Adaptive Algorithms for Tracking Tensor-Train Decomposition of Streaming Tensors

Le Trung Thanh, Karim Abed-Meraim, Nguyen Linh-Trung, Remy Boyer

Adaptive Algorithms for Tracking Tensor-Train Decomposition of Streaming Tensors

Conference Paper · June 2020

CITATIONS
0

READS
14

4 authors:

Le Trung Thanh
Université d'Orléans
11 PUBLICATIONS **12** CITATIONS
SEE PROFILE

Karim abed-meraim
Université d'Orléans
416 PUBLICATIONS **9,736** CITATIONS
SEE PROFILE

Nguyen Linh-Trung
Vietnam National University, Hanoi
61 PUBLICATIONS **733** CITATIONS
SEE PROFILE

Remy Boyer
Université de Lille
179 PUBLICATIONS **1,193** CITATIONS
SEE PROFILE

Adaptive Algorithms for Tracking Tensor-Train Decomposition of Streaming Tensors

Le Trung Thanh[1,2], Karim Abed-Meraim[1], Nguyen Linh Trung[2] and Remy Boyer[3]
[1]PRISME Laboratory, University of Orléans, Orléans, France
[2]AVITECH Institute, VNU University of Engineering and Technology, Hanoi, Vietnam
[3]CRIStAL, Université de Lille, Villeneuve d'Ascq, France

***Abstract*—Tensor-train (TT) decomposition has been an efficient tool to find low order approximation of large-scale, high-order tensors. Existing TT decomposition algorithms are either of high computational complexity or operating in batch-mode, hence quite inefficient for (near) real-time processing. In this paper, we propose a novel adaptive algorithm for TT decomposition of streaming tensors whose slices are serially acquired over time. By leveraging the alternating minimization framework, our estimator minimizes an exponentially weighted least-squares cost function in an efficient way. The proposed method can yield an estimation accuracy very close to the error bound. Numerical experiments show that the proposed algorithm is capable of adaptive TT decomposition with a competitive performance evaluation on both synthetic and real data.**

***Index Terms*—Tensor-train decomposition, adaptive algorithms, streaming tensors.**

I. Introduction

Tensor decomposition has been emerging as a new data analytic tool to discover new valuable information hidden in datasets [1]. Tensor is a multidimensional array, and it provides a natural representation for multivariate, high-dimensional data. Hence, tensor decomposition has rapidly found many applications in signal processing and machine learning problems [2].

Most well-known and widely-used tensor decompositions are CANDECOMP/PARAFAC (CP) decomposition [3] and Tucker decomposition [4] which are two extensions of singular value decomposition (SVD) for tensors. Under these two models, a tensor is factorized into a sequence of factor matrices acting on a reduced size core tensor [1]. As a result, they offer several advantages for data refinement, e.g. CP decomposition only costs a linear storage complexity w.r.t the order of tensors, while Tucker decomposition provides a practical low multilinear rank approximation for tensors. However, CP decomposition easily becomes ill-conditioned unless additional structural constraints apply. Tucker decomposition is stable, but its storage complexity grows exponentially with the order of tensors; this is the "curse of dimensionality". To alleviate their drawbacks, tensor-train (TT) decomposition, which is a special case of Hierarchical Tucker decomposition, has been introduced as a good alternative [5], [6].

TT decomposition is a factorization of a tensor into a multilinear product of 3-way tensors. Note that, TT decomposition of 3-way tensors is a constrained Tucker model called Tucker-2 [7]. TT decomposition provides a memory-saving representation for high-order tensors which costs only $\mathcal{O}(nIr^2)$ memory storage to represent a n-way tensor of size $I \times I \times \cdots \times I$ and rank r. Similar to Tucker decomposition, TT decomposition allows to decompose any tensor under a predefined error [5]. Moreover, on the TT model, basic mathematical operations can be efficiently performed to determine TT decomposition in a stable way, e.g. TT-SVD algorithm performs a sequence of SVD decompositions to unfolding matrices of the tensor [5], [8]. Although TT decomposition offers an efficient approach to represent high-order tensors, most existing TT algorithms are either of high computational complexity or operating in batch-mode, hence become inefficient for (near) real-time processing. In many applications, data acquisition is a time-varying process where data are serially observed or slowly changing with time. This leads to two essential issues: (i) problem of growing in size of tensors over time and (ii) problem of time-dependent low order approximation for dynamic sensing of a tensor. Therefore, there is a great interest in developing adaptive (online) TT decomposition algorithms to deal with this scenario.

In the literature, there are a several algorithms related to adaptive TT decomposition. Lubich *et al.* have proposed a dynamical tensor approximation-based TT format to account for time-dependent tensors acquired by dynamical systems [9], [10]. Inspired by the Dirac–Frenkel time-dependent variational principle, a projector-splitting integrator was applied to track the variation of such tensors over time. Note that, the time-dependent tensors are of fixed size, i.e., Lubich's method works in a batch setting. Liu *et al.* have introduced an incremental TT decomposition algorithm, namely iTTD, for decomposing tensors whose slices increase over time [11]. iTTD consists of two phases: (i) factorizing the newcoming tensor derived from new observations into TT-cores and (ii) appending the resulting TT-cores to past estimated TT-cores to form the new bigger TT-cores. As a result, iTTD can avoid recalculation for past observations, but its storage complexity grows with time. Further, iTTD does not exploit past estimations and considers the newcoming tensor as an individual separate tensor. Hence, it is essentially sensitive to time variation. iTTD is only suitable for static and small/moderate size tensors. These disadvantages motivate us to look for a scalable TT algorithm for decomposing streaming time-varying tensors.

II. Adaptive Tensor-Train Decomposition

Before introducing the problem statement, we first define some useful tensor operators used in this paper.

