifically, the space-domain CSI matrix at t-th time-slot on the k-th frequency carrier between receiver and transmitter is formulated as

$$\boldsymbol{H}[k,t] = \sum_{n=1}^{N} \sqrt{\frac{P_n}{M_n}} e^{-j2\pi(f_c + k f_{sc})\tau_n} \sum_{m=1}^{M_n} e^{j\frac{2\pi}{\lambda_c}|\mathbf{r}_{n,m}^T \mathbf{v}| t_s t} \left(\mathbf{e}_{rx,n,m} \mathbf{e}_{tx,n,m}^H\right) \in \mathbb{C}^{n_{\mathrm{rx}} \times n_{\mathrm{tx}}}, \tag{2}$$

where

- $n_{\text{rx}}, n_{\text{tx}}$ denote the number of antennas of receiver and transmitter, repsectively;
- N is the number of transmittion path cluster, M_n is the number of sub-path in n-th cluster, P_n/M_n is the average power of each sub-path;
- $e^{-j2\pi(f_c+kf_{sc})\tau_n}$ represents the phase-shift of the n-th path cluster with time-delay τ_n on the k-th frequency carrier, while f_c and f_{sc} are central frequency and bandwidth, repsectively;
- $e^{j\frac{2\pi}{\lambda_c}|\mathbf{r}_{n,m}^T\mathbf{v}|t_s t}$ is the Doppler Shift during time $t_s t$ with relative velocity $\mathbf{v}$ and direction vector $\mathbf{r}_{n,m}$, here λ_c is the wavelength, and t_s is the duration of single time-slot;
- $\mathbf{e}_{rx,n,m} \in \mathbb{C}^{n_{\text{rx}}\times 1}$, $\mathbf{e}_{tx,n,m} \in \mathbb{C}^{n_{\text{tx}}\times 1}$ represent the steering vector of sub-path (n,m) for the antenna arrays of receiver and transmitter, respectively, which are related to the wavelength, antenna distance in the arraies, angle of departure and angle of arrival of the antenna arrays.

If we ignore the slight influence of UE movement on the physical parameters τ_n, $\left|\mathbf{r}_{n,m}^T\mathbf{v}\right|$, $\mathbf{e}_{rx,n,m}$ and $\mathbf{e}_{tx,n,m}$, the CSI both in frequency-domain and space-domain changing with time-slot t constructs a 3-dimensional TTS:

$$\begin{aligned}\boldsymbol{\mathcal{H}}_t &= \sum_{n=1}^{N}\sum_{m=1}^{M_n}\sqrt{\frac{P_n}{M_n}}e^{j\frac{2\pi}{\lambda_c}|\mathbf{r}_{n,m}^T\mathbf{v}|t_s t}\mathbf{f}^{(n)}\otimes\mathbf{e}_{rx,n,m}\otimes\overline{\mathbf{e}}_{tx,n,m} \\ &= \boldsymbol{\mathcal{G}}_t\times_1\mathbf{F}\times_2\mathbf{E}_{rx}\times_3\overline{\mathbf{E}}_{tx}\in\mathbb{C}^{K\times n_{rx}\times n_{tx}}, t=1,2,3,\ldots,\end{aligned} \tag{3}$$

where $\mathbf{f}^{(n)} = \left(e^{-j2\pi(f_c+kf_{sc})\tau_n}\right)_k$, $\mathbf{F} = \left[\mathbf{f}^{(1)},\cdots,\mathbf{f}^{(N)}\right]$, $\mathbf{E}_{rx} = \left[\mathbf{e}_{rx,1,1},\cdots,\mathbf{e}_{rx,N,M_N}\right]$, $\mathbf{E}_{tx} = \left[\mathbf{e}_{tx,1,1},\cdots,\mathbf{e}_{tx,N,M_N}\right]$, and $(\boldsymbol{\mathcal{G}}_t)_{n,(n,m),(n,m)} = \sqrt{\frac{P_n}{M_n}}e^{j\frac{2\pi}{\lambda_c}|\mathbf{r}_{n,m}^T\mathbf{v}|t_s t}$ being all non-zeros entries of $\boldsymbol{\mathcal{G}}_t \in \mathbb{C}^{N\times\sum_{n=1}^{N}M_n\times\sum_{n=1}^{N}M_n}$.

When the propagation path clusters and sub-paths are few in number, such as in the mmWave communication, $\boldsymbol{\mathcal{G}}_t$ in (3) can be seemed as a compression of the original CSI tensor $\boldsymbol{\mathcal{H}}_t$. The matrices $\mathbf{F}, \mathbf{E}_{rx}, \mathbf{E}_{tx}$ represent different physical factors that influence CSI, and continue to change slightly over time due to the movement of the UE. And $\{\boldsymbol{\mathcal{G}}_t\}_t$ is continuously changing **over time**. Therefore, to predict the CSI, it is better to perform regression directly on $\{\boldsymbol{\mathcal{G}}_t\}_t$ and continuously update the matrices across all dimensions in an online manner, then compute the prediction result by multiplying all these factors. Obviously, the stability of the original CSI TTS relies on the UE movement. In the following subsection, we review a tensor decomposition strategy to handle (3) in a more generalized perspective.

2.4 Joint Tensor Factorization Strategy

Tensor factorization is a powerful approach to compressing high-dimensional data (Kolda and Bader, 2009). In this subsection, we provide a brief overview of a specific factorization tool, the (joint) Tucker decomposition, which has been shown to be effective in predicting nonstreaming TTS (Jing et al., 2018; Wang et al., 2024).

Tucker's decomposition factorizes a higher-order tensor into a core tensor multiplied by a set of projection matrices along each mode, such as the CSI tensor in (3). The core

tensor captures the essential information of the original tensor, while the projection matrices represent the feature subspaces. For the TTS, using the temporal continuity of the tensor and matrix factors, such as shown in (3), we employ the joint Tucker decomposition to compress the original TTS into a lower-dimensional representation while preserving the essential features. Specifically, the joint Tucker decomposition finds the $(R_1, \cdots, R_M)$-rank approximation of a given TTS with joint projection matrices, which is formulated as:

$$\boldsymbol{\mathcal{X}}_t \approx \boldsymbol{\mathcal{G}}_t^{(T)} \prod_{m=1}^{M} \times_m \boldsymbol{U}_m^{(T)}, \quad t \in [T], \tag{4}$$

where $\boldsymbol{\mathcal{G}}_t^{(T)} \in \mathbb{C}^{R_1 \times \ldots \times R_M}$ is the *core tensor* corresponding to $\boldsymbol{\mathcal{X}}_t$, and $\boldsymbol{U}_m^{(T)} \in \mathbb{U}^{I_m \times R_m}$ is considered the m-th joint feature subspace for all tensors in $\mathscr{X}_{[T]}$. Note that for a streaming TTS, $\boldsymbol{U}_m^{(T)}$ must be continuously updated over time T to accommodate the latest observation $\boldsymbol{\mathcal{X}}_T$.

As discussed in Jing et al. (2018), the core tensors obtained by joint Tucker decomposition can be viewed as a compact representation of the intrinsic interactions among multiple subspaces of features of TTS. This representation is more suitable for modeling temporal continuity than the original data. Moreover, the size of the core tensors is much smaller than the original TTS, which makes it computationally more efficient to apply AR models to the core tensor series. For example, we can use the AR(p) model to illustrate the temporal correlations among the core tensor series, which is formulated as

$$\boldsymbol{\mathcal{G}}_t^{(T)} = \sum_{i=1}^{p} \alpha_i^{(T)} \boldsymbol{\mathcal{G}}_{t-i}^{(T)} + \boldsymbol{\mathcal{E}}_t^{(T)}, \quad p+1 \le t \le T, \tag{5}$$

where $\{\alpha_i^{(T)}\}$ are the AR parameters and $\boldsymbol{\mathcal{E}}_t^{(T)}$ is the Gaussian white noise in $\boldsymbol{\mathcal{X}}_t$. With (5), the core tensor $\hat{\boldsymbol{\mathcal{G}}}_{T+1}^{(T)}$ of next observation data is predicted by

$$\hat{\boldsymbol{\mathcal{G}}}_{T+1}^{(T)} = \sum_{i=1}^{p} \alpha_i^{(T)} \boldsymbol{\mathcal{G}}_{T+1-i}^{(T)}. \tag{6}$$

Thus, the prediction of upcoming data is given by extending the temporal continuity of multi-aspect feature subspace to the next time point:

$$\hat{\boldsymbol{\mathcal{X}}}_{T+1} = \hat{\boldsymbol{\mathcal{G}}}_{T+1}^{(T)} \prod_{m=1}^{M} \times_m \boldsymbol{U}_m^{(T)}. \tag{7}$$

When dealing with streaming TTS, traditional *offline prediction* methods, such as MCAR (Jing et al., 2018) and BHT-ARIMA (Shi et al., 2020), suffer from high computational complexity as they require finding a new predictor for (1) after each observation without inheriting any information from previous predictors. This results in a time-consuming process of repeatedly solving the model during the streaming observation. To address this issue, we propose an *online prediction* method, TOPA, for (1) with the joint tensor-factorization strategy. With the latest observation, the proposed TOPA can quickly build a new predictor (7) by updating the previous compression factors of (4) in an online manner, which reduces the prediction time costs significantly.

3. TOPA for Streaming TTS

In this section, we introduce a novel two-stage online prediction algorithm for (1) called *Tucker-decomposition-based Online Prediction Algorithm* (TOPA). The first stage aims to find an initial predictor using starting TTS. This stage can be seen as a generalization of MCAR (Jing et al., 2018) with a simplified optimization problem. The second stage is the main focus of TOPA, which continuously updates the predictor and predicts the upcoming data as the streaming observation arrives. We online update the predictor by solving an incremental optimization problem with the previous predictor as initialization. At the end of this section, we analyze the convergence and computational complexity of TOPA.

3.1 Stage I: Initial Predictor with Starting TTS

In order to find a suitable initial predictor before receiving streaming data, the observer needs to observe a starting TTS as the initial input for TOPA. It is assumed that this starting TTS comprises T_0 samples and the streaming observation starts from time T_0+1.

By treating the starting TTS as a non-streaming TTS, we use the joint tensor factorization strategy introduced in Section 2.4 to create the initial predictor. The joint Tucker decomposition of starting TTS is formulated as (4) with $T = T_0$, which estimates the joint feature subspaces of TTS as $\mathscr{U}^{(T_0)} \triangleq \{\boldsymbol{U}_m^{(T_0)}\}_{m\in[M]}$ and the core tensor series as $\mathscr{G}_{[T_0]}^{(T_0)} \triangleq \{\boldsymbol{\mathcal{G}}_t^{(T_0)}\}_{t\in[T_0]}$. To be more generalized, we denote $f_{\mathscr{P}^{(T_0)}}(\cdot)$ as any desired AR-type model for $\mathscr{G}_{[T_0]}^{(T_0)}$, e.g., VAR model and ARIMA model, which is formulated as

$$\boldsymbol{\mathcal{G}}_t^{(T_0)} \approx f_{\mathscr{P}^{(T_0)}}(\mathscr{G}_{[t-1]}^{(T_0)}), \quad t \in [T_0], \tag{8}$$

where $\mathscr{P}^{(T_0)}$ represents all regression parameters in f and $\mathscr{G}_{[t]}^{(T_0)}$ are the first t core tensors of the starting TTS. For example, $\mathscr{P}^{(T)} = \{\alpha_i^{(T)}\}_{i\in[p]}$ in (5).

