ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-7605-5/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICASSP39728.2021.9413554

A FAST RANDOMIZED ADAPTIVE CP DECOMPOSITION FOR STREAMING TENSORS

Le Trung Thanh[⋆,†]*, Karim Abed-Meraim*[⋆]*, Nguyen Linh Trung*[†]*, and Adel Hafiane*[⋆]

[⋆]PRISME Laboratory, University of Orléans/INSA-CVL, France
[†]AVITECH Institute, VNU University of Engineering and Technology, Vietnam

ABSTRACT

In this paper, we introduce a fast adaptive algorithm for CANDECOMP/PARAFAC decomposition of streaming three-way tensors using randomized sketching techniques. By leveraging randomized least-squares regression and approximating matrix multiplication, we propose an efficient first-order estimator to minimize an exponentially weighted recursive least-squares cost function. Our algorithm is fast, requiring a low computational complexity and memory storage. Experiments indicate that the proposed algorithm is capable of adaptive tensor decomposition with a competitive performance evaluation on both synthetic and real data.

***Index Terms*—** CP/PARAFAC decomposition, adaptive algorithms, streaming tensors, randomized methods.

1. INTRODUCTION

Nowadays, massive datasets have been increasingly recorded, leading to "Big Data" [1]. The era of big data has brought powerful analysis techniques for discovering new valuable information hidden in the data. Tensor decomposition, which is one of these techniques, has been recently attracting much attention of engineers and researchers [2].

A tensor is a multi-dimensional (multiway) array and tensor decomposition represents a tensor as a sum of basic components [3]. One of the most widely-used tensor decompositions is CANDECOMP/PARAFAC (CP) decomposition seeking a low rank approximation for tensors [3]. "Workhorse" algorithms for the CP decomposition are based on the alternating least-squares (ALS) method. In online applications, data acquisition is a time-varying process where data are serially acquired (streamed). This leads to several critical issues [4], among them are the following: (i) growing in size of the underlying data, (ii) time-evolving models, and (iii) (near) real-time processing. The standard CP decomposition algorithms, however, either have high complexity or operate in batch mode, and thus may not be suitable for such online applications. Adaptive (online) CP decomposition has been introduced as an efficient solution with a lower complexity and memory storage.

In the literature of tensor decomposition, there have been several proposed algorithms for adaptive CP decomposition. Many of them are based on the subspace tracking approach in which estimators first track a low-dimensional tensor subspace, and then derive the loading factors from its Khatri-Rao structure. State-of-the-art algorithms include: PARAFAC-SDT and PARAFAC-RLST by Nion and Sidiropoulos in [5], PETRELS-based CP [6] and SOAP [7]. Among these algorithms, SOAP achieves a linear computational complexity w.r.t. tensor dimensions and rank. These algorithms, however, do not utilize the Khatri-Rao structure when tracking the tensor subspace, their estimation accuracy may be reasonable when they use a good initialization. Another class of methods is based on the alternating minimization approach in which we can estimate directly all factors but the one corresponding to the dimension growing over time. Morteza *et al.* have proposed a first-order algorithm for adaptive CP decomposition by applying the stochastic gradient descent method to the cost function [8]. An accelerated version for higher-order tensors (OLCP) has been proposed by Zhou *et al.* [9]. Similar to the first class, OLCP is highly sensitive to initialization. Smith *et al.* have introduced an adaptive algorithm for handling streaming sparse tensors called CP-stream [10]. Kasai has recently developed an efficient second-order algorithm to exploit the recursive least squares algorithm, called OLSTEC in [11]. Among these algorithms, OLSTEC provides a competitive performance in terms of estimation accuracy. However, the computational complexity of all algorithms mentioned above is still high, either $\mathcal{O}(IJr)$ or $\mathcal{O}(IJr^2)$, where I, J are two fixed dimensions of the tensor and r represents its CP rank. When dealing with large-scale streaming tensors, i.e. $IJ \gg r$, it is desired to develop adaptive algorithms with a much lower (sublinear) complexity.

In this study, we consider the problem of adaptive CP tensor decomposition using randomized techniques. It is mainly motivated by the fact that randomized algorithms help reduce computational complexity and memory storage of their conventional counterparts [12]. As a result, they have recently attracted a great deal of attention and achieved great success in large-scale data analysis and tensor decomposition in particular. With respect to the CP tensor model, Wang *et al.* have applied a sketching technique to develop a fast algorithm for orthogonal tensor decomposition [13]. Under certain conditions, the tensor sketch can be obtained without accessing the entire data [14]. Recently, Battaglino *et al.* have proposed a practical randomized CP decomposition [15]. Their work was to speed up the traditional ALS

This work was supported by the National Foundation for Science and Technology Development of Vietnam under Grant No. 102.04-2019.14.

