A novel tensor tracking algorithm for block-term decomposition of streaming tensors

SUBMISSION DATE / POSTED DATE

30-04-2023 / 03-05-2023

CITATION

Thanh, Le Trung; abed-meraim, karim; Ravier, Philippe; Buttelli, Olivier (2023): A novel tensor tracking algorithm for block-term decomposition of streaming tensors. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.22723453.v1

DOI

10.36227/techrxiv.22723453.v1

A NOVEL TENSOR TRACKING ALGORITHM FOR BLOCK-TERM DECOMPOSITION OF STREAMING TENSORS

Le Trung Thanh[⋆,†]*, Karim Abed-Meraim*[⋆]*, Philippe Ravier*[⋆]*, Olivier Buttelli*[⋆]

[⋆] University of Orléans, PRISME Laboratory, France

[†] VNU Univeristy of Engineering and Technology, AVITECH Institute, Vietnam

{trung-thanh.le, karim.abed-meraim, philippe.ravier, olivier.buttelli}@univ-orleans.fr

ABSTRACT

Block-term decomposition (BTD), which factorizes a tensor (aka a multiway array) into block components of low rank, has been a powerful processing tool for multivariate and high-dimensional data analysis. In this paper, we propose a novel tensor tracking method called SBTD for factorizing tensors derived from multidimensional data streams under the BTD format. Thanks to the alternating optimization framework, SBTD first applies a regularized least-squares solver to estimate the temporal factor of the underlying streaming tensor. Then, SBTD adopts an adaptive filter to track the non-temporal tensor factors over time by minimizing a weighted least-squares cost function. Numerical experiments indicate that SBTD is capable of tensor tracking with competitive performance compared to the state-of-the-art BTD algorithms.

***Index Terms*—** Tensor tracking, block-term decomposition, adaptive algorithms, data streams.

1. INTRODUCTION

Tensor tracking or streaming tensor decomposition has gradually gained popularity as many modern applications generate a huge number of data streams over the years [1]. Factorizing streaming tensors is however nontrivial and it is much different from batch tensor decomposition due to several inherent computational issues of stream processing [2]. For example, modern streaming datasets are often associated with high velocity and low veracity. High velocity requires (near) real-time processing, while low veracity demands robust algorithms to better deal with noisy and inconsistent data. Moreover, data acquisition is often a time-varying process where properties of data can change with time. These characteristics hinder the use of conventional tensor decomposition methods to streaming tensors as well as results in distinguishing requirements for tensor trackers, such as low latency, high scalability and adaptability, to name a few [1].

Block-term decomposition (BTD), which unifies the most two well-known and widely-used tensor decompositions (i.e., canonical polyadic and Tucker), has recently attracted much attention in the signal processing community [3, 4]. Particularly under the BTD format, a tensor (aka a multiway array) is factorized into several low-rank block terms/components. In this paper, we focus on investigating how to factorize a streaming tensor under a special form of BTD that expresses the underlying tensor as a sum of R rank-$(L_r, L_r, 1)$ block terms, see Fig. 1 for an illustration. This decomposition offers several appealing properties, such as uniqueness, stability, and interpretability [3, 5]. Accordingly, it has already found in many applications, such as blind source separation/system identification [6, 7], biomedical engineering [8, 9], and remote sensing [10, 11].

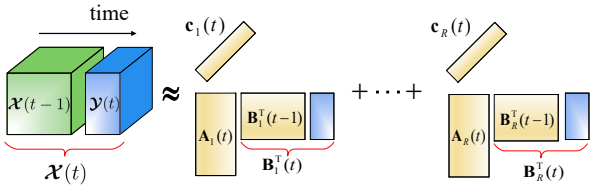

Fig. 1: Streaming rank-$(L_r, L_r, 1)$ BTD.

In the tensor literature, many online/adaptive/streaming algorithms have been developed for tensor tracking (see [1] for a comprehensive survey). Among them, O-BTD-RLS [12] and OnlineBTD [13] are capable of streaming BTD. Particularly, O-BTD-RLS adopts an iterative reweighted least-squares solver to track the underlying rank-$(L_r, L_r, 1)$ block terms of streaming tensors. The tracker promotes sparsity of tensor factors by using a sum of their $\ell_{1,2}$-norm as a regularization. O-BTD-RLS also has the potential to reveal the number of block terms and their rank with time. However, its design is specifically optimized for tracking tensors of which the last mode increases in size over time that limits its applications in practice.[1] OnlineBTD, on the other hand, uses the stochastic gradient descent (SGD) method to incrementally estimate the low rank-(N, M, P) block terms. To accelerate the online processing, it adopts (i) a fast matricized tensor times Kronecker product, (ii) an efficient pseudo-inverse operator based on LU factorization, and (iii) a dynamic programming strategy to avoid the duplicated Kronecker products. Similar to O-BTD-RLS, OnlineBTD works under the assumption that the last tensor factor evolves over time. However, its performance (i.e., estimation accuracy and convergence behavior) is not always good especially in noisy and nonstationary environments, see Fig. 4 for an ex-