The mode-$(n,1)$ contracted product of two tensors $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_n}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_m}$ with $I_n = J_1$, written as $\boldsymbol{\mathcal{A}} \times_n^1 \boldsymbol{\mathcal{B}}$, yields a new tensor $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J_2 \times \cdots \times J_m}$ such that

$$\boldsymbol{\mathcal{C}}(i_1, \ldots, i_{n-1}, j_2, \ldots, j_m) = \sum_{i_n=1}^{I_n} \boldsymbol{\mathcal{A}}(i_1, \ldots, i_n) \boldsymbol{\mathcal{B}}(i_n, j_2 \ldots, j_m).$$

The concatenation of $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_n}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{n-1} \times 1}$, written as $\boldsymbol{\mathcal{A}} \boxplus \boldsymbol{\mathcal{B}}$, yields a new tensor $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times (I_n+1)}$ such that

$$\boldsymbol{\mathcal{C}}(i_1, \ldots, i_{n-1}, i_n) = \begin{cases} \boldsymbol{\mathcal{A}}(i_1, \ldots, i_{n-1}, i_n) & \text{if } i_n \leq I_n \\ \boldsymbol{\mathcal{B}}(i_1, \ldots, i_{n-1}, 1) & \text{if } i_n = I_n + 1 \end{cases}.$$

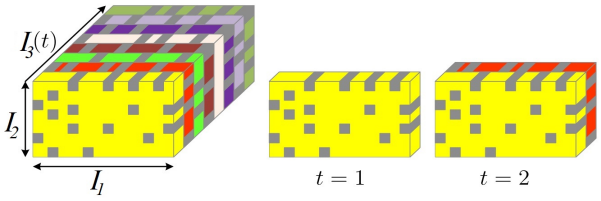


Fig. 1: Tracking TT decomposition of streaming tensors, adapted from [12].

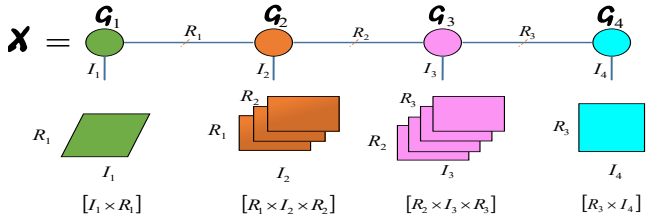


Fig. 2: A graph-based representation of TT decomposition of a 4-way tensor, reprinted from [6].

Consider a streaming n-way tensor $\boldsymbol{\mathcal{X}}[t] \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_n[t]}$ fixing all but the last "time" dimension $I_n[t]$. At time t, $\boldsymbol{\mathcal{X}}[t]$ is particularly obtained by appending a new slice $\overline{\boldsymbol{\mathcal{X}}}_t \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_{n-1}}$ to the previous observation $\boldsymbol{\mathcal{X}}[t-1]$ along the time dimension, i.e. $I_n[t] = I_n[t-1] + 1$. Instead of recomputing the batch TT decomposition for $\boldsymbol{\mathcal{X}}[t]$, we aim to develop an efficient update, both in computational complexity and memory storage, to obtain TT-cores of $\boldsymbol{\mathcal{X}}[t]$ from past estimations.

TT decomposition of $\boldsymbol{\mathcal{X}}[t]$ can be represented by a multilinear product of 3-way tensors called TT-cores:

$$\boldsymbol{\mathcal{X}}[t] = \boldsymbol{\mathcal{G}}_1[t] \times_2^1 \boldsymbol{\mathcal{G}}_2[t] \times_3^1 \cdots \times_n^1 \boldsymbol{\mathcal{G}}_n[t], \tag{1}$$

where $\mathbf{r}_{\text{TT}} = [r_1, r_2, \ldots, r_{n-1}]$ is a vector containing the TT-ranks, $\boldsymbol{\mathcal{G}}_1[t] \in \mathbb{R}^{I_1 \times r_1}$, $\boldsymbol{\mathcal{G}}_n[t] \in \mathbb{R}^{r_{n-1} \times I_n[t]}$ and $\boldsymbol{\mathcal{G}}_i[t] \in \mathbb{R}^{r_{i-1} \times I_i \times r_i}$, $i = 2, \ldots, n-1$, are the TT-cores. Fig. 2 shows an example of how to form a 4-way tensor using TT decomposition. In practice, (1) is only an approximate model in a noisy environment, i.e.,

$$\boldsymbol{\mathcal{X}}[t] = \boldsymbol{\mathcal{G}}_1[t] \times_2^1 \boldsymbol{\mathcal{G}}_2[t] \times_3^1 \cdots \times_n^1 \boldsymbol{\mathcal{G}}_n[t] + \boldsymbol{\mathcal{N}}[t], \tag{2}$$

where $\boldsymbol{\mathcal{N}}[t]$ is a noise tensor. The TT-cores can be estimated by solving the following minimization:

$$\begin{aligned} \{\boldsymbol{\mathcal{G}}_k[t]\}_{k=1}^n = \underset{\widetilde{\boldsymbol{\mathcal{X}}}, \{\boldsymbol{\mathcal{G}}_k\}_{k=1}^n}{\operatorname{argmin}} & \frac{1}{2}\|\boldsymbol{\mathcal{X}}[t] - \widetilde{\boldsymbol{\mathcal{X}}}\|_F^2, \\ \text{s.t. } & \widetilde{\boldsymbol{\mathcal{X}}} = \boldsymbol{\mathcal{G}}_1 \times_2^1 \boldsymbol{\mathcal{G}}_2 \times_3^1 \cdots \times_n^1 \boldsymbol{\mathcal{G}}_n. \end{aligned} \tag{3}$$

Problem (3) can be rewritten in the adaptive scheme as follows

$$\underset{\{\boldsymbol{\mathcal{G}}_k\}_{k=1}^n}{\operatorname{argmin}} \sum_{i=1}^{t} \lambda^{t-i} \left\| \overline{\boldsymbol{\mathcal{X}}}_i - \boldsymbol{\mathcal{G}}_1 \times_2^1 \cdots \times_n^1 \mathbf{g}_n[i] \right\|_F^2, \tag{4}$$

where $\overline{\boldsymbol{\mathcal{X}}}_i \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{n-1}}$ is the i-th slice of $\boldsymbol{\mathcal{X}}[t]$, $\mathbf{g}_n[i] \in \mathbb{R}^{r_{n-1} \times 1}$ is the i-th column of the last TT-core $\boldsymbol{\mathcal{G}}_n$ and a forgetting factor $\lambda \in (0, 1]$ is to discount the effect of past observations. The following steps describe the basic idea of our method for solving (4).