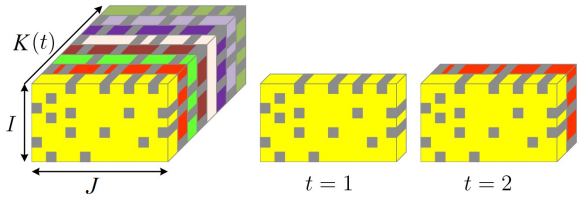


Fig. 1: Streaming tensor $\boldsymbol{\mathcal{X}}_t \in \mathbb{R}^{I\times J\times K(t)}$, reprinted from [6].

algorithm via randomized least-squares regressions. These algorithms, however, are constrained to batch-mode operations, hence not suitable for adaptive processing. Ma *et al.* introduced a randomized online CP decomposition for streaming tensors [16]. The algorithm can be considered as a randomized version of OLCP [9]. However, it is sensitive not only to initialization, but also to time-varying low-rank models. These drawbacks motivate us to look for a new efficient randomized algorithm for adaptive CP decomposition.

Notations

Scalars and vectors are denoted by lowercase letters (e.g., x) and boldface lowercase letters (e.g., $\mathbf{x}$), respectively. Boldface capital and bold calligraphic letters denote matrices (e.g., $\mathbf{X}$) and tensors (e.g., $\boldsymbol{\mathcal{X}}$), respectively. Operators $\circ, \odot$ and $\circledast$ denote the outer, Khatri-Rao, and Hadamard product, respectively. Matrix transpose is denoted by $(.)^\top$ and $\mathbf{X}(:,i)$ stands for the i-th column vector of matrix $\mathbf{X}$. Also, $\|.\|$ denotes the norm of a vector, matrix or tensor.

2. PRELIMINARIES AND PROBLEM STATEMENT

Consider a three-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I\times J\times K}$ of rank r. A CANDECOMP/PARAFAC (CP) decomposition of $\boldsymbol{\mathcal{X}}$ is expressed as follows:

$$\boldsymbol{\mathcal{X}} \triangleq [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!] = \sum_{i=1}^{r} \mathbf{A}(:,i)\circ\mathbf{B}(:,i)\circ\mathbf{C}(:,i), \quad (1)$$

where the full-rank $\mathbf{A} \in \mathbb{R}^{I\times r}$, $\mathbf{B} \in \mathbb{R}^{J\times r}$, and $\mathbf{C} \in \mathbb{R}^{K\times r}$ are called loading factors.

In order to decompose a tensor $\boldsymbol{\mathcal{X}}$ into r components under the CP model, we solve the following minimization:

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \left\|\boldsymbol{\mathcal{X}} - \widetilde{\boldsymbol{\mathcal{X}}}\right\|_F^2, \text{ s.t. } \widetilde{\boldsymbol{\mathcal{X}}} = \sum_{i=1}^{r} \mathbf{A}(:,i)\circ\mathbf{B}(:,i)\circ\mathbf{C}(:,i), \quad (2)$$

or its matrix representation

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \left\|\mathbf{X}^{(1)} - \widetilde{\mathbf{X}}\right\|_F^2, \text{ s.t. } \widetilde{\mathbf{X}}_k = \mathbf{A}\,\mathrm{diag}(\mathbf{c}_k)\mathbf{B}^\top, \quad (3)$$

where $\widetilde{\mathbf{X}} = \left[\widetilde{\mathbf{X}}_1\ \widetilde{\mathbf{X}}_2 \ldots \widetilde{\mathbf{X}}_K\right]$, $\mathbf{X}^{(1)} \in \mathbb{R}^{I\times JK}$ is a matricization of $\boldsymbol{\mathcal{X}}$ and $\mathrm{diag}(\mathbf{c}_k)$ is the diagonal matrix formed by $\mathbf{c}_k$, the k-th row of $\mathbf{C}$. "Workhorse" algorithms for CP decomposition are based on the alternating least-squares (ALS) approach [3]. CP decomposition is essentially unique under the following conditions [3]:

$$r \le K \text{ and } r(r-1) \le I(I-1)J(J-1)/2. \quad (4)$$

In this paper, we deal with a three-way tensor $\boldsymbol{\mathcal{X}}_t \in \mathbb{R}^{I\times J\times K(t)}$, where I, J are fixed and $K(t)$ varies with time, $\boldsymbol{\mathcal{X}}_t$ satisfies the conditions (4). At each time t, $\boldsymbol{\mathcal{X}}_t$ is obtained by appending a new slice $\mathbf{X}_t \in \mathbb{R}^{I\times J}$ to the previous tensor $\boldsymbol{\mathcal{X}}_{t-1}$, as shown in Fig. 1. Instead of recalculating batch CP decomposition of $\boldsymbol{\mathcal{X}}_t$, we aim to develop an update efficient in both computational complexity and memory storage, to obtain the factors of $\boldsymbol{\mathcal{X}}_t$.

In an adaptive scheme[1], we can reformulate (3) as follows:

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \left[f_t(\mathbf{A},\mathbf{B},\mathbf{C}) = \frac{1}{t}\sum_{k=1}^{t} \lambda^{t-k} \left\|\mathbf{X}_k - \mathbf{A}\,\mathrm{diag}(\mathbf{c}_k)\mathbf{B}^\top\right\|_F^2 \right], \quad (5)$$

where $\lambda \in (0,1]$ is a forgetting parameter aimed at discounting the past observations.

The minimization of (5) can be solved efficiently using the alternating minimization framework, which can be decomposed into three steps: (i) estimate $\mathbf{c}_t$, given $\mathbf{A}_{t-1}$ and $\mathbf{B}_{t-1}$; (ii) estimate $\mathbf{A}_t$, given $\mathbf{c}_t$ and $\mathbf{B}_{t-1}$; (iii) update $\mathbf{B}_t$, given $\mathbf{c}_t$ and $\mathbf{A}_t$. In this work, we will adapt this framework for developing our randomized adaptive CP algorithm.

3. PROPOSED METHOD

In this section, a fast adaptive CP decomposition algorithm using randomized techniques is developed. This method is referred to as ROLCP for Randomized OnLine CP. In particular, $\mathbf{c}_t$ is estimated first by using a randomized overdetermined least-squares method. After that, we introduce an efficient update for estimating factors $\mathbf{A}_t$ and $\mathbf{B}_t$ based on approximating matrix multiplication.

3.1. Estimation of $\mathbf{c}_t$

Given a new slice $\mathbf{X}_t$ and the two old factors $\mathbf{A}_{t-1}$ and $\mathbf{B}_{t-1}$, $\mathbf{c}_t$ can be estimated by solving the following minimization:

$$\min_{\mathbf{c}\in\mathbb{R}^r} \ \|\mathbf{H}_{t-1}\mathbf{c} - \mathbf{x}_t\|_2^2 + \frac{\rho_c}{2}\|\mathbf{c}\|_2^2, \quad (6)$$

where $\mathbf{x}_t = \mathrm{vec}(\mathbf{X}_t)$ and $\mathbf{H}_{t-1} = \mathbf{B}_{t-1} \odot \mathbf{A}_{t-1} \in \mathbb{R}^{IJ\times r}$ and ρ_c is a small positive parameter for regularization. Expression (6) is an overdetermined least-squares problem which requires $\mathcal{O}(IJr^2)$ flops w.r.t. time complexity to compute its exact solution $\mathbf{c}_{opt}$ in general [17]. Therefore, it becomes inefficient when dealing with high-dimensional (large-scale) tensors.

We here propose to solve a random sketch of (6) instead:

$$\mathbf{c}_t = \underset{\mathbf{c}\in\mathbb{R}^r}{\mathrm{argmin}} \ \|\mathcal{L}(\mathbf{H}_{t-1}\mathbf{c} - \mathbf{x}_t)\|_2^2 + \frac{\rho_c}{2}\|\mathbf{c}\|_2^2, \quad (7)$$

where $\mathcal{L}(.)$ is a sketching map that helps reduce the sample size and hence speed up the computation [12]. Indeed, we exploit that the Khatri-Rao product may increase the incoherence from its factors, thanks to the following proposition.