[1] In applications of tensor-based blind source separation or system identification for example, the mixing matrix of a fixed size is often recast into the last tensor factor of the corresponding rank-$(L_r, L_r, 1)$ BTD, see [6, 7, 14].

ample. In parallel, there also exist several other methods for factorizing tensors under the BTD format, such as BTD-ALS [5], BTD-AGL [9], BTD-HIRLS [4], BBTD [15], and NBTD-HIRLS [16]. However, all of the latter methods work in batch mode which means they operate and process over all or most of the stored data at once. Accordingly, they may not be suitable for streaming situations where data streams are continuously generated and collected over time. To overcome these drawbacks above, we propose in this study a new efficient and effective adaptive BTD method which exhibits competitive tracking performance even in noisy and time-varying environments.

Notations: Lowercase, boldface lowercase, boldface capital, and bold calligraphic letters denote scalars (e.g., x), vectors (e.g., $\mathbf{x}$), matrices (e.g., $\mathbf{X}$), and tensors (e.g., $\boldsymbol{\mathcal{X}}$), respectively. Symbols $\circ$, $\otimes$, $\odot$, $\boxplus$ denote the outer product, Hadamard product, Khatri-Rao product, and tensor concatenation, respectively. Rank-$(L_r, L_r, 1)$ BTD decomposition is denoted by $[\![.]\!]$. We denote by $\|.\|_F$ the Frobenius norm.

2. PROBLEM FORMULATION

2.1. Rank-$(L_r, L_r, 1)$ BTD

Consider a 3rd-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I\times J\times K}$ in batch setting. Its rank-$(L_r, L_r, 1)$ BTD factorization is expressed by [3]

$$\boldsymbol{\mathcal{X}} = \sum_{r=1}^{R} \mathbf{H}_r \circ \mathbf{c}_r = \sum_{r=1}^{R} \left(\mathbf{A}_r \mathbf{B}_r^\top\right) \circ \mathbf{c}_r, \tag{1}$$

where R is the number of block terms, $\mathbf{H}_r = \mathbf{A}_r\mathbf{B}_r^\top \in \mathbb{R}^{I\times J}$ is a low-rank matrix with 2 factors $\mathbf{A}_r \in \mathbb{R}^{I\times L_r}$ and $\mathbf{B}_r \in \mathbb{R}^{J\times L_r}$, and $\mathbf{c}_r \in \mathbb{R}^{K\times 1}$ is a non-zero vector. Let $\mathbf{A} = [\mathbf{A}_1, \ldots, \mathbf{A}_R]$, $\mathbf{B} = [\mathbf{B}_1, \ldots, \mathbf{B}_R]$, and $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_R]$. For short, we denote by $\boldsymbol{\mathcal{X}} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ the BTD model (1).

As the model (1) is trilinear with respect to $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, the "workhorse" approach for computing BTD is the alternating least-squares (ALS) method [5]. Like CP/PARAFAC, the BTD in (1) is essentially unique under mild conditions [3] which is a useful property for several applications, such as blind source separation [14].

Lemma 1 (Theorem 4.1 in [3])**.** *When* $\mathbf{A}$ *and* $\mathbf{B}$ *are full column rank, and* $\mathbf{C}$ *does not have proportional columns,* (1) *is essentially unique in the sense that there are only two indeterminacies of* $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$*, including the order of* R *blocks and the scale of* $\mathbf{H}_r$ *and* $\mathbf{c}_r$*.*

The full column rank condition in Lemma 1 implies that $\min(I, J) \geq \sum_{r=1}^{R} L_r$ which is easy to verify and achievable in practice when L_r and R are relatively small.