Let us denote $\boldsymbol{\mathcal{H}}[t] = \boldsymbol{\mathcal{G}}_1[t] \times_2^1 \cdots \times_{n-1}^1 \boldsymbol{\mathcal{G}}_{n-1}[t]$, and $\{\boldsymbol{\mathcal{G}}_i[t-1]\}_{i=1}^n$ be the old estimated TT-cores of $\boldsymbol{\mathcal{X}}[t-1]$. Under the assumption that TT-cores are either static or changing slowly, hence $\boldsymbol{\mathcal{H}}[t] \simeq \boldsymbol{\mathcal{H}}[t-1]$. Thus, we have

$$\begin{aligned} \boldsymbol{\mathcal{H}}[t] \times_n^1 \boldsymbol{\mathcal{G}}_n[t] &= \boldsymbol{\mathcal{X}}[t-1] \boxplus \overline{\boldsymbol{\mathcal{X}}}_t \\ &= \left(\boldsymbol{\mathcal{H}}[t-1] \times_n^1 \boldsymbol{\mathcal{G}}_n[t-1]\right) \boxplus \left(\boldsymbol{\mathcal{H}}[t] \times_n^1 \mathbf{g}_n[t]\right) \\ &\simeq \boldsymbol{\mathcal{H}}[t] \times_n^1 \left[\boldsymbol{\mathcal{G}}_n[t-1] \mid \mathbf{g}_n[t]\right]. \end{aligned}$$

Accordingly, we only need to estimate the last column vector $\mathbf{g}_n[t]$ of $\boldsymbol{\mathcal{G}}_n[t] \in \mathbb{R}^{r_{n-1} \times t}$ at time t, instead of re-estimating the whole $\boldsymbol{\mathcal{G}}_n[t]$ which becomes inefficient for a large t:

$$\boldsymbol{\mathcal{G}}_n[t] \simeq \left[\boldsymbol{\mathcal{G}}_n[t-1] \mid \mathbf{g}_n[t]\right]. \tag{5}$$

The vector $\mathbf{g}_n[t]$ can be updated by minimizing the t-th summand in (4)

$$\mathbf{g}_n[t] = \underset{\mathbf{g}_n \in \mathbb{R}^{r_{n-1} \times 1}}{\operatorname{argmin}} \left\| \overline{\boldsymbol{\mathcal{X}}}_t - \boldsymbol{\mathcal{H}}[t-1] \times_n^1 \mathbf{g}_n \right\|_F^2. \tag{6}$$

After that, we update TT-cores $\{\boldsymbol{\mathcal{G}}_k\}_{k=1}^{n-1}$ by

$$\boldsymbol{\mathcal{G}}_k[t] = \underset{\boldsymbol{\mathcal{G}}_k}{\operatorname{argmin}}\, f_t(\boldsymbol{\mathcal{G}}_k), \text{ with} \tag{7}$$

$$f_t(\boldsymbol{\mathcal{G}}_k) = \sum_{i=1}^{t} \lambda^{t-i} \left\| \overline{\boldsymbol{\mathcal{X}}}_i - \boldsymbol{\mathcal{A}}_k[t-1] \times_k^1 \boldsymbol{\mathcal{G}}_k \times_{k+1}^1 \boldsymbol{\mathcal{B}}_k[i] \right\|_F^2,$$

where the two dummy tensors are given by

$$\begin{aligned} \boldsymbol{\mathcal{A}}_k[t-1] &= \boldsymbol{\mathcal{G}}_1[t-1] \times_2^1 \cdots \times_{k-1}^1 \boldsymbol{\mathcal{G}}_{k-1}[t-1], \\ \boldsymbol{\mathcal{B}}_k[i] &= \boldsymbol{\mathcal{G}}_{k+1}[t-1] \times_{k+2}^1 \cdots \times_{n-1}^1 \boldsymbol{\mathcal{G}}_{n-1}[t-1] \times_n^1 \mathbf{g}_n[i]. \end{aligned}$$

We make the following assumptions for convenience of deploying our method: (A1) TT-cores $\{\boldsymbol{\mathcal{G}}_i\}_{i=1}^{n-1}$ may change slowly between two consecutive instances $t-1$ and t, i.e. $\boldsymbol{\mathcal{G}}_i[t] \simeq \boldsymbol{\mathcal{G}}_i[t-1]$; and (A2) TT-rank vector $\mathbf{r}_{\text{TT}} = [r_1, r_2, \ldots, r_{n-1}]$ is known and does not change with time.

III. Proposed Method

In this section, we propose an efficient first-order method, namely TT-FOA (which stands for TT adaptive decomposition using First-Order Approach), for tensor-train (TT) decomposition of streaming tensors by adapting the alternating minimization framework to the problem (4). The proposed algorithm consists of two main steps: (i) estimate $\mathbf{g}_n(t)$ first, given past estimated TT-cores; (ii) then we update TT-cores $\boldsymbol{\mathcal{G}}_i$ in parallel, given $\mathbf{g}_n[t]$ and remaining TT-cores. The pseudocode of TT-FOA is summarized in Algorithm 1.