Proposition 1. *(Lemma 4 in [15]): Given* $\mathbf{A} \in \mathbb{R}^{I\times r}$ *and* $\mathbf{B} \in \mathbb{R}^{J\times r}$, *we have* $\mu(\mathbf{A}\odot\mathbf{B}) \le \mu(\mathbf{A})\mu(\mathbf{B})$ *where the coherence* $\mu(\mathbf{M})$ *is defined as the maximum leverage score of* $\mathbf{M} \overset{SVD}{=} \mathbf{U}_M\boldsymbol{\Sigma}_M\mathbf{V}_M^H$, *i.e.,*

$$\mu(\mathbf{M}) = \max_j \ \ell_j(\mathbf{M}), \quad \textit{where} \quad \ell_j(\mathbf{M}) = \|\mathbf{U}_M(j,:)\|_2^2.$$

[1] In the adaptive scheme, two factors $\mathbf{A}$ and $\mathbf{B}$ may be changing slowly with time, i.e., $\mathbf{A} = \mathbf{A}_t$ and $\mathbf{B} = \mathbf{B}_t$. Our adaptive CP algorithm is able to estimate factors $\mathbf{A}$ and $\mathbf{B}$ as well as track their variations with time.

Intuitively, when a matrix has strong incoherence (i.e., low coherence), all rows are almost equally important [18]. Accordingly, in many cases, the uniform row-sampling can provide a good sketch for (6) in which each row has equal chance of being selected[2], thanks to the Khatri-Rao structure of $\mathbf{H}_{t-1}$. Once formulating (7), the traditional least-squares method is applied to estimate $\mathbf{c}_t$ with a much lower complexity $\mathcal{O}(nr^2)$ where n is the number of selected rows from $\mathbf{H}_{t-1}$ under an error bound:

$$\|\mathbf{H}_{t-1}\mathbf{c}_t - \mathbf{x}_t\|_2^2 + \frac{\rho_c}{2}\|\mathbf{c}_t\|_2^2 \leq (1+\epsilon)\|\mathbf{H}_{t-1}\mathbf{c}_{opt} - \mathbf{x}_t\|_2^2 + \frac{\rho_c}{2}\|\mathbf{c}_{opt}\|_2^2, \quad (8)$$

with high probability for some parameter $\epsilon \in (0,1)$ [17]. The closed-form solution of (7) is given by

$$\mathbf{c}_t = \Big[\rho_c \mathbf{I}_r + \sum_{(i,j)\in\Omega_t} (\mathbf{a}_i \circledast \mathbf{b}_j)^\top (\mathbf{a}_i \circledast \mathbf{b}_j)\Big]^{-1} \times \sum_{(i,j)\in\Omega_t} \mathbf{X}_t(i,j)(\mathbf{a}_i \circledast \mathbf{b}_j)^\top. \quad (9)$$

where Ω_t is the set of sampling entries, $\mathbf{a}_i$ and $\mathbf{b}_j$ are the i-th and j-th row vectors of $\mathbf{A}_{t-1}$ and $\mathbf{B}_{t-1}$ respectively.

3.2. Estimation of factors $\mathbf{A}_t$ and $\mathbf{B}_t$

Given the new slice $\mathbf{X}_t$ and past estimates of $\mathbf{C}$ and $\mathbf{B}$, $\mathbf{A}_t$ can be estimated by minimizing the following cost function:

$$\mathbf{A}_t = \underset{\mathbf{A}\in\mathbb{R}^{I\times r}}{\operatorname{argmin}} \left[\frac{1}{t}\sum_{k=1}^{t} \lambda^{t-k} \left\|\mathbf{X}_k - \mathbf{A}\,\mathrm{diag}(\mathbf{c}_k)\mathbf{B}_{t-1}^\top\right\|_F^2\right]. \quad (10)$$

To find the optimal $\mathbf{A}_t$, we set the derivative of (10) to zero,

$$\mathbf{A}\sum_{k=1}^{t} \lambda^{t-k}\mathbf{W}_k^\top\mathbf{W}_k = \sum_{k=1}^{t} \lambda^{t-k}\mathbf{X}_k\mathbf{W}_k, \quad (11)$$

where $\mathbf{W}_k = \mathbf{B}_{t-1}\,\mathrm{diag}(\mathbf{c}_k)$. Instead of solving (11) directly, we can obtain $\mathbf{A}_t$ in the following recursive way:

Let us denote $\mathbf{S}_t^{(\mathbf{A})} = \sum_{k=1}^{t} \lambda^{t-k}\mathbf{W}_i^\top\mathbf{W}_k$ and $\mathbf{R}_t^{(\mathbf{A})} = \sum_{k=1}^{t} \lambda^{t-k}\mathbf{X}_k\mathbf{W}_k$. Then, $\mathbf{R}_t^{(\mathbf{A})}$ and $\mathbf{S}_t^{(\mathbf{A})}$ can be updated recursively:

$$\mathbf{S}_t^{(\mathbf{A})} = \lambda\mathbf{S}_{t-1}^{(\mathbf{A})} + \mathbf{W}_t^\top\mathbf{W}_t, \quad (12)$$

$$\mathbf{R}_t^{(\mathbf{A})} = \lambda\mathbf{R}_{t-1}^{(\mathbf{A})} + \mathbf{X}_t\mathbf{W}_t. \quad (13)$$