2.2. Streaming Rank-$(L_r, L_r, 1)$ BTD

In this work, we consider a streaming tensor $\boldsymbol{\mathcal{X}}(t) \in \mathbb{R}^{I\times J(t)\times K}$ of which the mode 2 evolves with time (i.e., I and K are fixed while $J(t)$ is increasing over time). Suppose that at time t, we collect a block of data $\boldsymbol{\mathcal{Y}}(t) \in \mathbb{R}^{I\times W(t)\times K}$ with $W(t) \geq 1$. Hereby, the underlying tensor $\boldsymbol{\mathcal{X}}(t)$ is obtained by appending $\boldsymbol{\mathcal{Y}}(t)$ to old data $\boldsymbol{\mathcal{X}}(t-1)$, i.e., $\boldsymbol{\mathcal{X}}(t) = \boldsymbol{\mathcal{X}}(t-1) \boxplus \boldsymbol{\mathcal{Y}}(t)$ and $J(t) = J(t-1) + W(t)$, see Fig. 1 for an illustration.

Thanks to (1), we can express $\boldsymbol{\mathcal{X}}(t)$ and its block $\boldsymbol{\mathcal{Y}}(t)$ under the rank-$(L_r, L_r, 1)$ BTD format as

$$\boldsymbol{\mathcal{X}}(t) = [\![\mathbf{A}(t), \mathbf{B}(t), \mathbf{C}(t)]\!], \tag{2}$$

$$\boldsymbol{\mathcal{Y}}(t) = [\![\mathbf{A}(t), \underline{\mathbf{B}}(t), \mathbf{C}(t)]\!], \tag{3}$$

where $\underline{\mathbf{B}}(t) = [\underline{\mathbf{B}}_1(t), \underline{\mathbf{B}}_2(t), \ldots, \underline{\mathbf{B}}_R(t)]$ is the t-th block of rows of $\mathbf{B}(t) \in \mathbb{R}^{J(t)\times\sum_{r=1}^{R} L_r}$. The problem of tensor tracking under the rank-$(L_r, L_r, 1)$ BTD format is stated as follows.

> **Problem**: At time t, given a block of new data $\boldsymbol{\mathcal{Y}}(t)$ and old estimate of $\boldsymbol{\mathcal{X}}(t-1) = [\![\mathbf{A}(t-1), \mathbf{B}(t-1), \mathbf{C}(t-1)]\!]$, we want to incrementally estimate $\mathbf{A}(t)$, $\mathbf{B}(t)$, and $\mathbf{C}(t)$ of $\boldsymbol{\mathcal{X}}(t) = \boldsymbol{\mathcal{X}}(t-1) \boxplus \boldsymbol{\mathcal{Y}}(t)$ in time.

Our main goal is to develop an efficient adaptive algorithm for tracking $\mathbf{A}(t)$, $\mathbf{B}(t)$ and $\mathbf{C}(t)$ effectively. To support our development in Section III, we suppose that the BTD model (2) is either fixed or changing slowly over time, and the number of blocks R and their rank-$(L_r, L_r, 1)$ are available.

3. PROPOSED METHOD

In this section, we introduce a novel tensor tracking algorithm for tracking the underlying low rank-$(L_r, L_r, 1)$ BTD of streaming tensors, called streaming BTD (SBTD). Thanks to the alternating optimization framework, we first update $\mathbf{B}(t)$, given old estimations of $\mathbf{A}(t-1)$ and $\mathbf{C}(t-1)$, then estimate $\mathbf{A}(t)$ and $\mathbf{C}(t)$ given $\mathbf{A}(t-1), \mathbf{C}(t-1)$ and $\mathbf{B}(t)$. In what follows, we describe the way how to track these factors over time. Complexity analysis of SBTD is then provided.

3.1. Estimation of $\mathbf{B}(t)$

Generally, $\mathbf{B}(t)$ is obtained by solving the following problem

$$\mathbf{B}(t) = \underset{\mathbf{B}}{\operatorname{argmin}} \left\| \boldsymbol{\mathcal{X}}(t) - [\![\mathbf{A}(t-1), \mathbf{B}, \mathbf{C}(t-1)]\!] \right\|_F^2. \tag{4}$$

As the size of $\mathbf{B}(t)$ increases with time, minimizing (4) directly is possible but turns out to be inefficient when t is large.