Remark 1. Notice that the order of AR model, denoted $p(< T_0)$, defines the number of past data that have direct temporal correlations with the current data, i.e.,

$$f_{\mathscr{P}^{(T_0)}}(\mathscr{G}_{[t-1]}^{(T_0)}) = f_{\mathscr{P}^{(T_0)}}(\boldsymbol{\mathcal{G}}_{t-p}^{(T_0)}, \cdots, \boldsymbol{\mathcal{G}}_{t-1}^{(T_0)}).$$

For convenience, we still use $f_{\mathscr{P}^{(T_0)}}(\mathscr{X}_{[t]})$ to represent the model with a little abuse of notation when there is no ambiguity about the order.

With tensor factorization and regression, TOPA predicts $\boldsymbol{\mathcal{X}}_{T_0+1}$ by inversely projecting the predicted core tensor with the joint feature subspaces:

$$\begin{aligned}\hat{\boldsymbol{\mathcal{X}}}_{T_0+1} &= \hat{\boldsymbol{\mathcal{G}}}_{T_0+1} \prod_{m=1}^{M} \times_m \boldsymbol{U}_m^{(T_0)} = f_{\mathscr{P}^{(T_0)}}\left(\mathscr{G}_{[T_0]}^{(T_0)}\right) \prod_{m=1}^{M} \times_m \boldsymbol{U}_m^{(T_0)} \\ &=: h_{T_0}(\mathscr{X}_{[T_0]}),\end{aligned} \tag{9}$$

where $h_{T_o}(\cdot)$ depends on $\mathscr{U}^{(T_0)}, \mathscr{G}_{[T_0]}^{(T_0)}$ and $\mathscr{P}^{(T_0)}$. Define

$$F_{T_0}\left(\mathscr{G}_{[T_0]}^{(T_0)}, \mathscr{P}^{(T_0)}, \mathscr{U}^{(T_0)}\right) = \sum_{t=p+1}^{T_0} \|\boldsymbol{\mathcal{G}}_t^{(T_0)} - f_{\mathscr{P}^{(T_0)}}(\mathscr{G}_{[t-1]}^{(T_0)})\|_F^2 + \varphi \sum_{t=1}^{T_0} \|\boldsymbol{\mathcal{G}}_t^{(T_0)} - \boldsymbol{\mathcal{X}}_t \prod_{m=1}^{M} \times_m (\boldsymbol{U}_m^{(T_0)})^H\|_F^2,$$

where the first term is used to minimize the error of core tensor series regression, and the second term is used to regularize the joint Tucker decomposition of the starting TTS with a residual minimization formulation and regularization parameter $\varphi > 0$. By (9), we can reformulate (1) as the following optimization problem:

$$\begin{aligned} \min_{\mathscr{G}_{[T_0]}^{(T_0)}, \mathscr{P}^{(T_0)}, \mathscr{U}^{(T_0)}} & F_{T_0}\left(\mathscr{G}_{[T_0]}^{(T_0)}, \mathscr{P}^{(T_0)}, \mathscr{U}^{(T_0)}\right) \\ \text{s.t.} \quad & \boldsymbol{U}_m^{(T_0)} \in \mathbb{U}^{I_m \times R_M}, \quad \forall m \in [M]. \end{aligned} \tag{10}$$

According to the multi-variable and least-squared structure of problem (10), we propose a proximal alternating minimization algorithm to solve it. In this algorithm, we solve a series of subproblems with a proximal regularized term (Kaplan and Tichatschke, 1998) of (10) in an alternating manner, with each subproblem fixing all variables except the one being updated. That is to say, it alternatively updates only one of $\mathscr{P}^{(T_0)}$, $\mathscr{U}^{(T_0)}$, and $\mathscr{G}_{[T_0]}^{(T_0)}$ at a time. Fortunately, each subproblem with respect to $\mathscr{P}^{(T_0)}$, $\mathscr{U}^{(T_0)}$, and $\mathscr{G}_{[T_0]}^{(T_0)}$ has closed-form solution. We give the updated derivation process in details.

Update regression parameters: For AR model, the subproblem of updating $\mathscr{P}^{(T_0)} = \{\boldsymbol{\alpha}^{(T_0)} = [\alpha_1^{(T_0)}, \cdots, \alpha_p^{(T_0)}]\}$ with proximal term and step size λ is formulated as

$$\begin{aligned} \boldsymbol{\alpha}_{k+1}^{(T_0)} &= \arg\min_{\boldsymbol{\alpha}} F_{T_0}\left(\mathscr{G}_{[T_0],k}^{(T_0)}, \{\boldsymbol{\alpha}\}, \mathscr{U}_k^{(T_0)}\right) + \frac{\lambda}{2}\|\boldsymbol{\alpha} - \boldsymbol{\alpha}_k^{(T_0)}\|_F^2 \\ &= \arg\min_{\boldsymbol{\alpha}} \sum_{t=p+1}^{T_0} \|\boldsymbol{\mathcal{G}}_{t,k}^{(T_0)} - \sum_{i=1}^{p} \alpha_i \boldsymbol{\mathcal{G}}_{t-i,k}^{(T_0)}\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{\alpha} - \boldsymbol{\alpha}_k^{(T_0)}\|_F^2. \end{aligned} \tag{11}$$

The closed-form solution to (11) is

$$\boldsymbol{\alpha}_{k+1}^{(T_0)} = \left(\boldsymbol{R} + \frac{\lambda}{2}\boldsymbol{I}_p\right)^{-1}\left(\boldsymbol{r} + \frac{\lambda}{2}\boldsymbol{\alpha}_k^{(T_0)}\right), \tag{12}$$

where

$$\boldsymbol{R}_{i,j} = \sum_{t=p+1}^{T_0} \left\langle \boldsymbol{\mathcal{G}}_{t-i,k}^{(T_0)}, \boldsymbol{\mathcal{G}}_{t-j,k}^{(T_0)} \right\rangle, \quad 1 \le i, j \le p, \tag{13}$$

and

$$\boldsymbol{r}_i = \sum_{t=p+1}^{T_0} \left\langle \boldsymbol{\mathcal{G}}_{t-i,k}^{(T_0)}, \boldsymbol{\mathcal{G}}_{t,k}^{(T_0)} \right\rangle, \quad 1 \le i \le p, \tag{14}$$

Remark 2. When T_0 is large enough, the entries of $\boldsymbol{R}$ and $\boldsymbol{r}$ can be seemed as approximations of the autocorrelation function of $\mathscr{G}_{[T_0],k}^{(T_0)}$:

$$r(j-i) = \mathbb{E}\left\langle \boldsymbol{\mathcal{G}}_t, \boldsymbol{\mathcal{G}}_{t+j-i} \right\rangle.$$

Then, we have

$$\boldsymbol{R}_{i,j} \approx r(i-j), \quad \boldsymbol{r}_i \approx r(I).$$

With these approximations, when $\lambda = 0$, (12) is the solution to the known Yule-Walker equation (Chen et al., 2011).

Update joint projection matrices $\mathscr{U}^{(T+1)}$: For $m = 1$ to M, the subproblem of updating $\boldsymbol{U}_m^{(T_0)}$ with proximal term and step size λ is formulated as

$$\begin{aligned}
\boldsymbol{U}_{m,k+1}^{(T_0)} &= \underset{\boldsymbol{U}\in\mathbb{U}^{I_m\times R_m}}{\arg\min}\ F_{T_0}\Big(\mathscr{G}_{[T_0],k}^{(T_0)}, \mathscr{P}_{k+1}^{(T_0)}, \{\boldsymbol{U}_{i,k+1}^{(T_0)}\}_{i=1}^{m-1} \cup \{\boldsymbol{U}\} \cup \{\boldsymbol{U}_{j,k}^{(T_0)}\}_{j=m+1}^{M}\Big) \\
&\qquad + \frac{\lambda}{2}\|\boldsymbol{U} - \boldsymbol{U}_{m,k}^{(T_0)}\|_F^2 \\
&= \underset{\boldsymbol{U}\in\mathbb{U}^{I_m\times R_m}}{\arg\min}\ \varphi \sum_{t=1}^{T_0} \|\boldsymbol{\mathcal{G}}_{t,k}^{(T_0)} - \boldsymbol{\mathcal{H}}_t \times_m \boldsymbol{U}^H\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{U} - \boldsymbol{U}_{m,k}^{(T_0)}\|_F^2 \\
&= \underset{\boldsymbol{U}\in\mathbb{U}^{I_m\times R_m}}{\arg\max}\ \Re\mathrm{trace}\left(\boldsymbol{U}^H\left(\sum_{t=1}^{T_0}(\boldsymbol{\mathcal{H}}_t)_{(m)}(\boldsymbol{\mathcal{G}}_{t,k}^{(T_0)})_{(m)}^H + \frac{\lambda}{2\varphi}\boldsymbol{U}_{m,k}^{(T_0)}\right)\right),
\end{aligned} \tag{15}$$

where the last equality holds by utilizing the orthogonal constraints $\boldsymbol{U}^H\boldsymbol{U} = \boldsymbol{I}_{R_m}$, the symbol $\Re\mathrm{trace}(\boldsymbol{\mathcal{X}})$ denotes the trace of the real part of $\boldsymbol{\mathcal{X}}$, and

$$\boldsymbol{\mathcal{H}}_t = \boldsymbol{\mathcal{X}}_t \prod_{i<m} \times_i (\boldsymbol{U}_{i,k+1}^{(T_0)})^H \prod_{j>m} \times_j (\boldsymbol{U}_{j,k}^{(T_0)})^H. \tag{16}$$

With orthogonal constraints, the closed-form solution of (15) is

$$\boldsymbol{U}_{m,k+1}^{(T_0)} = \boldsymbol{L}_{m,k+1}\boldsymbol{R}_{m,k+1}^H, \tag{17}$$

where $\boldsymbol{L}_{m,k+1} \in \mathbb{U}^{I_m\times R_m}$ and $\boldsymbol{R}_{m,k+1} \in \mathbb{U}^{R_m\times R_m}$ are the left and right singular matrices of

$$\sum_{t=1}^{T_0}(\boldsymbol{\mathcal{H}}_t)_{(m)}(\boldsymbol{\mathcal{G}}_{t,k}^{(T_0)})_{(m)}^H + \frac{\lambda}{2\varphi}\boldsymbol{U}_{m,k}^{(T_0)}. \tag{18}$$

Update core tensor series $\mathscr{G}_{[T_0]}^{(T_0)}$: For $t = 1$ to T_0, the subproblem of updating $\boldsymbol{U}_m^{(T_0)}$ with proximal term and step size λ is formulated as

$$\begin{aligned}
\boldsymbol{\mathcal{G}}_{t,k+1}^{(T_0)} = \underset{\boldsymbol{\mathcal{G}}_t}{\arg\min}\ & F_{T_0}\Big(\mathscr{G}_{[t-1],k+1}^{(T_0)} \cup \{\boldsymbol{\mathcal{G}}_t\} \cup \{\boldsymbol{\mathcal{G}}_{s,k}^{(T_0)}\}_{s=t+1}^{T+1}, \mathscr{P}_{k+1}^{(T_0)}, \mathscr{U}_{k+1}^{(T_0)}\Big) \\
& + \frac{\lambda}{2}\|\boldsymbol{\mathcal{G}}_t - \boldsymbol{\mathcal{G}}_{t,k}^{(T_0)}\|_F^2.
\end{aligned} \tag{19}$$