A. Estimation of $\mathbf{g}_n[t]$

Given a new slice $\overline{\boldsymbol{\mathcal{X}}}_t$ and past estimated TT-cores, $\mathbf{g}_n[t]$ can be estimated by solving the optimization (6)

$$\mathbf{g}_n[t] = \underset{\mathbf{g}_n \in \mathbb{R}^{r_{n-1} \times 1}}{\operatorname{argmin}} \|\overline{\boldsymbol{\mathcal{X}}}_t - \boldsymbol{\mathcal{H}}[t-1] \times_n^1 \mathbf{g}_n\|_F^2 + \frac{\rho}{2}\|\mathbf{g}_n\|_2^2,$$

where ρ is a small positive parameter for regularization. It can be reformulated via its matrix-vector representation as follows

$$\underset{\mathbf{g}_n \in \mathbb{R}^{r_{n-1} \times 1}}{\operatorname{argmin}} \|\overline{\mathbf{x}}[t] - \mathbf{H}[t-1]\mathbf{g}_n\|_2^2 + \frac{\rho}{2}\|\mathbf{g}_n\|_2^2, \tag{8}$$

where $\overline{\mathbf{x}}[t] = \operatorname{vec}(\overline{\boldsymbol{\mathcal{X}}}_t)$ and $\mathbf{H}[t-1] \in \mathbb{R}^{I_1 \ldots I_{n-1} \times r_{n-1}}$ is the unfolding matrix of $\boldsymbol{\mathcal{H}}[t-1]$.

Problem (8) is an overdetermined least-squares (LS) regression, it can be efficiently solved by using the randomized sketching technique [13], as

$$\underset{\mathbf{g}_n \in \mathbb{R}^{r_{n-1} \times 1}}{\operatorname{argmin}} \|\mathcal{L}(\mathbf{H}[t-1])\mathbf{g}_n - \mathcal{L}(\overline{\mathbf{x}}[t])\|_2^2 + \frac{\rho}{2}\|\mathbf{g}_n\|_2^2, \tag{9}$$

Algorithm 1: TT-FOA: First-Order Adaptive Tensor-Train Decomposition

Input:
+ Observations $\{\overline{\boldsymbol{\mathcal{X}}}_t\}_{t=1}^{\infty}$, $\overline{\boldsymbol{\mathcal{X}}}_t \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{n-1}}$,
+ TT-rank $\mathbf{r}_{\text{TT}} = [r_1, r_2, \ldots, r_{n-1}]$,
+ Forgetting factor λ.

Output: TT-cores $\{\boldsymbol{\mathcal{G}}_i[t]\}_{i=1}^{n}$.

Initialization

$\{\boldsymbol{\mathcal{G}}_i[0]\}_{i=1}^{n-1}$ are initialized randomly,
$\{\mathbf{S}_i[0]\}_{i=1}^{n-1} = \mathbf{I}$.

for $t = 1, 2 \ldots$ **do**

Step 1: *Estimate* $\mathbf{g}_n[t]$

$$\boldsymbol{\mathcal{H}}[t-1] = \mathbf{G}_1[t-1] \times_2^1 \cdots \times_{n-1}^1 \boldsymbol{\mathcal{G}}_{n-1}[t-1]$$
$$\mathbf{H}[t-1] = \text{unfolding}(\boldsymbol{\mathcal{H}}[t-1], [I_1 I_2 \ldots I_{n-1}, r_{n-1}])$$
$$\Omega[t] = \text{randsample}([1, I_1 I_2 \ldots I_{n-1}])$$
$$\overline{\mathbf{x}}_\Omega[t] = \text{vec}(\overline{\boldsymbol{\mathcal{X}}}_t)$$
$$\mathbf{g}_n[t] = \mathbf{H}_{\Omega[t]}[t-1]^{\#} \overline{\mathbf{x}}_{\Omega[t]}[t]$$
$$\boldsymbol{\Delta}[t] = \overline{\boldsymbol{\mathcal{X}}}_t - \boldsymbol{\mathcal{H}}[t-1] \times_n^1 \mathbf{g}_n[t]$$

Step 2: *Update TT-cores* $\boldsymbol{\mathcal{G}}_k$ *in parallel*

$$\boldsymbol{\mathcal{A}}_k[t-1] = \mathbf{G}_1[t-1] \times_2^1 \cdots \times_{k-1}^1 \boldsymbol{\mathcal{G}}_{k-1}[t-1]$$
$$\mathbf{A}_k[t-1] = \text{unfolding}(\boldsymbol{\mathcal{A}}_k[t], [r_{k-1}, I_1 I_2 \ldots I_{k-1}])$$
$$\boldsymbol{\mathcal{B}}_k[t] = \boldsymbol{\mathcal{G}}_{k+1}[t-1] \times_{k+2}^1 \ldots \boldsymbol{\mathcal{G}}_{n-1}[t-1] \times_n^1 \mathbf{g}_n[t]$$
$$\mathbf{B}_k[t] = \text{unfolding}(\boldsymbol{\mathcal{B}}_k[t], [r_k, I_{k+1} I_{k+2} \ldots I_{n-1}])$$
$$\mathbf{W}_k[t] = \mathbf{B}_k[t] \otimes \mathbf{A}_k[t-1]$$
$$\mathbf{S}_k[t] = \lambda \mathbf{S}_k[t-1] + \mathbf{W}_k[t]\mathbf{W}_k[t]^T$$
$$\mathbf{V}_k[t] = \big(\mathbf{S}_k[t]\big)^{-1} \mathbf{W}_k[t]^T$$
$$\boldsymbol{\Delta}_k[t] = \text{unfolding}(\boldsymbol{\Delta}[t], [I_k, r_{k-1} r_k])$$
$$\mathbf{G}_k[t] = \mathbf{G}_k[t-1] + \boldsymbol{\Delta}_k[t]\mathbf{V}_k[t]^T$$
$$\boldsymbol{\mathcal{G}}_k[t] = \text{reshape}(\mathbf{G}_k[t], [r_{k-1}, I_k, r_k])$$