Here, we exploit the fact that during the tracking process, the two factors of fixed size $\mathbf{A}$ and $\mathbf{C}$ do not change much between two consecutive time instances, i.e., $\mathbf{A}(t) \approxeq \mathbf{A}(t-1)$ and $\mathbf{C}(t) \approxeq \mathbf{C}(t-1)$ for almost t. Accordingly, we obtain

$$\begin{aligned}
\boldsymbol{\mathcal{X}}(t) &= \boldsymbol{\mathcal{X}}(t-1) \boxplus \boldsymbol{\mathcal{Y}}(t) \\
&= [\![\mathbf{A}(t-1), \mathbf{B}(t-1), \mathbf{C}(t-1)]\!] \boxplus [\![\mathbf{A}(t), \underline{\mathbf{B}}(t), \mathbf{C}(t)]\!] \\
&\approxeq [\![\mathbf{A}(t), [\mathbf{B}(t-1)^\top \mid \mathbf{B}(t)^\top]^\top, \mathbf{C}(t)]\!].
\end{aligned} \tag{5}$$

In parallel, when $\mathbf{B}(t-1)$ satisfies the uniqueness condition of BTD in Lemma 1 (i.e., full rank), adding a block of row vectors to $\mathbf{B}(t-1)$ results in a new matrix which still satisfies this condition, thanks to Lemma 2.

Lemma 2. *Given a matrix* $\mathbf{Z} \in \mathbb{R}^{n\times m}$ *of rank* $r \le \min(m,n)$. *Adding any row vector* $\mathbf{b}$ *to* $\mathbf{Z}$ *does not decrease its rank*

$$\operatorname{rank}\left(\left[\mathbf{Z}^\top \mid \mathbf{b}^\top\right]^\top\right) \ge r \quad \forall \mathbf{b} \in \mathbb{R}^{1\times m}. \tag{6}$$

Proof. It is a corollary of Cauchy interlace theorem [17]. □

Together with (5), we can approximate $\mathbf{B}(t)$ as follows

$$\mathbf{B}(t) \approx \left[\mathbf{B}_P(t-1)^\top \mid \underline{\mathbf{B}}_P(t)^\top\right]^\top. \tag{7}$$

Here, $(.)_P$ denotes the permutation operation according to $\mathbf{A}(t)$ and $\mathbf{C}(t)$ to be presented later in Sec. 3.2 and the block $\underline{\mathbf{B}}(t)$ is derived from the following minimization

$$\underline{\mathbf{B}}(t) = \underset{\underline{\mathbf{B}}}{\operatorname{argmin}} \left\|\boldsymbol{\mathcal{Y}}(t) - \left[\!\left[\mathbf{A}(t-1), \underline{\mathbf{B}}, \mathbf{C}(t-1)\right]\!\right]\right\|_F^2. \tag{8}$$

To solve (8), we can recast it into the following form

$$\underline{\mathbf{B}}(t) = \underset{\underline{\mathbf{B}}}{\operatorname{argmin}} \left\|\mathbf{Y}^{(2)}(t)^\top - \mathbf{H}_B(t)\underline{\mathbf{B}}^\top\right\|_F^2, \tag{9}$$

where $\mathbf{Y}^{(2)}(t)$ is the mode-2 unfolding matrix of $\boldsymbol{\mathcal{Y}}(t)$ and

$$\mathbf{H}_B(t) = \left[\mathbf{c}_1(t-1) \otimes \mathbf{A}_1(t-1) \middle| \ \ldots \right. \\ \left. \ldots \ \middle| \mathbf{c}_R(t-1) \otimes \mathbf{A}_R(t-1)\right]. \tag{10}$$

The minimizer of (9) is then obtained by applying a regularized least-squares (LS) solver

$$\underline{\mathbf{B}}(t) = \left[\left(\mathbf{H}_B(t)^\top \mathbf{H}_B(t) + \alpha \mathbf{I}\right)^{-1} \mathbf{H}_B(t)^\top \mathbf{Y}^{(2)}(t)^\top\right]^\top, \tag{11}$$

where the inclusion of a small regularization parameter $\alpha > 0$ is to avoid pathological cases in practice. Note that, as $\mathbf{H}_B(t)$ is of Kronecker structure, we can apply the randomized LS method to speed up the computation of $\underline{\mathbf{B}}(t)$ when dealing with large-scale tensors. See our companion works on streaming CP and Tucker decompositions in [18, 19] for examples.