Algorithm 1 TOPA - Stage I

Input: TTS $\mathscr{X}_{[T_0]}$, and autoregressive model f, step size $\lambda > 0$, tolerance $\epsilon > 0$ and iteration number $k = 0$

Output: Core tensor series $\mathscr{G}_{[T_0]}^{(T_0)}$, autoregressive parameters $\mathscr{P}^{(T_0)}$ and joint projection matrices $\mathscr{U}^{(T_0)}$

Initialize joint projection matrices $\mathscr{U}_0^{(T_0)} = \{\boldsymbol{U}_{m,0}^{(T_0)}\}_{m=1}^M$ randomly.
Initialize core tensor series:
$\mathscr{G}_{[T_0],0}^{(T_0)} = \{\boldsymbol{\mathcal{G}}_{t,0}^{(T_0)} := \boldsymbol{\mathcal{X}}_t \prod_{m=1}^M \times_m (\boldsymbol{U}_{m,0}^{(T_0)})^H\}_{t=1}^{T_0}$.

while $k \geq 0$ **do**
  Compute regression parameters $\mathscr{P}_{k+1}^{(T_0)}$ for f with $\mathscr{G}_{[T_0],k}^{(T_0)}$ via solving subproblem (11).
  for $m = 1 : M$ **do**
    Compute the left and right singular matrices $\boldsymbol{L_{m,k+1}}$ and $\boldsymbol{R_{m,k+1}}$ respectively of $\boldsymbol{\mathcal{W}}^{(T_0)}$ defined by (22).
    Update $\boldsymbol{U}_{m,k+1}^{(T_0)}$ via (17).
  end for
  for $t = 1 : p$ **do**
    Update $\boldsymbol{\mathcal{G}}_{t,k+1}^{(T_0)}$ via (20).
  end for
  for $t = p+1 : T_0$ **do**
    Update $\boldsymbol{\mathcal{G}}_{t,k+1}^{(T_0)}$ via (21).
  end for
  if $\|\mathscr{G}_{[T_0],k+1}^{(T_0)} - \mathscr{G}_{[T_0],k}^{(T_0)}\|_F^2 + \|\mathscr{U}_{k+1}^{(T_0)} - \mathscr{U}_k^{(T_0)}\|_F^2 + \|\mathscr{P}_{k+1}^{(T_0)} - \mathscr{P}_k^{(T_0)}\|_F^2 < \epsilon$ **then**
    Break;
  end if
  $k \leftarrow k + 1$
end while
return $\mathscr{P}^{(T_0)} = \mathscr{P}_k^{(T_0)}$, $\mathscr{U}^{(T_0)} = \mathscr{U}_k^{(T_0)}$, $\mathscr{G}_{[T_0]}^{(T_0)} = \mathscr{G}_{[T_0],k}^{(T_0)}$

Since (19) is a quadratic optimization problem, when the gradient of the objective function with respect to $\boldsymbol{\mathcal{G}}_t$ vanishes, we can obtain the solution to (19):

$$\boldsymbol{\mathcal{G}}_{t,k+1}^{(T_0)} = \frac{1}{\varphi + \frac{\lambda}{2}} \left[\varphi \boldsymbol{\mathcal{X}}_t \prod_{m=1}^M \times_m (\boldsymbol{U}_{m,k+1}^{(T_0)})^H + \frac{\lambda}{2} \boldsymbol{\mathcal{G}}_{t,k}^{(T_0)} \right] \quad \text{if } t \leq p, \tag{20}$$

and

$$\boldsymbol{\mathcal{G}}_{t,k+1}^{(T_0)} = \frac{1}{1 + \varphi + \frac{\lambda}{2}} \Big[f_{\mathscr{P}_{k+1}^{(T_0)}}(\mathscr{G}_{[t-1],k+1}^{(T_0)}) + \varphi \boldsymbol{\mathcal{X}}_t \prod_{m=1}^M \times_m (\boldsymbol{U}_{m,k+1}^{(T_0)})^H + \frac{\lambda}{2} \boldsymbol{\mathcal{G}}_{t,k}^{(T_0)} \Big] \quad \text{if } t > p. \tag{21}$$

As t traverses $[T+1]$, $\mathscr{G}_{[T_0],k}^{(T_0)}$ is updated to $\mathscr{G}_{[T_0],k+1}^{(T_0)} := \{\boldsymbol{\mathcal{G}}_{t,k+1}^{(T_0)}\}_{t \in [T]}$.

According to (16) and (18), define

$$\boldsymbol{\mathcal{W}}^{(T_0)} = \sum_{t=1}^{T_0} \left(\boldsymbol{\mathcal{X}}_t \prod_{i<m} \times_i (\boldsymbol{U}_{i,k+1}^{(T_0)})^H \prod_{j>m} \times_j (\boldsymbol{U}_{j,k}^{(T_0)})^H \right)_{(m)} \times \left(\boldsymbol{\mathcal{G}}_{t,k}^{(T_0)} \right)_{(m)}^H + \frac{\lambda}{2\varphi} \boldsymbol{U}_{m,k}^{(T_0)}. \quad (22)$$

Then, we summarize this stage of TOPA in Algorithm 1. In each iteration, we first update the regression model of the core tensor series in terms of current iteration results of core tensors (line 2 in Algorithm 1), and then update the joint projection matrices and core tensor series in the joint Tucker decomposition structure (line $3-6$ and line $7-12$, respectively).

By executing Algorithm 1, we can build an initial predictor (9) for the starting TTS, which is the basis for the subsequent online prediction stage.

3.2 Stage II: Online Predictor Updating and Prediction

In this stage, new streaming TTS data arrive in sequence. After each data sampling, we update the previous predictor online and then predict the upcoming TTS data with the updated predictor. The online manner relies on the same assumption of temporal continuity of considered TTS as discussed in Section 2.4. In this stage, we repeat *online updating* and *prediction* for online prediction, as illustrated in Figure 1, where T_0 in (11), (17) and (19) should be replaced by $T+1$. We list the main two steps as follows.

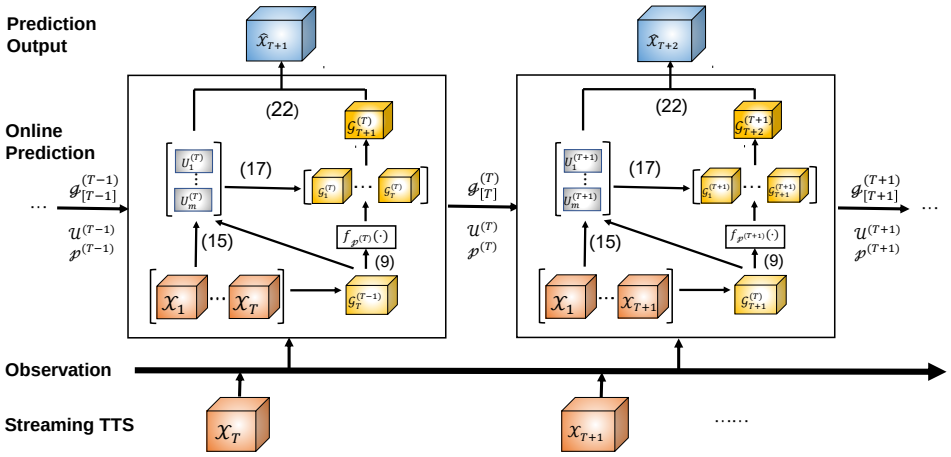

Figure 1: Stage II of TOPA for Streaming TTS

Online updating step: At time $T+1 (> T_0)$, new TTS data $\boldsymbol{\mathcal{X}}_{T+1}$ arrives. Similar to Stage I, the process of finding new predictor $h_{T+1}(\cdot)$ can be formulated as

$$\begin{aligned} \min_{\substack{\mathscr{G}_{[T+1]}^{(T+1)}, \mathscr{P}^{(T+1)}, \\ \mathscr{U}^{(T+1)}}} \quad & F_{T+1}\left(\mathscr{G}_{[T+1]}^{(T+1)}, \mathscr{P}^{(T+1)}, \mathscr{U}^{(T+1)}\right) \\ \text{s.t.} \quad & \boldsymbol{U}_m^{(T+1)} \in \mathbb{U}^{I_m \times R_M}, \quad \forall m \in [M], \end{aligned} \quad (23)$$

where

$$F_{T+1}\left(\mathscr{G}_{[T+1]}^{(T+1)}, \mathscr{P}^{(T+1)}, \mathscr{U}^{(T+1)}\right) = \sum_{t=p+1}^{T+1} \|\boldsymbol{\mathcal{G}}_t^{(T+1)} - f_{\mathscr{P}^{(T+1)}}(\mathscr{G}_{[t-1]}^{(T+1)})\|_F^2 + \varphi \sum_{t=1}^{T+1} \|\boldsymbol{\mathcal{G}}_t^{(T+1)} - \boldsymbol{\mathcal{X}}_t \prod_{m=1}^{M} \times_m (\boldsymbol{U}_m^{(T+1)})^H\|_F^2.$$

Under the assumption of temporal continuity of TTS, the joint feature subspaces among TTS data and core tensor series change smoothly as observation continues (Jing et al., 2018). Moreover, compared to $F_T(\cdot)$, $F_{T+1}(\cdot)$ contains only two additional terms that relate to $\boldsymbol{\mathcal{X}}_{T+1}$. Thus the change in the objective function for the online optimization problem is relatively small. When $h_T(\cdot)$ is accurate enough, we can employ it as the initialization for solving (23). Specifically, we take $\mathscr{G}_{[T]}^{(T)}$, $\mathscr{U}^{(T)}$ and $\mathscr{P}^{(T)}$ as the initialization of (23), and use an alternative updating scheme to solve (23) in closed forms. The details of alternative updating are shown in the online updating step of Algorithm 2. The updates of $\mathscr{G}_{[T+1]}^{(T+1)}$, $\mathscr{P}^{(T+1)}$, and $\mathscr{U}^{(T+1)}$ are similar to (12), (17), (20) and (21). We omit the process of update here.

Online prediction step: After online updating the predictor with new observation, TOPA predicts the upcoming tensor data at time $T+2$ by

$$\begin{aligned}\hat{\boldsymbol{\mathcal{X}}}_{T+2} &= f_{\mathscr{P}^{(T+1)}}\left(\mathscr{G}_{[T+1]}^{(T+1)}\right) \prod_{m=1}^{M} \times_m \boldsymbol{U}_m^{(T+1)} \\ &=: h_{T+1}(\mathscr{X}_{[T+1]}). \end{aligned} \tag{24}$$

In the practice of our proposed online manner, we only need to run a few iterations to find a predictor accurate enough, owing to the inheritance of the previous solution. In contrast, offline prediction algorithms, such as those discussed in Jing et al. (2018) and Shi et al. (2020), need to perform multiple iterations with random initialization to solve (23), which is inefficient and repetitive during streaming observations. The simulations in Section 5 demonstrate that our online updating scheme performs as well as those offline methods, even with just one iteration.

3.3 Convergence and Complexity Analysis

In this subsection, we analyze the convergence and complexity of TOPA. Since the online updating process of Algorithm 2 is similar to Algorithm 1, we only provide convergence analysis for Algorithm 1 here.