end

where $\mathcal{L}(.)$ is a sketching map. Thanks to the Kronecker structure of $\mathbf{H}[t-1]$, uniform random sampling can provide a good sketch for $\mathbf{H}[t-1]$. Accordingly, we can select rows of $\mathbf{H}[t-1]$ as well as $\overline{\mathbf{x}}[t]$ at random to form the sketch $\mathbf{H}_\Omega[t-1] \in \mathbb{R}^{|\Omega| \times r_{n-1}}$ and a sampled vector $\overline{\mathbf{x}}_\Omega[t] \in \mathbb{R}^{|\Omega| \times 1}$, where Ω denotes the set of sampling rows. Therefore, $\mathbf{g}_n[t]$ can be efficiently updated by applying the ridge regression method to (9), whose closed-form is given by

$$\mathbf{g}_n[t] = (\mathbf{H}_\Omega[t-1]^T \mathbf{H}_\Omega[t-1] + \rho \mathbf{I}_{r_{n-1}})^{-1} \mathbf{H}_\Omega[t-1]^T \overline{\mathbf{x}}_\Omega[t]. \tag{10}$$

As a result, the last TT-core $\mathbf{G}_n[t]$ is updated as follows

$$\mathbf{G}_n[t] = \big[\mathbf{G}_n[t-1], \mathbf{g}_n[t]\big]. \tag{11}$$

B. Estimation of TT-cores

Given the new slice $\overline{\boldsymbol{\mathcal{X}}}_t$ and past estimations, the k-th TT-core $\boldsymbol{\mathcal{G}}_k[t]$ can be estimated by minimizing the matrix-representation of the objective function (7), as follows

$$\mathbf{G}_k[t] = \underset{\mathbf{G}_k \in \mathbb{R}^{I_k \times r_k r_{k-1}}}{\text{argmin}} f(\mathbf{G}_k) = \sum_{i=1}^{t} \lambda^{t-i} \left\| \overline{\mathbf{X}}_i^{(k)} - \mathbf{G}_k \mathbf{W}_k[i] \right\|_F^2, \tag{12}$$

where $\mathbf{G}_k[t]$ is the mode-2 matricization of $\boldsymbol{\mathcal{G}}_k[t]$, $\overline{\mathbf{X}}_i^{(k)}$ is the mode-k matricization of $\overline{\boldsymbol{\mathcal{X}}}_i$; $\mathbf{W}_k[i] = \mathbf{B}_k[i] \otimes \mathbf{A}_k[t-1]$ where $\otimes$ denotes the Kronecker product, $\mathbf{A}_k[t-1]$ and $\mathbf{B}_k[i]$ are the unfolding matrices of $\boldsymbol{\mathcal{A}}_k[t-1]$ and $\boldsymbol{\mathcal{B}}_k[i]$ respectively; The local optimal $\boldsymbol{\mathcal{G}}_k[t]$ can be obtained by setting the first derivative of $f(\mathbf{G}_k)$ to zero:

$$\mathbf{G}_k \sum_{i=1}^{t} \lambda^{t-i} \mathbf{W}_k[i]\mathbf{W}_k[i]^T = \sum_{i=1}^{t} \lambda^{t-i} \overline{\mathbf{X}}_i^{(k)} \mathbf{W}_k[i]^T. \tag{13}$$

From that, we can obtain $\mathbf{G}_k[t]$ in the recursive way as follows: Let us denote $\mathbf{S}_k[t] = \sum_{i=1}^{t} \lambda^{t-i} \mathbf{W}_k[i]\mathbf{W}_k[i]^T$ and $\mathbf{R}_k[t] = \sum_{i=1}^{t} \lambda^{t-i} \overline{\mathbf{X}}_i^{(k)} \mathbf{W}_k[i]^T$. The two matrices $\mathbf{R}_k[t]$ and $\mathbf{S}_k[t]$ can be updated recursively:

$$\mathbf{S}_k[t] = \lambda \mathbf{S}_k[t-1] + \mathbf{W}_k[t]\mathbf{W}_k[t]^T, \tag{14}$$
$$\mathbf{R}_k[t] = \lambda \mathbf{R}_k[t-1] + \overline{\mathbf{X}}_t^{(k)} \mathbf{W}_k[t]^T. \tag{15}$$

Therefore, (13) can be rewritten as

$$\begin{aligned}\mathbf{G}_k \mathbf{S}_k[t] &= \lambda \mathbf{R}_k[t-1] + \overline{\mathbf{X}}_t^{(k)} \mathbf{W}_k[t]^T \\ &= \lambda \mathbf{G}_k[t-1]\mathbf{S}_k[t-1] + \overline{\mathbf{X}}_t^{(k)} \mathbf{W}_k[t]^T \\ &= \mathbf{G}_k[t-1]\mathbf{S}_k[t] + (\overline{\mathbf{X}}_t^{(k)} - \mathbf{G}_k[t-1]\mathbf{W}_k[t])\mathbf{W}_k[t]^T.\end{aligned}$$

Let the residual matrix $\boldsymbol{\Delta}_k[t]$ and coefficient matrix $\mathbf{V}_k[t]$ be

$$\boldsymbol{\Delta}_k[t] = \overline{\mathbf{X}}_t^{(k)} - \mathbf{G}_k[t-1]\mathbf{W}_k[t], \tag{16}$$
$$\mathbf{V}_k[t] = \mathbf{W}_k[t]^T \big(\mathbf{S}_k[t]\big)^{-1}. \tag{17}$$

We obtain a simple rule for updating $\mathbf{G}_k[t]$ as follows

$$\mathbf{G}_k[t] = \mathbf{G}_k[t-1] + \boldsymbol{\Delta}_k[t]\mathbf{V}_k[t]. \tag{18}$$

After that, the TT-core $\boldsymbol{\mathcal{G}}_k[t]$ will be derived from reshaping $\mathbf{G}_k[t]$ into a 3-way tensor of size $r_{k-1} \times I_k \times r_k$.