3.2. Estimation of $\mathbf{A}(t)$ and $\mathbf{C}(t)$

Given $\mathbf{B}(t)$ and old estimates $\mathbf{A}(t-1)$ and $\mathbf{C}(t-1)$, we update $\mathbf{A}(t)$ and $\mathbf{C}(t)$ by solving the following minimization

$$\{\mathbf{A}(t), \mathbf{C}(t)\} = \underset{\mathbf{A},\mathbf{C}}{\operatorname{argmin}} \sum_{\tau=1}^{t} \beta^{t-\tau} \left\|\boldsymbol{\mathcal{Y}}(\tau) - \left[\!\left[\mathbf{A}, \underline{\mathbf{B}}(\tau), \mathbf{C}\right]\!\right]\right\|_F^2 \\ + \frac{\rho}{2}\left(\|\mathbf{A}\|_F^2 + \|\mathbf{C}\|_F^2\right), \tag{12}$$

where $0 < \beta \le 1$ is a forgetting factor aiming to reduce the effect of old observations and $\rho > 0$ is a small regularization parameter. Here, the first term of (12) measures the residual error between the observed value and the estimated value of data blocks, while the second term of (12) is for avoiding pathological cases in practice.

As both terms of (12) are convex, the alternating optimization framework can be useful for updating $\mathbf{A}(t)$ and $\mathbf{C}(t)$:

$$\min_{\mathbf{A}} \sum_{\tau=1}^{t} \beta^{t-\tau} \left\|\mathbf{Y}^{(1)}(\tau) - \mathbf{A}\mathbf{W}_A(\tau)\right\|_F^2 + \frac{\rho}{2}\|\mathbf{A}\|_F^2, \tag{13}$$

$$\min_{\mathbf{C}} \sum_{\tau=1}^{t} \beta^{t-\tau} \left\|\mathbf{Y}^{(3)}(\tau) - \mathbf{C}\mathbf{W}_C(\tau)\right\|_F^2 + \frac{\rho}{2}\|\mathbf{C}\|_F^2, \tag{14}$$

where $\mathbf{W}_A(\tau)$ and $\mathbf{W}_C(\tau)$ are defined as

$$\mathbf{W}_A(\tau) = \left[\mathbf{c}_1(t-1) \otimes \underline{\mathbf{B}}_1(\tau) \middle| \ldots \right. \\ \left. \ldots \ \middle| \mathbf{c}_R(t-1) \otimes \underline{\mathbf{B}}_R(\tau)\right]^\top, \tag{15}$$

$$\mathbf{W}_C(\tau) = \left[\left(\mathbf{A}_1(t-1) \odot \underline{\mathbf{B}}_1(\tau)\right)\mathbf{1}_{L_1} \middle| \ldots \right. \\ \left. \ldots \ \middle| \left(\mathbf{A}_R(t-1) \odot \underline{\mathbf{B}}_R(\tau)\right)\mathbf{1}_{L_R}\right]^\top, \tag{16}$$

with $\mathbf{1}_L = [1, 1, \ldots, 1]^\top$ of size $L \times 1$. In particular, taking the natural gradient of (13) and (14) to zero, $\mathbf{A}(t)$ and $\mathbf{C}(t)$ can be estimated efficiently over time. Due to the space limitation, we present the update rule of $\mathbf{A}(t)$ only and omit its derivation for brevity:

$$\mathbf{A}(t) = \mathbf{D}_A(t)\left(\mathbf{S}_A(t) + \rho\mathbf{I}\right)^{-1}, \tag{17}$$

where $\mathbf{D}_A(t)$ and $\mathbf{S}_A(t)$ are recursively updated as follows

$$\mathbf{D}_A(t) = \beta\mathbf{D}_A(t-1) + \mathbf{Y}^{(1)}(t)\mathbf{W}_A(t), \tag{18}$$

$$\mathbf{S}_A(t) = \beta\mathbf{S}_A(t-1) + \mathbf{W}_A(t)^\top\mathbf{W}_A(t). \tag{19}$$

The update rule (17) is inexpensive as it uses simple matrix multiplications and additions, and an inverse operation of a small size matrix $\mathbf{S}_A(t) + \rho\mathbf{I}$ (i.e., its size is $(\sum_{r=1}^R L_r) \times (\sum_{r=1}^R L_r)$ independent of data dimension, so its inverse is not expensive). At $t = 0$, we set $\mathbf{D}_A(0)$ and $\mathbf{S}_A(0)$ to zeros.

We update $\mathbf{C}(t)$ in the same way as $\mathbf{A}(t)$.