Denote the sequence generated by Algorithm 1 as $\left\{\mathcal{W}_k^{(T_0)} = \left(\mathscr{G}_{[T_0],k}^{(T_0)}, \mathscr{P}_k^{(T_0)}, \mathscr{U}_k^{(T_0)}\right)\right\}$. Let $\mathcal{W}_*^{(T_0)} = \left(\mathscr{G}_{[T_0],*}^{(T_0)}, \mathscr{P}_*^{(T_0)}, \mathscr{U}_*^{(T_0)}\right)$ be any of its accumulation point. Let $N(\boldsymbol{U}_{m,*}^{(T_0)})$ be the normal cone of $\mathbb{U}^{I_m \times R_m}$ at $\boldsymbol{U}_{m,*}^{(T_0)}$, as defined in Rockafellar and Wets (2009). Then, we have the following convergence result, of which the proof is presented in Appendix A.

Algorithm 2 TOPA - Stage II

Input: Streaming TTS $\mathscr{X}_{[T]}$ from $T = T_0$, autoregressive model f, step size $\lambda > 0$, and tolerance $\epsilon > 0$

Output: Online prediction results $\{\hat{\boldsymbol{\mathcal{X}}}_t\}_{t=T_0+2}^{\infty}$

while observing new data $\boldsymbol{\mathcal{X}}_{T+1}$ at time $T+1$: do
  $k \leftarrow 0$
  step 1: online updating
  $\boldsymbol{\mathcal{G}}_{T+1}^{(T)} := \frac{f_{\mathscr{P}^{(T)}}(\mathscr{G}_{[T]}^{(T)}) + \varphi \boldsymbol{\mathcal{X}}_{T+1} \prod_{m=1}^{M} \times_m (\boldsymbol{U}_m^{(T)})^H}{1+\varphi}$
  $\mathscr{P}_k^{(T+1)} = \mathscr{P}^{(T)}, \mathscr{U}_k^{(T+1)} = \mathscr{U}^{(T)}, \mathscr{G}_{[T+1],k}^{(T+1)} = \mathscr{G}_{[T+1]}^{(T)}$
  while $k \geq 0$ **do**
    Compute regression parameters $\mathscr{P}_{k+1}^{(T+1)}$ for f with $\mathscr{G}_{[T+1],k}^{(T+1)}$ via solving (11).
    for $m = 1 : M$ **do**
      Compute the left and right singular matrices $\boldsymbol{L_{m,k+1}}$ and $\boldsymbol{R_{m,k+1}}$ respectively of $\boldsymbol{\mathcal{W}}^{(T+1)}$ defined similar to (22).
      Update $\boldsymbol{U}_{m,k+1}^{(T+1)} \leftarrow \boldsymbol{L}_{m,k+1} \boldsymbol{R}_{m,k+1}^H$
    end for
    for $t = 1 : p$ **do**
      $\boldsymbol{\mathcal{G}}_{t,k+1}^{(T+1)} \leftarrow \frac{\varphi \boldsymbol{\mathcal{X}}_t \prod_{m=1}^{M} \times_m (\boldsymbol{U}_{m,k+1}^{(T+1)})^H + \frac{\lambda}{2} \boldsymbol{\mathcal{G}}_{t,k}^{(T+1)}}{\varphi + \frac{\lambda}{2}}.$
    end for
    for $t = p+1 : T+1$ **do**
      $\boldsymbol{\mathcal{G}}_{t,k+1}^{(T+1)} \leftarrow \frac{f_{\mathscr{P}_{k+1}^{(T+1)}}(\mathscr{G}_{[t-1],k+1}^{(T+1)}) + \varphi \boldsymbol{\mathcal{X}}_t \prod_{m=1}^{M} \times_m (\boldsymbol{U}_{m,k+1}^{(T+1)})^H + \frac{\lambda}{2} \boldsymbol{\mathcal{G}}_{t,k}^{(T+1)}}{1 + \varphi + \frac{\lambda}{2}}.$
    end for
    if $\|\mathscr{G}_{[T+1],k+1}^{(T+1)} - \mathscr{G}_{[T+1],k}^{(T+1)}\|_F^2 + \|\mathscr{U}_{k+1}^{(T+1)} - \mathscr{U}_k^{(T+1)}\|_F^2 + \|\mathscr{P}_{k+1}^{(T+1)} - \mathscr{P}_k^{(T+1)}\|_F^2 < \epsilon$ **then**
      Break;
    end if
    $k \leftarrow k+1$
  end while
  $\mathscr{P}^{(T+1)} = \mathscr{P}_k^{(T+1)}, \quad \mathscr{U}^{(T+1)} = \mathscr{U}_k^{(T+1)}, \quad \mathscr{G}_{[T+1]}^{(T+1)} = \mathscr{G}_{[T+1],k}^{(T+1)}$
  step 2: online prediction
    $\hat{\boldsymbol{\mathcal{X}}}_{T+2} = f_{\mathscr{P}^{(T+1)}}(\mathscr{G}_{[T+1]}^{(T+1)}) \prod_{m=1}^{M} \times_m \boldsymbol{U}_m^{(T+1)}.$
  return $\hat{\boldsymbol{\mathcal{X}}}_{T+2}$
  $T \leftarrow T+1$
end while

Theorem 1. *Assume that the starting TTS and the regression parameter sequence $\{\mathscr{P}_k^{(T_0)}\}_k$ generated by Algorithm 1 are bounded. Then, for any limit points $\mathcal{W}_*^{(T_0)}$ of the sequence*

$\left\{\mathcal{W}_k^{(T_0)}\right\}$, *we have*

$$\frac{\partial F_{T_0}\left(\mathcal{W}_*^{(T_0)}\right)}{\partial \mathscr{P}^{(T_0)}} = \mathbf{0}, \quad \frac{\partial F_{T_0}\left(\mathcal{W}_*^{(T_0)}\right)}{\partial \mathscr{G}_{[T_0]}^{(T_0)}} = \mathbf{0}, \tag{25}$$

and

$$-\frac{\partial F_{T_0}\left(\mathcal{W}_*^{(T_0)}\right)}{\partial \boldsymbol{U}_m^{(T_0)}} \in N(\boldsymbol{U}_{m,*}^{(T_0)}). \tag{26}$$

In other words, any limit point of the iteration sequence generated by Algorithm 1 is the stationary point of (10).

The computational complexity of TOPA-Stage II is dominated by updating joint projection matrices and core tensor series. For each iteration in online prediction, the total computational complexity is $O\left(M^2T\bar{I}^M\bar{R} + MT\bar{R}^M\bar{I}\right)$, where $\bar{I} = (\prod_{m\in[M]} I_m)^{1/M}$ and $\bar{R} = (\prod_{m\in[M]} R_m)^{1/M}$ are the geometric average of the scale of TTS data and core tensors, respectively. For details, we list the computational complexity of each iteration during online prediction by TOPA in Table 1. Here we denote the total order of AR model f is p, which is assumed much less than the scale of tensor data.

Step	Computational Complexity
Line 4 of TOPA-Stage II	$O\left(M\bar{I}^M\bar{R}\right)$
Computing $\mathscr{P}_{k+1}^{(T+1)}$	$O\left(p^3\bar{R}^M\right)$
(17)	$O\left(MT((M-1)\bar{I}^M\bar{R} + \bar{R}^M\bar{I})\right)$
(19)	$O\left(T(M\bar{I}^M\bar{R} + p\bar{R}^M)\right)$
Total	$O\left(M^2T\bar{I}^M\bar{R} + MT\bar{R}^M\bar{I}\right)$

Table 1: Computational Complexity of Stage II in TOPA

4. TOPA-AAW: TOPA with Automatically Adaptive Weights

In real-world scenarios, when streaming TTS data are observed continuously over a relatively long period, the effectiveness of stale data in revealing the latest statistical patterns in streaming TTS gradually diminishes. This leads to an increasing cumulative prediction error, such as wireless CSI (Letaief and Zhang, 2006), in which the intrinsic statistical patterns evolve rapidly as observations continue, and the data becomes outdated quickly. To address this issue, we propose an automatically adaptive weight (AAW) regularization method with a time sliding window τ for modifying (23). The AAW gradually decreases the weights of stale data to reduce their impacts on prediction accuracy. To further eliminate the interference caused by heavily noisy data, we introduce an automatic factor in AAW to reduce the weights of heavily noisy data.

The optimization problem for online prediction, once $\boldsymbol{\mathcal{X}}_{T+1}$ is observed at time $T+1$, is modified as:

$$\begin{aligned} \min_{\substack{\mathscr{P}^{(T+1)},\mathscr{U}^{(T+1)},\\ \mathscr{G}^{(T+1)}_{[T+1]}}} \; & F^{(\mathrm{AAW})}_{T+1}(\mathscr{G}^{(T+1)}_{[T+1]},\mathscr{P}^{(T+1)},\mathscr{U}^{(T+1)}) \\ \text{s.t.} \quad & \boldsymbol{U}^{(T+1)}_m \in \mathbb{U}^{I_m\times R_M}, \quad \forall m\in[M], \end{aligned} \tag{27}$$

where

$$\begin{aligned} F^{(\mathrm{AAW})}_{T+1}(\mathscr{G}^{(T+1)}_{[T+1]},\mathscr{P}^{(T+1)},\mathscr{U}^{(T+1)}) = \sum_{t=T-\tau+2}^{T+1} \Big(& \|\boldsymbol{\mathcal{G}}^{(T+1)}_t - f_{\mathscr{P}^{(T+1)}}(\mathscr{G}^{(T+1)}_{[t-1]})\|^2_F \\ & + \varphi\omega^{(T+1)}_t \|\boldsymbol{\mathcal{G}}^{(T+1)}_t - \boldsymbol{\mathcal{X}}_t \prod_{m=1}^{M} \times_m (\boldsymbol{U}^{(T+1)}_m)^H\|^2_F \Big), \end{aligned}$$

and $\omega^{(T+1)}_t$ is the AAW

$$\omega^{(T+1)}_t = \begin{cases} (1-\alpha^{t-(T-\tau+1)})\max\{\beta, 1-\epsilon^{(T+1)}_t\}, & T-\tau+2 \le t \le T, \\ 1, & t = T+1 \end{cases} \tag{28}$$

with Tucker-decomposition residual error

$$\epsilon^{(T+1)}_t = \frac{\|\boldsymbol{\mathcal{X}}_t - \boldsymbol{\mathcal{G}}^{(T+1)}_t \prod_{m=1}^{M} \times_m \boldsymbol{U}^{(T+1)}_m\|^2_F}{\|\boldsymbol{\mathcal{X}}_t\|^2_F}. \tag{29}$$

Here $\alpha \in (0,1)$ is the damping parameter for reducing the accumulative prediction error from stale data. The 'max' factor is negatively related to the residual error $\epsilon^{(T+1)}_t$ of joint tensor decomposition for streaming TTS in the last prediction, while $\beta \in (0,1)$ is the minimum residual factor.

Remark 3. The parameter β is configured as a threshold for checking the deviation of the joint Tucker decomposition for each data in TTS, which can help prevent the data with significant decomposition residual errors from being entirely discarded. For stable TTS with gently evolving joint subspaces, such as two real-world data sets in Section 5.2, β can be broadly chosen from 0 to a positive number close to 1 since the threshold is inactive. For unstable TTS with fast-evolving joint subspaces, such as the wireless channel discussed in Section 5.1, β should be set properly to deal with fast-fading or noisy channel states. In addition, we specifically discuss the effects of choice of α and φ in Section 5.2.