We also note that when dealing with large-scale and high-rank tensors (i.e. $r_i \approx I_i$), TT-FOA can be sped up by using its stochastic approximation. We refer to this method as the stochastic TT-FOA. In particular, the gradient $\nabla f(\mathbf{G}_k)$ can be approximated by the instantaneous gradient of the last summand of $f(\mathbf{G}_k)$. Thus, $\mathbf{S}_k[t]$ can be computed by

$$\mathbf{S}_k[t] \simeq \mathbf{W}_k[t]\mathbf{W}_k^T[t]. \tag{19}$$

Accordingly, the matrix $\mathbf{V}_k[t]$ in (17) can be derived directly from the right inverse of $\mathbf{W}_k$. As a result, the stochastic TT-FOA not only skips several operations, but also saves a memory storage of $\mathcal{O}(r_{k-1}^2 r_k^2)$ for storing $\mathbf{S}_k[t]$ at time t. However, the stochastic approximation achieves a lower convergence rate than the original TT-FOA, see Fig. 6 for an illustration.

C. Computational Complexity and Memory Storage Analysis

For convenience of the analysis, we assume that the fixed dimensions of the tensor are equal to I while its TT-rank is $\mathbf{r}_{\text{TT}} = [r, r, \ldots, r]$. In terms of computational complexity, TT-FOA first requires $\mathcal{O}(|\Omega| r^2)$ flops for computing $\mathbf{g}_n[t]$ by using the randomized LS method at time t. The cost for updating the k-th TT-core, $\boldsymbol{\mathcal{G}}_k[t]$, comes from matrix-matrix products except an inverse operation for $\mathbf{S}_k[t]$, hence it costs $\mathcal{O}(I^{n-1} r^2)$ flops in general. It is due to that the matrix $\mathbf{S}_k[t]$ is of size $r^2 \times r^2$, thus the computation of $(\mathbf{S}_k[t])^{-1}$ is not expensive and independent of the tensor dimension. Therefore, the overall computational complexity is $\mathcal{O}(I^{n-1} r^2)$. In term of memory storage, TT-FOA does not require to save the observation data at each time, it totally costs $\mathcal{O}\big((n-1)(Ir^2 + r^4)\big)$ for storing $n-1$ TT-cores and $n-1$ matrices $\mathbf{S}_k[t]$. When the stochastic TT-FOA is applied, the memory storage is only $\mathcal{O}\big((n-1)Ir^2\big)$.

IV. Evaluations

In this section, we conduct several experiments on both synthetic and real data to evaluate the performance of TT-FOA for adaptive TT decomposition. Experiments are implemented in MATLAB platform and are available online[1].

[1] https://sites.google.com/view/thanhtbt/.

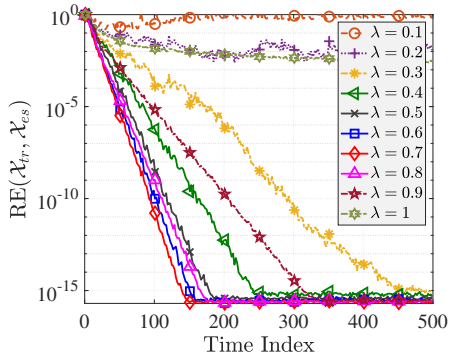


Fig. 3: Effect of the forgetting factor λ on the performance of TT-FOA.

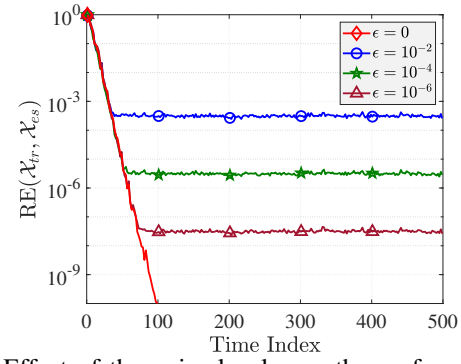


Fig. 4: Effect of the noise level ϵ on the performance of TT-FOA.

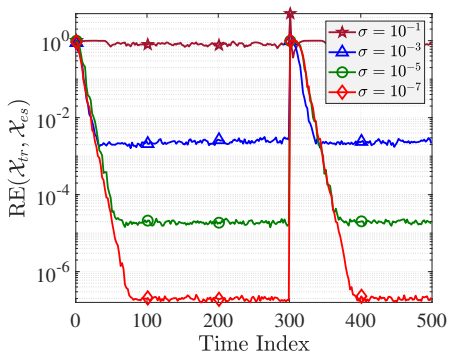


Fig. 5: Effect of the time-varying factor σ on the performance of TT-FOA in the case of noise-free.

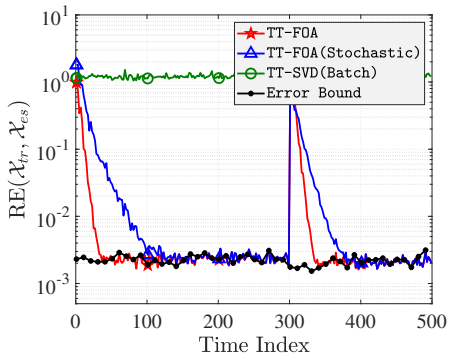


Fig. 6: Performance of three TT decomposition algorithms in a time-varying scenario: The noise level $\epsilon = 10^{-1}$ and the time variance factor $\sigma = 10^{-4}$.