3.3. Computational complexity

Let $D = IK$, $S = \sum_{r=1}^R L_r$, and assume $W(t) = W$ for all t. At time t, the estimation of $\underline{\mathbf{B}}(t)$ requires $\mathcal{O}(DS + WDS^2)$ flops. The update of $\mathbf{A}(t)$ comes from the computation of $\mathbf{W}_A(t)$, $\mathbf{D}_A(t)$, and $\mathbf{S}_A(t)$ which costs $\mathcal{O}(W(KS + DS + KS^2))$ flops in total. Similarly, the computation of $\mathbf{C}(t)$ costs $\mathcal{O}(W(IS + DR + IR^2))$ flops. As $D > I + K$ and $S > R$, the overall complexity of SBTD is $\mathcal{O}(WDS^2)$ flops.

4. NUMERICAL EXPERIMENTS

In this section, we conduct some simulations to demonstrate the tracking ability of SBTD w.r.t. three aspects: (i) effect of random noise on its performance, (ii) its effectiveness in time-varying environments, and (iii) performance comparison with the state-of-the-art adaptive BTD algorithms.

Experiment setup: At each time t, $\boldsymbol{\mathcal{Y}}(t)$ is generated under the following model

$$\boldsymbol{\mathcal{Y}}(t) = \left[\!\left[\mathbf{A}(t), \underline{\mathbf{B}}(t), \mathbf{C}(t)\right]\!\right] + \sigma_n \boldsymbol{\mathcal{N}}(t) \ \in \mathbb{R}^{I\times W\times K}. \tag{20}$$

Here, $\underline{\mathbf{B}}(t) \in \mathbb{R}^{W\times\sum_{r=1}^R L_r}$ is a Gaussian matrix of zero-mean and unit-variance entries, $\boldsymbol{\mathcal{N}}(t) \in \mathbb{R}^{I\times W\times K}$ is a noise tensor whose entries are i.i.d. from $\mathcal{N}(0,1)$. The two factors $\mathbf{A}(t)$ and $\mathbf{C}(t)$ are varied as follows

$$\mathbf{A}(t) = \mathbf{A}(t-1) + \varepsilon\mathbf{N}_A(t) \ \in \mathbb{R}^{I\times\sum_{r=1}^R L_r}, \tag{21}$$

$$\mathbf{C}(t) = \mathbf{C}(t-1) + \varepsilon\mathbf{N}_C(t) \ \in \mathbb{R}^{K\times R}, \tag{22}$$

where ε is a time-varying factor, $\mathbf{N}_A(t)$ and $\mathbf{N}_C(t)$ are two matrices of standard normal random Gaussian noises.

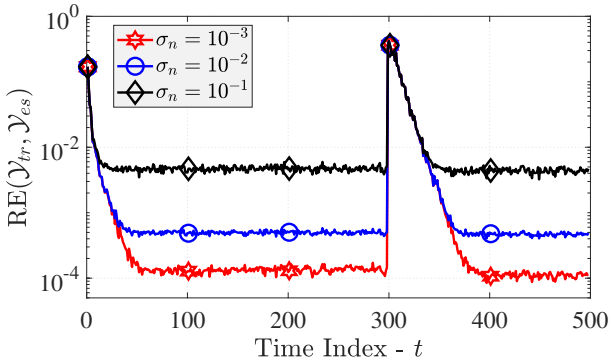

Fig. 2: Tracking ability of SBTD in noisy environments – effect of the noise level σ_n.

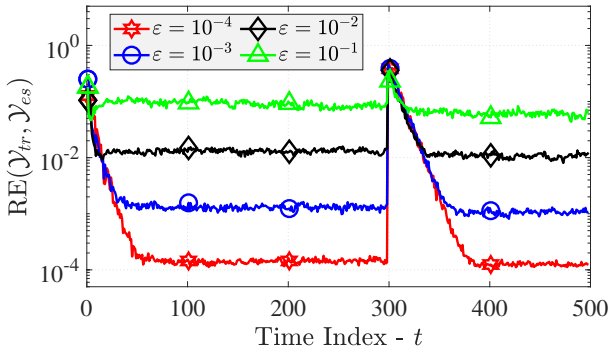

Fig. 3: Tracking ability of SBTD in time-varying environments – effect of the time-varying factor ε.

To measure the tracking ability of algorithms, we use the following evaluation metric

$$\mathrm{RE}(\mathcal{Y}_{tr}, \mathcal{Y}_{es}) = \|\mathcal{Y}_{tr} - \mathcal{Y}_{es}\|_F / \|\mathcal{Y}_{tr}\|_F, \qquad (23)$$

where $\mathcal{Y}_{tr}$ (resp. $\mathcal{Y}_{es}$) refers to the ground truth (resp. estimate).