Since the structure of (27) is similar to (23) except for the sliding time window and AAW, we solve (27) in a similar online manner as Algorithm 2. To avoid repetitive presentation of the formulas, we give the online updating steps and the prediction steps of TOPA-AAW in Appendix B.

With the time sliding window, TOPA-AAW further reduces the computational complexity to τ/T times that of TOPA. In terms of Table 1, the total computational complexity of each iteration in TOPA-AAW is $O\left(M^2\tau\bar{I}^M\bar{R} + M\tau\bar{R}^M\bar{I}\right)$.

5. Numerical Experiments

In this section, we evaluate the performance of TOPA in various scenarios, including synthetic low-rank TTS, wireless channel simulations, and two real-world data sets. All numerical experiments are conducted using MATLAB R2018b on a Windows PC with a quad-core Intel(R) Core(TM) 2.0GHz CPU and 8 GB RAM.

To the best of our knowledge, TOPA is the first method for online prediction of generic streaming TTS. In order to provide a reference for comparing the performance of TOPA, we also conduct five other non-online prediction methods:

- **BHT-ARIMA** (Shi et al., 2020) and **MCAR** (Jing et al., 2018): These are two effective offline one-shot prediction methods for TTS. We run these offline methods using the same online observations at each sampling time as the online methods by building their predictors from scratch with multiple iterations to forecast the next tensor data. BHT-ARIMA employs BHT, an Hankelized tensor expansion technique, to further exploit the temporal relationships among TTS and uses a structure similar to MCAR to formulate predictors. The offline methods perform sufficient iterations until convergence in each prediction, thereby revealing good prediction performance.
- **TOPA-init**: To illustrate the importance of the online updating for predictors, we test the performance of TOPA without model updating in some experiments. In other words, we use the initial predictor obtained from Algorithm 1 and the streaming observations to forecast upcoming data at each sampling time.
- **LSTM** (Hochreiter and Schmidhuber, 1997) and **GMRL** (Deng et al., 2023): These are two neural-network-based prediction methods. Long Short-Term Memory (LSTM) is a classical time-series forecasting method in the deep learning area, which uses a type of recurrent neural network (RNN) architecture that is designed to process and retain information over long sequences. GMRL is a latest work based on neural networks equipped with the tensor decomposition framework.

When comparing the time costs of LSTM and GMRL, we did not account for the training time costs of neural networks, though it may cost much time in practice. On the other hand, owing to the predictor inheritance of TOPA, we conduct TOPA and TOPA-AAW with **only one iteration** at each sampling time during the online prediction. The purpose of such experimental setup is not to unfairly compare the speed of our algorithms, but to determine whether our online methods can achieve accuracy similar to that of offline methods while reducing the number of iterations and thus saving on prediction time costs.

In addition, taking two real-world data sets in Section 5.2 as examples, we discuss the effects of parameter choice for TOPA-AAW. We evaluate the prediction accuracy of algorithms with the Normalized Root Mean Square Error (NRMSE) metric:

$$\text{NRMSE} = \frac{\sqrt{\sum_{t\geq T+1}\|\hat{\boldsymbol{\mathcal{X}}}_t - \boldsymbol{\mathcal{X}}_t\|_F^2}}{\sqrt{\sum_{t\geq T+1}\|\boldsymbol{\mathcal{X}}_t\|_F^2}}. \tag{30}$$

The standard error (SE) of prediction errors is commonly used to assess the significance of differences among prediction results. In the Monte Carlo experiments presented in Section 5.1, we use the average SE for this evaluation.

5.1 Synthetic Data Sets

In this subsection, we evaluate the performance of TOPA in synthetic low-rank TTS and wireless channel simulations. The numerical results indicate that TOPA and TOPA-AAW have evident advantage on time-consuming and prediction accuracy.

Synthetic Low-Rank Streaming TTS: We generate a low-rank noisy streaming TTS with a similar process as Sun et al. (2020, Sec. 6.3). The low-rank TTS structure follows the joint Tucker decomposition. We first generate the core tensor series $\{\boldsymbol{\mathcal{G}}_t \in \mathbb{C}^{4\times4\times4}\}_{t=1}^{70}$ with ARIMA$(3,1,0)$ model. Then, we generate the joint feature subspace matrices $\{\boldsymbol{U}_m \in \mathbb{U}^{20\times4}\}_{m=1}^{3}$, by orthogonalizing randomly generated matrices with i.i.d. $\mathcal{N}(0,1)$ entries. The generated noisy TTS are formulated as

$$\boldsymbol{\mathcal{X}}_t = \boldsymbol{\mathcal{G}}_t \prod_{m=1}^{M} \times_m \boldsymbol{U}_m + \rho \|\boldsymbol{\mathcal{G}}_t\|_F \boldsymbol{\mathcal{E}}_t \in \mathbb{C}^{20\times20\times20}, \tag{31}$$

where $\rho = 0.1$ is the noise parameter and $\boldsymbol{\mathcal{E}}_t$ is the noise tensor with i.i.d. $\mathcal{N}(0,1)$ entries. Note that we have $\|\boldsymbol{\mathcal{G}}_t\|_F = \|\boldsymbol{\mathcal{G}}_t \prod_{m=1}^{M} \times_m \boldsymbol{U}_m\|_F$ and ρ represents the ratio of noise in TTS.

The generated TTS with noise is divided into two sets: the first $T_0 = 20$ data in the training set, and the rest 50 in the test set. The regularization parameters for TOPA-AAW are set as $\varphi = 10, \alpha = 0.4, \beta = 0.4$. All numerical results are shown in Figure 2 and Table 2.

Figure 2 compares the average NRMSE of different prediction methods with 10000 Monte Carlo experiments, each independently and randomly generates a TTS as (31). The "TOPA-init" line in Figure 2 shows that prediction error increases as prediction continues without model online updating due to the noises in TTS. With the online updating manner, TOPA reveals almost the same performance as MCAR, while BHT-ARIMA has a little advantage over these two methods. Two neural-network-based methods do not perform as well as other algorithms, similar to the analysis in Kong et al. (2024), because these methods treat the data as vectors and fail to preserve the tensor structure of data, thereby ignoring the implicit regression patterns of the core tensor series.

From Table 2, it is easy to see that among all these methods, TOPA and TOPA-AAW are much more efficient owing to the one-loop algorithms, while also keep high prediction accuracy similar to the offline methods. Moreover, the absolute prediction errors do not show large fluctuations, which reflects the stability of our algorithms.

	TOPA	TOPA-AAW	MCAR	BHT-ARIMA	GMRL	LSTM
Time(ms)	<u>47.8</u>	**25.1**	77.4	238.4	200.7	72.2
NRMSE	0.0776	<u>0.0760</u>	0.0778	**0.0750**	0.3299	0.1390
Average SE	0.0026	0.0030	0.0026	0.0024	0.0284	0.0393

Table 2: Average time costs, NRMSE and average SE of different algorithms on synthetic TTS

Wireless CSI Prediction: We consider the wireless channel prediction problem with high-frequency observation and wild fluctuation of statistical features. For fifth-generation wireless communication, many massive multiple-input multiple-output (MIMO) transmission techniques highly rely on accurate CSI acquisition, such as precoding and beamforming.

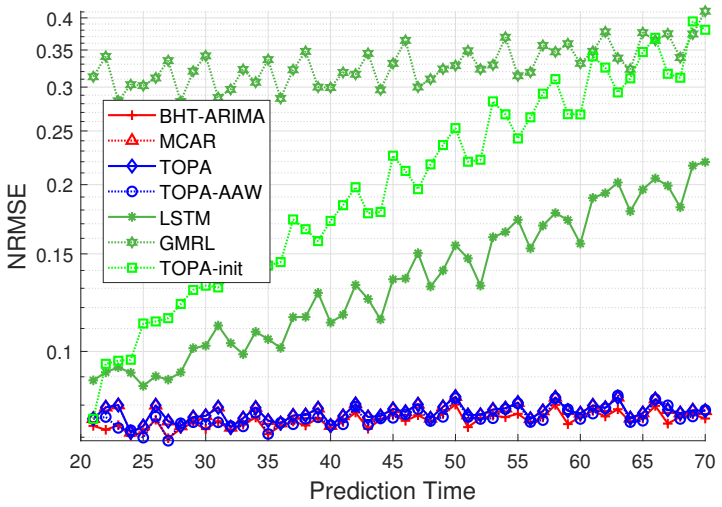

Figure 2: Performance of different prediction methods on synthetic TTS (31). TOPA and TOPA-AAW reveal almost the same performance as the offline methods.

However, the observation of CSI and later data processing usually quickly become outdated due to the short coherent time. CSI prediction is a natural way to tackle this issue. For time-varying channels, the online manner is important to predict CSI in upcoming time slots.

We use QuaDRiGa (Jaeckel et al., 2014) to generate realistic CSI streaming data for single base-station (BS) and single user-equipment (UE) downlink transmission. BS and UE are equipped with 16 and 2 antennas, respectively. The transmission occupies the 700MHz frequency with 10MHz bandwidth and 50 subcarriers. During the CSI observation, BS is static, and UE moves with 1.5m/s speed. The CSI observation frequency is configured as $f_s = 25$Hz, so we sample CSI per 40ms. Then, the CSI observation produces a streaming TTS $\{\boldsymbol{\mathcal{X}}_t \in \mathbb{C}^{16\times 2\times 50}\}_{t=1}^{\infty}$, of which $(\boldsymbol{\mathcal{X}}_t)_{ijk}$ denotes the channel state at time t on k-th subcarrier between BS's i-th antenna and UE's j-th antenna.

We run 1000 times Monte Carlo (MC) simulations to evaluate the performance of our proposed algorithms. For each MC simulation, we randomly select a continuous part of CSI TTS with 70 samples and let the first $T_0 = 20$ CSI tensors be the training set, while the last 50 tensors are the streaming data observed in the following 50 sampling time. In order to make the core tensors sufficiently represent the intrinsic interactions among multiple feature subspaces of the original TTS, we estimate the rank of the core tensors with the rank-adaptive method (Xiao and Yang, 2021; Liu et al., 2023). With this method, according to the singular value distribution of the unfolding matrix of each mode, we truncate the first original tensor data to obtain the rank-estimation of the whole original TTS. For the synthetic CSI data, the scale of core tensors is set as $(10, 2, 20)$, which significantly compresses the TTS data. The regularization parameters for the residual of tensor decomposition are set as $\varphi = 10, \alpha = 0.9, \beta = 0.6$. Since the CSI fluctuates wildly, we shorten the time sliding window in TOPA-AAW to $\tau = 8$. To balance the model complexity and the

prediction accuracy, we use the Bayesian Information Criterion (BIC, Schwarz, 1978) for the original TTS. During MC simulations, all tensor-based prediction methods employ the ARIMA$(2, 1, 1)$ model to build temporal relations among the core tensor series. To further improve the adaptability of TOPA/TOPA-AAW for the time-varying property of CSI, we conduct two iterations instead in the online update step of Algorithm 2. All numerical results are shown in Figure 3 and Table 3.