A. Synthetic Data

We generate streaming 4-way tensors $\boldsymbol{\mathcal{X}}[t] \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4[t]}$ of a TT-rank vector $\mathbf{r}_{\mathrm{TT}} = [r_1, r_2, r_3]$ as follows:

$$\overline{\boldsymbol{\mathcal{X}}}_t = \boldsymbol{\mathcal{G}}_1[t] \times_2^1 \boldsymbol{\mathcal{G}}_2[t] \times_3^1 \boldsymbol{\mathcal{G}}_3[t] \times_4^1 \mathbf{g}_4[t] + \epsilon \overline{\boldsymbol{\mathcal{N}}}[t],$$

where the 3-way tensor $\overline{\boldsymbol{\mathcal{X}}}_t \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is the t-th slice of $\boldsymbol{\mathcal{X}}[t]$; $\overline{\boldsymbol{\mathcal{N}}}[t]$ is a Gaussian noise tensor of the same size with $\overline{\boldsymbol{\mathcal{X}}}_t$ and ϵ controls the noise level; the last column $\mathbf{g}_4[t]$ of TT-core $\boldsymbol{\mathcal{G}}_4[t]$ is a random vector living on $\mathbb{R}^{r_3}$ space; TT-cores $\boldsymbol{\mathcal{G}}_1[t], \boldsymbol{\mathcal{G}}_2[t]$ and $\boldsymbol{\mathcal{G}}_3[t]$ are, respectively, of size $I_1 \times r_1$, $r_1 \times I_2 \times r_2$ and $r_2 \times I_3 \times r_3$ given by

$$\boldsymbol{\mathcal{G}}_i[t] = (1 - \sigma)\boldsymbol{\mathcal{G}}_i[t-1] + \sigma \boldsymbol{\mathcal{N}}_i[t],$$

where σ controls the variation of the TT-cores between two consecutive instances, $\boldsymbol{\mathcal{N}}_1[t] \in \mathbb{R}^{I_1 \times r_1}$ and $\boldsymbol{\mathcal{N}}_i[t] \in \mathbb{R}^{r_{i-1} \times I_i \times r_i}$ are noise tensors whose entries are i.i.d from the Gaussian distribution with zero-mean and unit-variance.

To measure the estimation accuracy, we use the relative error (RE) metric given by

$$\mathrm{RE}(\boldsymbol{\mathcal{X}}_{tr}, \boldsymbol{\mathcal{X}}_{es}) = \|\boldsymbol{\mathcal{X}}_{tr} - \boldsymbol{\mathcal{X}}_{es}\|_F \;/\; \|\boldsymbol{\mathcal{X}}_{tr}\|_F,$$

where $\boldsymbol{\mathcal{X}}_{tr}$(resp. $\boldsymbol{\mathcal{X}}_{es}$) refers to the true tensor (resp. estimated tensor).

The choice of forgetting factor λ plays a central role in how fast TT-FOA converges. Fig. 3 shows the experimental results of applying the algorithm to a static and free-noise tensor whose size is $10 \times 12 \times 15 \times 500$ and its TT-rank is $\mathbf{r}_{\mathrm{TT}} = [2, 3, 5]$. We can see that the relative error is minimized when λ is round 0.7. TT-FOA fails when λ is close to its infimum or supremum. We then fix $\lambda = 0.7$ in the next experiments.

To study the effect of noise on the performance of our algorithm, we vary the value of the noise level ϵ and access its estimation on the same tensor above. The result is shown in Fig. 4. When we reduce the noise, relative error (RE) between the ground truth and estimation degrades gradually and converges towards a steady state error bound. Note that the convergence rate of the algorithm is not affected by the noise level but only its estimation error.

We next consider a scenario where TT-cores change slowly with time and abruptly at instant $t = 300$. Fig. 5 shows the performance of TT-FOA applying to the same free-noise tensor versus the time-varying factor σ. In the same manner to the effect of the noise level, TT-FOA's estimation accuracy goes down when σ increases, but converges towards a steady state error. Fig. 6 shows a performance comparison among three TT decomposition algorithms when the value of the noise level ϵ and the time-varying factor σ are 10^{-1} and 10^{-4} respectively. The batch algorithm TT-SVD fails in this time-varying scenario, while TT-FOA and its stochastic version can track successfully the variation of the tensor along the time, which yields to an estimation accuracy very close to the error bound (i.e. steady state error). The result also indicates that the convergence rate of TT-FOA is faster than that of its stochastic version. This is probably because the convergence rate of the stochastic TT-FOA is limited by its noisy/stochastic approximation of the true gradient.

B. Real Data

In order to provide empirical evidences of applying TT-FOA to real data, we use a surveillance video sequence[2], and a functional MRI data[3]. The video data contains 1546 frames of size 128×160, while the fMRI data includes 20 abdominal scans of size $256 \times 256 \times 14$.

[2]http://www.changedetection.net/

[3]https://github.com/colehawkins/

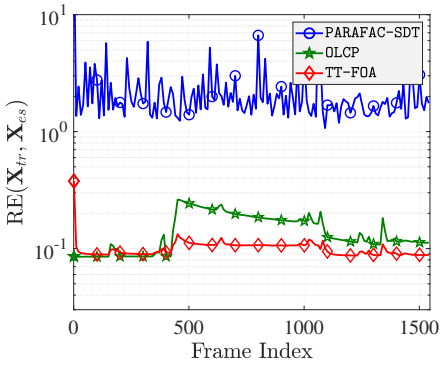


Fig. 7: Track surveillance video: TT-rank $\mathbf{r}_{\mathrm{TT}} = [15, 15]$ and CP-rank $r_{\mathrm{CP}} = 15$.