Noisy Environments: By varying the value of σ_n, we can evaluate the effect of noise on the performance of SBTD. We select the noise level σ_n among $\{10^{-1}, 10^{-2}, 10^{-3}\}$ and then measure the performance of SBTD on a streaming tensor of size $20 \times 2500 \times 30$ with three block terms ($R = 3$). Particularly, these blocks are composed of factors of rank $L_1 = 2$, $L_2 = 3$, and $L_3 = 4$, respectively. At each time t, we collect and process a data batch of size $W = 5$. The time-varying factor is set to $\varepsilon = 10^{-4}$ and an abrupt change is specifically created at $t = 300$ to evaluate how fast SBTD converges. We fix the forgetting factor and two regularization parameters respectively at $\beta = 0.9$ and $\alpha = \rho = 10^{-4}$ in all testing cases.

The experimental results are illustrated in Fig. 2. We can see that SBTD is capable of tracking the underlying rank-$(L_r, L_r, 1)$ BTD of streaming tensors in noisy environments. Indeed, the noise level σ_n does not affect the convergence rate of SBTD but its steady-state error. The lower the value of σ_n is, the better the estimation accuracy SBTD achieves.

Nonstationary Environments: Next, we investigate the tracking ability of SBTD in dynamic environments by varying the time-varying factor ε in the range $[10^{-4}, 10^{-1}]$. Here, we reuse the same experiment setup as in the previous task, except the noise level which is fixed at $\sigma_n = 10^{-3}$. Similar to the effect of noise, the convergence rate of SBTD is not much affected by ε but its estimation accuracy only, see Fig 3.

Performance Comparison: Finally, we compare SBTD with OnlineBTD [13] and BTD-ALS [5].[2] As OnlineBTD requires a training set of data samples, 10% of temporal slices (data streams) are used to initialize its warm start. Meanwhile, BTD-ALS is a batch BTD algorithm that usually processes data in batches/blocks. Thus, we define a window of 20 successive tensor slices as a batch and use old estimation of factors as a starting point for BTD-ALS at time t. In this task, we still use the same synthetic tensor above but with 1500 temporal slices, i.e., we consider the first 300 time instances. The noise level and time-varying factor are set to $\sigma_n = 10^{-3}$ and $\varepsilon = 10^{-4}$, respectively. As can be seen from Fig. 4, our algorithm outperforms OnlineBTD and BTD-ALS in this context. The tracking error of OnlineBTD seems unstable and fluctuates widely, while that of BTD-ALS tends to increase as time passes. Only SBTD converges to steady-state error.

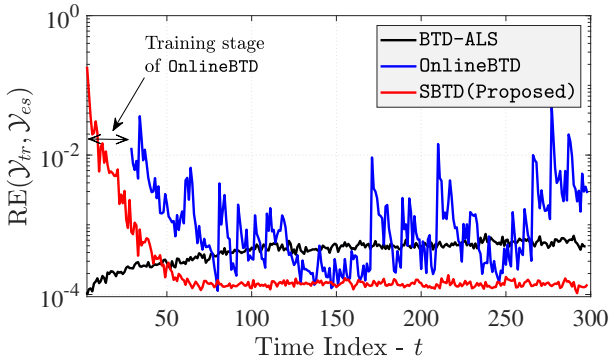

Fig. 4: Performance comparison among BTD algorithms.

5. CONCLUSIONS

In this paper, we have addressed the problem of streaming block-term decomposition. A novel adaptive algorithm called SBTD has been proposed to incrementally estimate the underlying rank-$(L_r, L_r, 1)$ block terms of streaming tensors over time. Experimental results show that SBTD is capable of tensor tracking in both noisy and time-varying environments with high estimation accuracy, and it outperforms other adaptive BTD algorithms. In future works, it would be of great interest to further extend SBTD for tracking the number of blocks and their rank with time. Another interesting direction would be to develop a robust variant of SBTD for dealing with data corruptions and missing observations.

[2] As O-BTD-RLS [12] is not designed for dealing with streaming tensors of which the second mode evolves with time, we omit it here since we cannot have a fair performance comparison between SBTD and O-BTD-RLS.

6. REFERENCES

[1] L. T. Thanh, K. Abed-Meraim, N. L. Trung, and A. Hafiane, "A contemporary and comprehensive survey on streaming tensor decomposition," *IEEE Trans. Knowl. Data Eng.*, 2022, early access, doi: 10.1109/TKDE.2022.3230874.