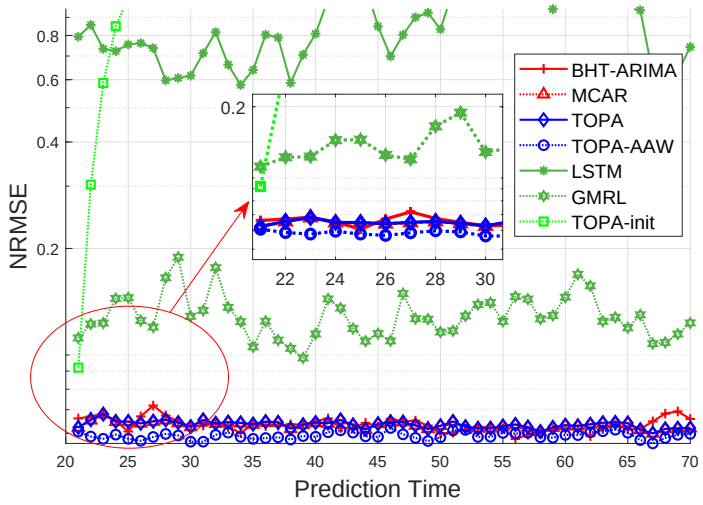

Figure 3: Performance of different prediction methods on CSI prediction. TOPA-AAW has the best performance, while TOPA performs as well as the offline methods.

	TOPA	TOPA-AAW	MCAR	BHT-ARIMA	GMRL	LSTM
Time(ms)	186.8	<u>37.6</u>	915.8	2634.7	**22.6**	50.7
NRMSE	0.0636	**0.0591**	0.0635	0.0634	0.1295	0.9170
Average SE	0.0161	0.0181	0.0160	0.0158	0.0190	0.2854

Table 3: Average Time costs, NRMSE and Average SE of different algorithms on CSI prediction

Figure 3 illustrates the average NRMSE of CSI online prediction for different prediction methods. The "TOPA-init" line in Figure 3 reveals the extreme change in the wireless environment, even in low-speed mobility scenarios. As an online method, TOPA shows very close performance to two offline methods, while the prediction accuracy of TOPA-AAW is much better than TOPA and MCAR, owing to the good adaptation of AAW to the wireless channel evolution.

Table 3 presents the average time costs and NRMSE of six methods in each prediction. It should be emphasised that the training of the GMRL and LSTM methods cost up to a few minutes, which is not considered in Table 3. TOPA-AAW is much more efficient than other methods except GMRL, while has the best prediction performance. Table 3 illustrates

that the difference in the number of necessary alternative iterations for building predictors leads to a significant difference in the time costs between online and offline methods.

Remark 4. In these experiments, we do not adjust the scale of the core tensors. For very long TTS, we can periodically re-estimate the rank of latest tensor data and change the scale of core tensors correspondingly. The dimensions of subspaces are also changed, and the feature subspaces are computed by re-truncating each unfolding matrix with the new rank.

5.2 Real-world Data Sets: USHCN and NASDAQ100

In this subsection, We apply our prediction algorithm to two real-world data sets:

- *USHCN*[1]: this data set records the monthly climate data of 1218 weather stations in the United States during the past 100 years, including four statistical features: monthly mean maximum temperature, monthly mean minimum temperature, monthly minimum temperature, and monthly total precipitation. 120 meteorological stations with relatively complete data are screened for numerical experiments. With their quarterly average observation data from 1940 to 2014, we establish a 75-length TTS with the scale $120 \times 4 \times 4$ of each tensor data.

- *NASDAQ100*[2]: This data set records the daily business data of 102 NASDAQ-listed companies over 90 days in 2014, including five statistical features: opening quotation, highest quotation, lowest quotation, closing quotation and adjusting the closing price. Therefore, we obtain a 90-length TTS with the scale 102×5 of each matrix data.

We divide the screened USHCN TTS into two parts: the first $T_0 = 40$ data are used for finding the initial predictor and predicting the data at sampling time 41, and the last 35 data are regarded as the streaming observation of USHCN TTS. In terms of the discussion in Jing et al. (2018, Sec. V), we choose AR(2) model to build the temporal correlations among core tensor series. In the structure of joint tensor decomposition, the scale of core tensors is given as $12 \times 4 \times 4$. Since the statistical characteristics of data are nearly stable, we configure a wide time sliding window in TOPA-AAW with length $\tau = 20$.

For the NASDAQ100 data set, we find the initial predictor with the first $T_0 = 60$ traffic flow data, and observe the last 30 data stream. The ARIMA$(3, 1, 0)$ model captures the statistical characteristics in NASDAQ100 TTS. The time sliding window used in TOPA-AAW is configured as $\tau = 20$.

We first research the effects of different choices of α and φ in (27). Figure 4 shows the average NRMSE of the proposed TOPA-AAW algorithm with different values of parameters φ and α. Regarding Figure 4 (a), we can see that the best φ of USHCN and NASDAQ100 data sets are 0.2 and 20, respectively. For both two data sets, the prediction performance is unsatisfactory when φ is relatively small, which manifests the necessity of regularization. For stable TTS, small φ makes the regularization term close to 0 with sufficiently accurate joint Tucker decomposition, hence the objective function in (27) is dominated by the first term, which leads to inaccurate joint decomposition with online prediction continuing.

1. This data set is in https://www.ncei.noaa.gov/pub/data/ushcn/v2.5/
2. This data set is in https://github.com/Karin-Karlsson/stockdata

Furthermore, when φ is large enough, the accuracy of TOPA-AAW is improved to a stable level. From Figure 4 (b), the best α for USHCN and NASDAQ100 data sets is close to 1, which illustrates the effectiveness of reducing the weight of stale data. The prediction performance is extremely poor when α is set as 1, since the regularization in (27) with $\alpha = 1$ only retains the last term with time $t = T+1$. In brief, when $\alpha < 1$ and φ are large enough, the accuracy of TOPA-AAW is not very sensitive to the choice of these parameters, which allows us to choose the parameter more flexibly. In the following experiments, we configure the regularization parameters for TOPA-AAW as: $\varphi = 0.5$ and $\alpha = 0.98$ for USHCN, and $\varphi = 20$ and $\alpha = 0.99$ for NASDAQ100. Moreover, as discussed in Remark 3, we set $\beta = 0.5$.

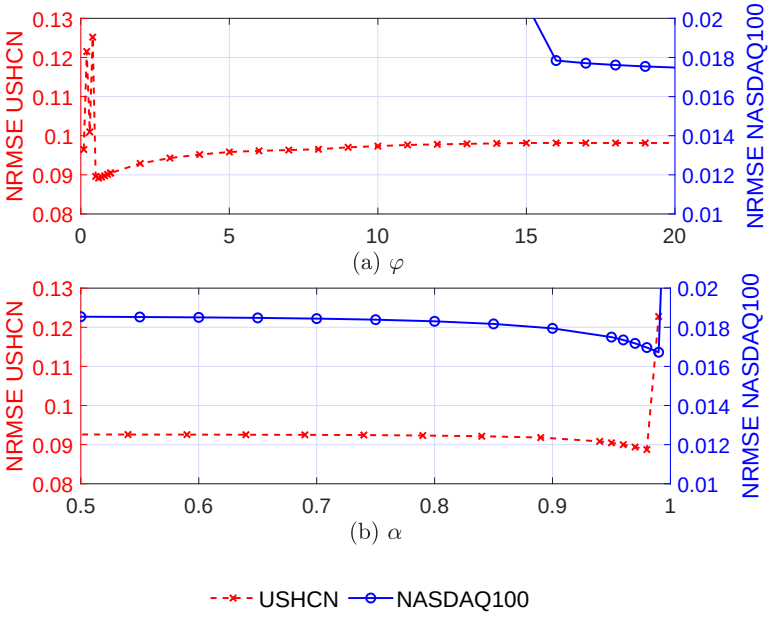

Figure 4: Effect of parameter φ and α for the TOPA-AAW algorithm on two real-world data sets: (a) average NRMSE with $\alpha = 0.95$ and different φ, and (b) average NRMSE with $\varphi_{\text{UHSCN}} = 0.5$, $\varphi_{\text{NASDAQ100}} = 20$ and different α. When $\alpha < 1$ and φ are large enough, the accuracy of TOPA-AAW is not sensitive to the choice of these parameters.

Figure 5 and Table 4 show the average NRMSE and time costs of the proposed TOPA-AAW algorithm with different scales of core tensors, respectively. We implement the experiments with three different sizes of the core tensors: large scale with dimensions of $80 \times 4 \times 4$, medium scale with dimensions of $12 \times 4 \times 4$, and small scale with dimensions of $3 \times 3 \times 3$. The results show that a larger scale of core tensors leads to higher prediction accuracy with more time costs. Therefore, choosing the right size of core tensor is a trade-off between prediction accuracy and time costs.

Next we focus on the impact of the length of the time sliding window on the prediction performance of TOPA-AAW. As shown in Figure 6, a longer sliding time window (with

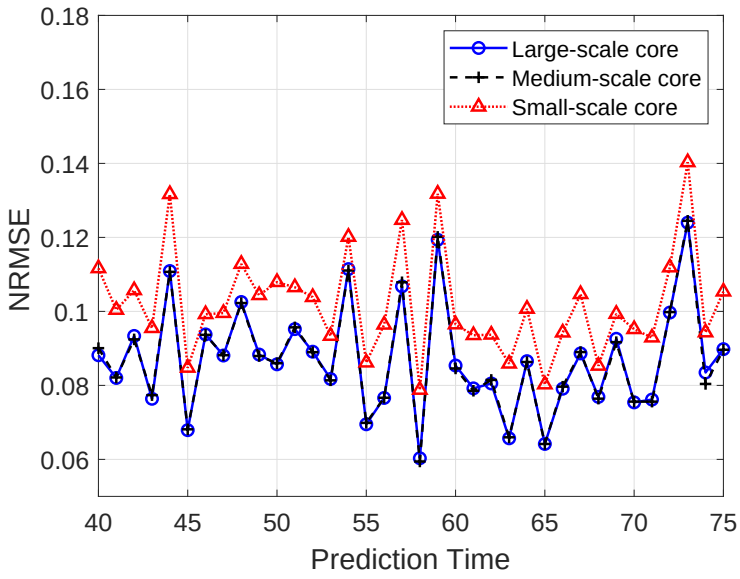

Figure 5: Performance of different sizes of core tensors for TOPA-AAW algorithm on USHCN data set.

	Large scale	Medium scale	Small scale
Time(ms)	8.7	5.1	4.8

Table 4: Average time costs of different sizes of core tensors for TOPA-AAW algorithm on USHCN data set

$\tau = 5$ and $\tau = 10$) leads to better prediction performance. On the other side, the complexity analysis in Section 3.3 proposes that the computational complexity of TOPA-AAW is positively proportional to the time window length τ. With the trade-off between prediction accuracy and time costs, we choose the length of time sliding window $\tau = 20$ in the following experiments.

Figure 7 depicts the comparison results of prediction performance. TOPA reveals nearly the same accuracy for two real-world data sets as offline methods, though it runs much fewer iterations than MCAR in each prediction. TOPA-AAW can further improve the accuracy of TOPA by tracking the latest statistical patterns in TTS. In the NASDAQ100 data set, the prediction accuracy of TOPA-AAW is nearly 20% better than TOPA and MCAR, while three methods show significant advantages over BHT-ARIMA and two neural-network-based methods.