(a) Original Frame (b) PARAFAC-SDT

(c) OLCP (d) TT-FOA

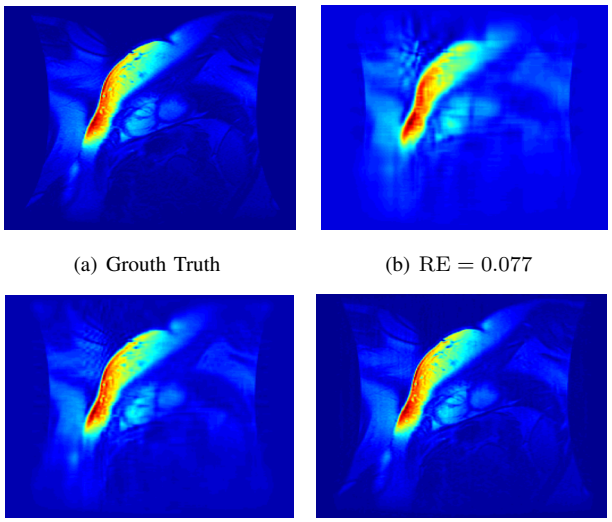
Fig. 8: Reconstructed 1345-th frame.

The first task is to track surveillance video. We compare TT-FOA against the two state-of-the-art adaptive CP tensor decompositions, including PARAFAC-SDT [14] and OLCP [15]. In order to apply these algorithms effectively, color video frames are converted into grayscale. The CP-rank and TT-rank are set at 15 and $[15, 15]$ respectively. Moreover, the 100 first video frames are trained to obtain the good initialization for PARAFAC-SDT and OLCP. The results indicate that TT-FOA outperforms these adaptive CP decompositions, as shown in Fig. 7 and Fig. 8. In particular, PARAFAC-SDT fails to track video frame while OLCP achieves a worse estimation accuracy than our algorithm.

The second task is to demonstrate the effect of TT-rank $\mathbf{r}_{\mathrm{TT}}$ on the low-rank approximation of the fMRI tensor. The abdominal scans are seen as tensor slices in the online setting. Results of tracking the low-rank component of the last scan are shown in Fig. 9. The estimated low-rank fMRI scan deviates from its ground truth when the TT-rank decreases, and hence the relative error increases.

V. Conclusions

In this paper, we proposed an efficient first-order method for tensor-train decomposition in the adaptive scheme. TT-FOA and its stochastic version are able to track the tensor-train representation of streaming tensors from noisy and high-dimensional data with high accuracy, even when they come from time-dependent observations. The effectiveness of TT-FOA is numerically validated by several experiments on both synthetic and real data.



(a) Grouth Truth (b) RE $= 0.077$

(c) RE $= 0.036$ (d) RE $= 0.007$

Fig. 9: Effect of TT-rank on the low-rank approximation of fMRI scans: (a) original MRI scan, (b)-(d) low-rank approximation images for $\mathbf{r}_{\mathrm{TT}}$ of $[10, 10]$, $[20, 20]$ and $[50, 50]$ respectively.

VI. Acknowledgment

This work was funded by the National Foundation for Science and Technology Development (NAFOSTED) of Vietnam under grant number 102.04-2019.14.

References

[1] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, 2009.

[2] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3551–3582, 2017.

[3] R. A. Harshman, "Foundations of the PARAFAC procedure: Models and conditions for an explanatory multimodal factor analysis," *UCLA Work. Papers Phonet.*, vol. 16, no. 1-84, 1970.

[4] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM J. Matrix Anal. Appl.*, vol. 21, no. 4, pp. 1253–1278, 2000.

[5] I. V. Oseledets, "Tensor-train decomposition," *SIAM J. Sci. Comput.*, vol. 33, no. 5, pp. 2295–2317, Sep. 2011.

[6] A. Cichocki, N. Lee, I. Oseledets, A. H. Phan, Q. Zhao, D. P. Mandic *et al.*, "Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions," *Found. Trends Mach. Learn.*, vol. 9, no. 4-5, pp. 249–429, 2016.

[7] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

[8] Y. Zniyed, R. Boyer, A. de Almeida, and G. Favier, "A TT-based hierarchical framework for decomposing high-order tensors," *SIAM J. Sci. Comput.*, 2020.

[9] C. Lubich, T. Rohwedder, R. Schneider, and B. Vandereycken, "Dynamical approximation by hierarchical Tucker and tensor-train tensors," *SIAM J. Matrix Anal. Appl.*, vol. 34, no. 2, pp. 470–494, 2013.

[10] C. Lubich, B. Vandereycken, and H. Walach, "Time integration of rank-constrained Tucker tensors," *SIAM J. Numer. Anal.*, vol. 56, no. 3, pp. 1273–1290, 2018.

[11] H. Liu, L. T. Yang, Y. Guo, X. Xie, and J. Ma, "An incremental tensor-train decomposition for cyber-physical-social big data," *IEEE Trans. Big Data*, pp. 1–1, 2018.

[12] T. Minh-Chinh, N. Viet-Dung, N. Linh-Trung, and K. Abed-Meraim, "Adaptive PARAFAC decomposition for third-order tensor completion," in *Int. Conf. Com. Elect.* IEEE, 2016, pp. 297–301.

[13] M. W. Mahoney *et al.*, "Randomized algorithms for matrices and data," *Found. Trends Mach. Learn.*, vol. 3, no. 2, pp. 123–224, 2011.

[14] D. Nion and N. D. Sidiropoulos, "Adaptive algorithms to track the parafac decomposition of a third-order tensor," *IEEE Trans. Signal Process.*, vol. 57, no. 6, pp. 2299–2310, 2009.

[15] S. Zhou, N. X. Vinh, J. Bailey, Y. Jia, and I. Davidson, "Accelerating online CP decompositions for higher order tensors," in *Int. Conf. Know. Disc. Data Min.*, 2016, pp. 1375–1384.