[2] T. Akidau, S. Chernyak, and R. Lax, *Streaming Systems: The What, Where, When, and How of Large-Scale Data Processing*. O'Reilly Media, Inc, 2018.

[3] L. De Lathauwer, "Decompositions of a higher-order tensor in block terms – Part II: Definitions and uniqueness," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 3, pp. 1033–1066, 2008.

[4] A. A. Rontogiannis, E. Kofidis, and P. V. Giampouras, "Block-term tensor decomposition: Model selection and computation," *IEEE J. Sel. Topics Signal Process.*, vol. 15, no. 3, pp. 464–475, 2021.

[5] L. De Lathauwer and D. Nion, "Decompositions of a higher-order tensor in block terms – Part III: Alternating least squares algorithms," *SIAM J. Matrix Anal. Appl.*, vol. 30, pp. 1067–1083, 2008.

[6] M. Sørensen, F. Van Eeghem, and L. De Lathauwer, "Blind multichannel deconvolution and convolutive extensions of canonical polyadic and block term decompositions," *IEEE Trans. Signal Process.*, vol. 65, no. 15, pp. 4132–4145, 2017.

[7] N. Govindarajan, E. N. Epperly, and L. De Lathauwer, "(L_r,L_r,1)-decompositions, sparse component analysis, and the blind separation of sums of exponentials," *SIAM J. Matrix Anal. Appl.*, vol. 43, no. 2, pp. 912–938, 2022.

[8] B. Hunyadi, D. Camps, L. Sorber, W. V. Paesschen, M. D. Vos, S. V. Huffel, and L. D. Lathauwer, "Block term decomposition for modelling epileptic seizures," *EURASIP J. Adv. Signal Process.*, vol. 2014, no. 1, pp. 1–19, 2014.

[9] J. H. d. M. Goulart, P. M. R. de Oliveira, R. C. Farias, V. Zarzoso, and P. Comon, "Alternating group lasso for block-term tensor decomposition and application to ECG source separation," *IEEE Trans. Signal Process.*, vol. 68, pp. 2682–2696, 2020.

[10] F. Xiong, J. Zhou, and Y. Qian, "Hyperspectral restoration via L_0 gradient regularized low-rank tensor factorization," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 12, pp. 10 410–10 425, 2019.

[11] M. Ding, X. Fu, T.-Z. Huang, J. Wang, and X.-L. Zhao, "Hyperspectral super-resolution via interpretable block-term tensor modeling," *IEEE J. Sel. Top. Signal Process.*, vol. 15, no. 3, pp. 641–656, 2020.

[12] A. A. Rontogiannis, E. Kofidis, and P. V. Giampouras, "Online rank-revealing block-term tensor decomposition," in *Proc. Asilomar Conf. Signals Syst. Comput.*, 2021, pp. 1678–1682.

[13] E. Gujral and E. E. Papalexakis, "OnlineBTD: Streaming algorithms to track the block term decomposition of large tensors," in *Proc. IEEE Int. Conf. Data Sci. Adv. Anal.*, 2020, pp. 168–177.

[14] L. De Lathauwer, "Blind separation of exponential polynomials and the decomposition of a tensor in rank-(L_r,L_r,1) terms," *SIAM J. Matrix Anal. Appl.*, vol. 32, no. 4, pp. 1451–1474, 2011.

[15] P. Giampouras, A. A. Rontogiannis, and E. Kofidis, "Block-term tensor decomposition model selection and computation: The Bayesian way," *IEEE Trans. Signal Process.*, vol. 70, pp. 1704–1717, 2022.

[16] E. Kofidis, P. V. Giampouras, and A. A. Rontogiannis, "A projected Newton-type algorithm for rank-revealing nonnegative block-term tensor decomposition," in *Proc. Eur. Signal Process. Conf.*, 2022, pp. 1961–1965.

[17] B. N. Parlett, *The Symmetric Eigenvalue Problem*, ser. Classics in Applied Mathematics. SIAM, 1998.

[18] L. T. Thanh, K. Abed-Meraim, N. L. Trung, and A. Hafiane, "A fast randomized adaptive CP decomposition for streaming tensors," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2021, pp. 2910–2914.

[19] L. T. Thanh, K. Abed-Meraim, N. L. Trung, and A. Hafiane, "Tracking online low-rank approximations of incomplete high-order streaming tensors," *Elsevier Patterns*, 2023, to appear, doi: 10.36227/techrxiv.19704034.