Table 5 presents the average time costs of six prediction methods. Owing to the online manner, TOPA/TOPA-AAW can provide prediction results efficiently and accurately.

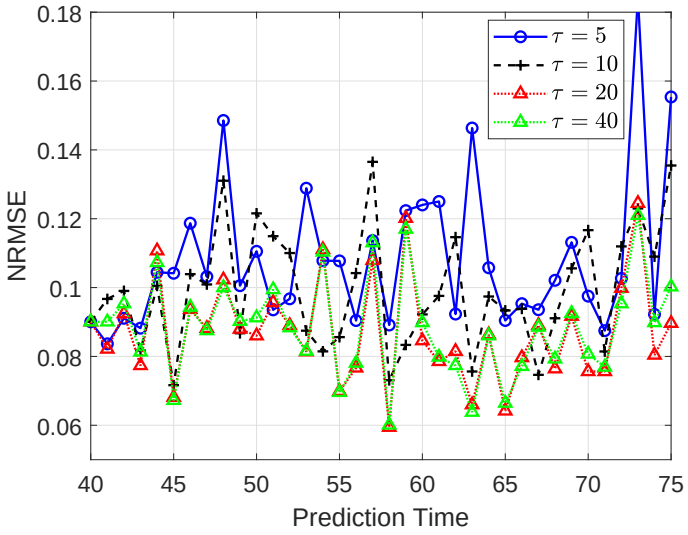

Figure 6: Performance of different sliding time window lengths for TOPA-AAW algorithm on USHCN data set.

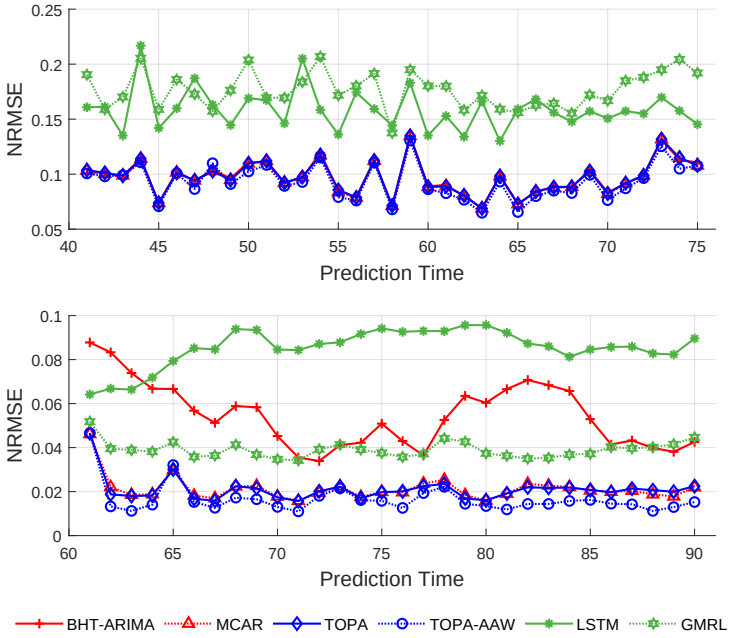

Figure 7: Performance of different prediction methods on two real-world data sets: (a) USHCN, and (b) NASDAQ100. TOPA-AAW shows the best performance owing to AAW.

Time(ms)	TOPA	TOPA-AAW	MCAR	BHT-ARIMA	GMRL	LSTM
USHCN	59.2	**5.1**	328.7	1583.1	15.0	34.6
NASDAQ100	53.5	<u>11.4</u>	263.0	419.2	**6.9**	26.2

Table 5: Average time costs of different algorithms on USHCN and NASDAQ100 data sets

6. Conclusions and Some Remarks

In this paper, we present a joint-tensor-factorization-based online prediction algorithm for streaming tensor time series. By leveraging tensor factorization, our algorithm effectively compresses the streaming data while capturing the underlying intrinsic correlations. The proposed online updating scheme significantly enhances the speed of predictor updating and can maintain high prediction accuracy. We also analyze the convergence of the proposed algorithm. Additionally, we introduce automatically adaptive weights to address the challenge of data staleness in streaming data. Through the numerical experiments in various scenarios, we observe promising results that validate the effectiveness and efficiency of our approach.

TOPA will contribute to advancing the research on tensor-neural and semi-parametric forecasting architectures. For tensor-augmented neural network systems, e.g., TEAFormers (Kong et al., 2024), our proposed online Tucker decomposition procedure can serve as a low-cost block to further reduce the input dimensionality for subsequent neural layers and the computational redundancy in real-time high-frequency prediction scenarios. On the other hand, the proposed algorithm TOPA can be naturally integrated with semi-parametric tensor factor models (e.g., STEFA model in Chen et al., 2024), in which the projection matrices are further decomposed into covariate-explainable (by non-parametric functions) and covariate-independent components. By adjusting the objective function and adding the constraints to the covariates for the projection matrices in TOPA, the online prediction scheme would enable more interpretable online updates of the tensor factorization.

Acknowledgments

Acknowledgements The work of Liping Zhang is supported by the National Natural Science Foundation of China (Grant No. 12571323, 12171271, 12271291). Defeng Sun' research is supported by the Hong Kong RGC Senior Research Fellow Scheme [No. SRFS22235S02] and the GRF Grant 15307822.

Appendix A. Proof of Theorem 1

Proof Denote $\mathcal{W}_k^{(T_0)} = (\mathscr{G}_{[T_0],k}^{(T_0)}, \mathscr{P}_k^{(T_0)}, \mathscr{U}_k^{(T_0)})$. In terms of the alternative manner of Algorithm 1, we have

$$\begin{aligned} F_{T_0}\left(\mathscr{G}_{[T_0],k}^{(T_0)}, \mathscr{P}_k^{(T_0)}, \mathscr{U}_k^{(T_0)}\right) \geq & F_{T_0}\left(\mathscr{G}_{[T_0],k}^{(T_0)}, \mathscr{P}_{k+1}^{(T_0)}, \mathscr{U}_k^{(T_0)}\right) + \frac{\lambda}{2}\|\mathscr{P}_{k+1}^{(T_0)} - \mathscr{P}_k^{(T_0)}\|_F^2 \\ \geq & F_{T_0}\left(\mathscr{G}_{[T_0],k}^{(T_0)}, \mathscr{P}_{k+1}^{(T_0)}, \mathscr{U}_{k+1}^{(T_0)}\right) + \frac{\lambda}{2}\|\mathscr{P}_{k+1}^{(T_0)} - \mathscr{P}_k^{(T_0)}\|_F^2 \\ & + \frac{\lambda}{2}\|\mathscr{U}_{k+1}^{(T_0)} - \mathscr{U}_k^{(T_0)}\|_F^2 \\ \geq & F_{T_0}\left(\mathscr{G}_{[T_0],k+1}^{(T_0)}, \mathscr{P}_{k+1}^{(T_0)}, \mathscr{U}_{k+1}^{(T_0)}\right) + \frac{\lambda}{2}\|\mathcal{W}_{k+1}^{(T_0)} - \mathcal{W}_k^{(T_0)}\|_F^2, \end{aligned} \tag{32}$$

where the three inequalities are due to the optimization subproblems (11), (15), and (19), respectively. Therefore, via recursion, we have

$$F_{T_0}\left(\mathscr{G}_{[T_0],0}^{(T_0)}, \mathscr{P}_0^{(T_0)}, \mathscr{U}_0^{(T_0)}\right) \geq F_{T_0}\left(\mathscr{G}_{[T_0],k}^{(T_0)}, \mathscr{P}_k^{(T_0)}, \mathscr{U}_k^{(T_0)}\right) + \sum_{i=1}^{k}\frac{\lambda}{2}\|\mathcal{W}_i^{(T_0)} - \mathcal{W}_{i-1}^{(T_0)}\|_F^2. \tag{33}$$

For any k and $t \leq T_0$, with the boundedness of $\boldsymbol{\mathcal{X}}_t$ and $\boldsymbol{U}_{m,k}^{(T_0)}$, we have

$$\begin{aligned} \|\boldsymbol{\mathcal{G}}_{t,k}^{(T_0)}\|_F^2 \leq & \|\boldsymbol{\mathcal{G}}_{t,k}^{(T_0)} - \boldsymbol{\mathcal{X}}_t \prod_{m=1}^{M} \times_m (\boldsymbol{U}_{m,k}^{(T_0)})^H\|_F^2 + 2\|\boldsymbol{\mathcal{X}}_t\|_F^2 \\ \leq & 2F_{T_0}\left(\mathscr{G}_{[T_0],k}^{(T_0)}, \mathscr{P}_k^{(T_0)}, \mathscr{U}_k^{(T_0)}\right) + 2\|\boldsymbol{\mathcal{X}}_t\|_F^2 \\ \leq & 2F_{T_0}\left(\mathscr{G}_{[T_0],0}^{(T_0)}, \mathscr{P}_0^{(T_0)}, \mathscr{U}_0^{(T_0)}\right) + 2\|\boldsymbol{\mathcal{X}}_t\|_F^2. \end{aligned} \tag{34}$$

Hence $\{\mathscr{G}_{[T_0],k}^{(T_0)}\}_k$ is bounded. Therefore, $\{\mathcal{W}_k^{(T_0)}\}$ is bounded and has at least one limit point.

With (33), we have

$$\sum_{i=1}^{+\infty}\|\mathcal{W}_i^{(T_0)} - \mathcal{W}_{i-1}^{(T_0)}\|_F^2 < \infty, \tag{35}$$

which implies

$$\mathcal{W}_{k+1}^{(T_0)} - \mathcal{W}_k^{(T_0)} \to \mathbf{0}. \tag{36}$$

For any limit point $\mathcal{W}_*^{(T_0)} = (\mathscr{G}_{[T_0],*}^{(T_0)}, \mathscr{P}_*^{(T_0)}, \mathscr{U}_*^{(T_0)})$ of $\{\mathcal{W}_k^{(T_0)}\}$, there exists a subsequence $\{\mathcal{W}_{k_i}^{(T_0)}\}_i$ that converges to $\mathcal{W}_*^{(T_0)}$. With (36), we have $\mathcal{W}_{k_i+1}^{(T_0)} \to \mathcal{W}_*^{(T_0)}$.

For any t and i, since $\boldsymbol{\mathcal{G}}_{t,k_i+1}^{(T_0)}$ is the solution to (19) with index k_i, we have

$$\frac{\partial F_{T_0}}{\partial \boldsymbol{\mathcal{G}}_t^{(T_0)}}\big(\mathscr{G}_{[t-1],k_i+1}^{(T_0)} \cup \{\boldsymbol{\mathcal{G}}_{t,k_i+1}^{(T_0)}\} \cup \{\boldsymbol{\mathcal{G}}_{s,k_i}^{(T_0)}\}_{s=t+1}^{T+1}, \mathscr{P}_{k_i+1}^{(T_0)}, \mathscr{U}_{k_i+1}^{(T_0)}\big) + \lambda\left(\boldsymbol{\mathcal{G}}_{t,k_i+1}^{(T_0)} - \boldsymbol{\mathcal{G}}_{t,k_i}^{(T_0)}\right) = \mathbf{0}. \tag{37}$$

Let $i \to +\infty$ in (37), we have

$$\frac{\partial F_{T_0}\left(\mathcal{W}_*^{(T_0)}\right)}{\partial \boldsymbol{\mathcal{G}}_t^{(T_0)}} = \mathbf{0}. \tag{38}$$

Similarly, we have

$$\frac{\partial F_{T_0}\left(\mathcal{W}_*^{(T_0)}\right)}{\partial \mathscr{P}^{(T_0)}} = \mathbf{0}. \tag{39}$$

Then, (25) is proved.

For any m and i, since $\boldsymbol{U}_{m,k_i+1}^{(T_0)}$ is the solution to (15) with index k_i, by Theorem 8.15 in Rockafellar and Wets (2009), we have

$$\begin{aligned} -\frac{\partial F_{T_0}}{\partial \boldsymbol{U}_m^{(T_0)}}\Big(\mathscr{G}_{[T_0],k_i}^{(T_0)}, \mathscr{P}_{k_i+1}^{(T_0)}, \{\boldsymbol{U}_{i,k_i+1}^{(T_0)}\}_{i=1}^{m-1} \cup \{\boldsymbol{U}_{m,k_i+1}^{(T_0)}\} \cup \{\boldsymbol{U}_{j,k_i}^{(T_0)}\}_{j=m+1}^{M}\Big) \\ - \lambda\left(\boldsymbol{U}_{m,k_i+1}^{(T_0)} - \boldsymbol{U}_{m,k_i}^{(T_0)}\right) \in N(\boldsymbol{U}_{m,k_i+1}^{(T_0)}). \end{aligned} \tag{40}$$

Let $i \to +\infty$, with Proposition 6.6 in Rockafellar and Wets (2009) and (40), we obtain

$$-\frac{\partial F_{T_0}}{\partial \boldsymbol{U}_m^{(T_0)}}\left(\mathcal{W}_*^{(T_0)}\right) \in N(\boldsymbol{U}_{m,*}^{(T_0)}). \tag{41}$$

Hence, (26) holds. ■

Appendix B. Online Updating and Prediction Steps of TOPA-AAW

In this appendix section, we present the detailed steps of TOPA-AAW. The online updating steps are similar to the Algorithm 2, while the prediction step follows the same formula as Algorithm 2.

With the addition of the new parameters in TOPA-AAW, the online updating steps are modified as follows:

Estimate New Core Tensor:

$$\boldsymbol{\mathcal{G}}_{T+1}^{(T)} = \frac{1}{1+\varphi\omega_{T+1}^{(T+1)}}\left(f_{\mathscr{P}^{(T)}}(\mathscr{G}_{[T]}^{(T)}) + \varphi\omega_{T+1}^{(T+1)}\boldsymbol{\mathcal{X}}_{T+1}\prod_{m=1}^{M}\times_m(\boldsymbol{U}_m^{(T)})^H\right).$$

Update Projection Matrices: In $(k+1)$-th iteration, for $m = 1$ to M,

$$\boldsymbol{U}_{m,k+1}^{(T+1)} = \boldsymbol{L}_{m,k+1}(\boldsymbol{R}_{m,k+1})^H,$$

where $\boldsymbol{L}_{m,k+1}$ and $\boldsymbol{R}_{m,k+1}$ are the left and right singular matrices of

$$\sum_{t=T-\tau+2}^{T+1}\omega_t^{(T+1)}\Big(\boldsymbol{\mathcal{X}}_t\prod_{i<m}\times_i(\boldsymbol{U}_{i,k+1}^{(T+1)})^H\prod_{j>m}\times_j(\boldsymbol{U}_{j,k}^{(T+1)})^H\Big)_{(m)} \times \Big(\boldsymbol{\mathcal{G}}_{t,k}^{(T+1)}\Big)_{(m)}^H + \frac{\lambda}{2\varphi}\boldsymbol{U}_{m,k}^{(T+1)},$$

respectively.

Update core tensor series: In $(k+1)$-th iteration, for $t = T-\tau+2$ to $T+1$,

$$\boldsymbol{\mathcal{G}}_{t,k+1}^{(T+1)} = \frac{1}{1+\varphi\omega_t^{(T+1)}+\frac{\lambda}{2}}\Big[f_{\mathscr{P}_{k+1}^{(T+1)}}(\mathscr{G}_{[t-1],k+1}^{(T+1)}) + \varphi\omega_t^{(T+1)}\boldsymbol{\mathcal{X}}_t\prod_{m=1}^{M}\times_m(\boldsymbol{U}_{m,k+1}^{(T+1)})^H + \frac{\lambda}{2}\boldsymbol{\mathcal{G}}_{t,k}^{(T+1)}\Big].$$

References

Fjolla Ademaj, Martin Taranetz, and Markus Rupp. 3GPP 3D MIMO channel model: A holistic implementation guideline for open source simulation tools. *EURASIP Journal on Wireless Communications and Networking*, 2016(1):1–14, 2016.

Dawon Ahn, Seyun Kim, and U. Kang. Accurate online tensor factorization for temporal tensor streams with missing values. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, pages 2822–2826, 2021.

Elynn Y Chen, Dong Xia, Chencheng Cai, and Jianqing Fan. Semi-parametric tensor factor analysis by iteratively projected singular value decomposition. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 86(3):793–823, 2024.

Weitian Chen, Brian D. Anderson, Manfred Deistler, and Alexander Filler. Solutions of Yule-Walker equations for singular AR processes. *Journal of Time Series Analysis*, 32 (5):531–538, 2011.

Xinyu Chen and Lijun Sun. Bayesian temporal factorization for multidimensional time series prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44 (9):4659–4673, 2021.

Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

Jiewen Deng, Jinliang Deng, Renhe Jiang, and Xuan Song. Learning Gaussian mixture representations for tensor time series forecasting. *arXiv preprint arXiv:2306.00390*, 2023.

James D. Hamilton. *Time Series Analysis*. Princeton University Press, 2020.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Stephan Jaeckel, Leszek Raschkowski, Kai Börner, and Lars Thiele. Quadriga: A 3-D multi-cell channel model with time evolution for enabling virtual field trials. *IEEE Transactions on Antennas and Propagation*, 62(6):3242–3256, 2014.

Peiguang Jing, Yuting Su, Xiao Jin, and Chengqian Zhang. High-order temporal correlation model learning for time-series prediction. *IEEE Transactions on Cybernetics*, 49(6):2385–2397, 2018.

Alexander Kaplan and Rainer Tichatschke. Proximal point methods and nonconvex optimization. *Journal of Global Optimization*, 13(1):389–406, 1998.

Hwanjin Kim, Sucheol Kim, Hyeongtaek Lee, Chulhee Jang, Yongyun Choi, and Junil Choi. Massive MIMO channel prediction: Kalman filtering vs. machine learning. *IEEE Transactions on Communications*, 69(1):518–528, 2020.

Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

Linghang Kong, Elynn Chen, Yuzhou Chen, and Yuefeng Han. TEAFormers: Tensor-augmented transformers for multi-dimensional time series forecasting. *arXiv preprint arXiv:2410.20439*, 2024.

Taehyung Kwon, Inkyu Park, Dongjin Lee, and Kijung Shin. Slicenstitch: Continuous CP decomposition of sparse tensor streams. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 816–827. IEEE, 2021.

Khaled Ben Letaief and Ying Jun Zhang. Dynamic multiuser resource allocation and adaptation for wireless systems. *IEEE Wireless Communications*, 13(4):38–47, 2006.

Richard Lewis and Gregory C. Reinsel. Prediction of multivariate time series by autoregressive model fitting. *Journal of Multivariate Analysis*, 16(3):393–411, 1985.

Yaguang Li and Cyrus Shahabi. A brief overview of machine learning methods for short-term traffic forecasting and future directions. *Sigspatial Special*, 10(1):3–9, 2018.

Lihong Liu, Hui Feng, Tao Yang, and Bo Hu. MIMO-OFDM wireless channel prediction by exploiting spatial-temporal correlation. *IEEE Transactions on Wireless Communications*, 13(1):310–319, 2013.

Siqi Liu, Xiaoyu Shi, and Qifeng Liao. Rank-adaptive tensor completion based on Tucker decomposition. *Entropy*, 25(2):225, 2023.

Changqing Luo, Jinlong Ji, Qianlong Wang, Xuhui Chen, and Pan Li. Channel state information prediction for 5G wireless communications: A deep learning approach. *IEEE Transactions on Network Science and Engineering*, 7(1):227–236, 2018.

Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. Traffic flow prediction with big data: A deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873, 2014.

Altti Ilari Maarala, Mika Rautiainen, Miikka Salmi, Susanna Pirttikangas, and Jukka Riekki. Low latency analytics for streaming traffic data with apache spark. In *2015 IEEE International Conference on Big Data*, pages 2855–2858. IEEE, 2015.

Alessandra Pascale and Monica Nicoli. Adaptive bayesian network for traffic flow prediction. In *2011 IEEE Statistical Signal Processing Workshop*, pages 177–180. IEEE, 2011.

R. Tyrrell Rockafellar and Roger J-B Wets. *Variational Analysis*, volume 317. Springer Science & Business Media, 2009.

Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2): 461–464, 1978.

Ashish Sharma, Dinesh Bhuriya, and Upendra Singh. Survey of stock market prediction using machine learning approach. In *2017 International Conference of Electronics, Communication and Aerospace Technology*, pages 506–509. IEEE, 2017.

Qiquan Shi, Jiaming Yin, Jiajun Cai, Andrzej Cichocki, Tatsuya Yokota, Lei Chen, Mingxuan Yuan, and Jia Zeng. Block Hankel tensor ARIMA for multiple short time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5758–5766. AAAI, 2020.

Shun-Yao Shih, Fan-Keng Sun, and Hung-yi Lee. Temporal pattern attention for multivariate time series forecasting. *Machine Learning*, 108(1):1421–1441, 2019.

Yiming Sun, Yang Guo, Charlene Luo, Joel Tropp, and Madeleine Udell. Low-rank Tucker approximation of a tensor from streaming data. *SIAM Journal on Mathematics of Data Science*, 2(4):1123–1150, 2020.

Huachun Tan, Yuankai Wu, Bin Shen, Peter J. Jin, and Bin Ran. Short-term traffic prediction based on dynamic tensor completion. *IEEE Transactions on Intelligent Transportation Systems*, 17(8):2123–2133, 2016.

Di Wang, Yao Zheng, and Guodong Li. High-dimensional low-rank tensor autoregressive time series modeling. *Journal of Econometrics*, 238(1):105544, 2024.

Dong Xia, Anru R Zhang, and Yuchen Zhou. Inference for low-rank tensors—no need to debias. *The Annals of Statistics*, 50(2):1220–1245, 2022.

Chuanfu Xiao and Chao Yang. A rank-adaptive higher-order orthogonal iteration algorithm for truncated Tucker decomposition. *arXiv preprint arXiv:2110.12564*, 2021.

Ke Xu, Elynn Chen, and Yuefeng Han. Statistical inference for low-rank tensor models. *arXiv preprint arXiv:2501.16223*, 2025.

Hsiang-Fu Yu, Nikhil Rao, and Inderjit S. Dhillon. Temporal regularized matrix factorization for high-dimensional time series prediction. *Advances in Neural Information Processing Systems*, 29, 2016.

Eric Zivot and Jiahui Wang. Vector autoregressive models for multivariate time series. *Modeling Financial Time Series with S-PLUS*, pages 369–412, 2006.