Using (12) and (13), (11) becomes

$$\begin{aligned}\mathbf{A}\mathbf{S}_t^{(\mathbf{A})} &= \lambda\mathbf{R}_{t-1}^{(\mathbf{A})} + \mathbf{X}_t\mathbf{W}_t \\ &= \lambda\mathbf{A}_{t-1}\mathbf{S}_{t-1}^{(\mathbf{A})} + \mathbf{X}_t\mathbf{W}_t \\ &= \mathbf{A}_{t-1}\mathbf{S}_t^{(\mathbf{A})} + \big(\mathbf{X}_t - \mathbf{A}_{t-1}\mathbf{W}_t^\top\big)\mathbf{W}_t.\end{aligned}$$

Let the residual matrix be $\mathbf{\Delta}_t = \mathbf{X}_t - \mathbf{A}_{t-1}\mathbf{W}_t^\top$ and the coefficient matrix $\mathbf{V}_t = \big(\mathbf{S}_t^{(\mathbf{A})}\big)^{-1}\mathbf{W}_t^\top$. From that, we derive a simple rule for updating $\mathbf{A}_t$ as follows

$$\mathbf{A}_t = \mathbf{A}_{t-1} + \mathbf{\Delta}_t\mathbf{V}_t^\top. \quad (14)$$

Besides, we can find further an approximation of (14) in order to speed up the update by using a sampling technique [12]:

$$\mathbf{A}_t \approx \mathbf{A}_{t-1} + \widetilde{\mathbf{\Delta}}_t\widetilde{\mathbf{V}}_t^\top, \quad (15)$$

where $\widetilde{\mathbf{\Delta}}_t$ and $\widetilde{\mathbf{V}}_t$ are randomized version of $\mathbf{\Delta}_t$ and $\mathbf{V}_t$ respectively, m is the number of columns of $\widetilde{\mathbf{\Delta}}_t$. In particular, we first compute the leverage score of each row of $\mathbf{W}_t$:

$$\ell_j(\mathbf{W}_t) = \|\mathbf{W}_t(j,:)\|_2, \text{ for } j = 1,2,\ldots,J. \quad (16)$$

After that, we will pick m columns of $\mathbf{\Delta}_t$ and $\mathbf{V}_t$ with a probability proportional to $\ell_j(\mathbf{W}_t)$.

Similarly, we also update $\mathbf{B}_t$ in the same way to $\mathbf{A}_t$.

3.3. Performance analysis

With respect to memory storage, ROLCP requires $\mathcal{O}(2r^2 + (I+J)r)$ in each time instant t, in particular for $\mathbf{A}_{t-1}, \mathbf{B}_{t-1}$ and two matrices $\mathbf{S}_t^{(\mathbf{A})}$ and $\mathbf{S}_t^{(\mathbf{B})}$. In terms of computational complexity, computation of $\mathbf{c}_t$ requires $\mathcal{O}(|\Omega_t|r^2)$ while updating $\mathbf{A}_t$ and $\mathbf{B}_t$ demands $\mathcal{O}((I+J)(m+r)r)$.

The following lemma indicates the convergence of ROLCP.

Lemma 1. *Assume that (A1) $\{\mathbf{X}_t\}_{t=1}^{\infty}$ are independent and identically distributed from a data-generation distribution $\mathbb{P}_{data}$ having a compact set $\mathcal{V}$; and (A2) the true loading factors $\{\mathbf{A}_t, \mathbf{B}_t\}_{t=1}^{\infty}$ are bounded, i.e., $\|\mathbf{A}_t\|_F^2 \leq \kappa_A < \infty$ and $\|\mathbf{B}_t\|_F^2 \leq \kappa_B < \infty$. If $\{\mathbf{A}_t, \mathbf{B}_t\}_{t=1}^{\infty}$ are generated by ROLCP, the sequence converges to a stationary point of the empirical loss function $f_t(.)$ when $t \to \infty$.*

Due to the space limitation, its proof is omitted here.

4. EXPERIMENTS

In this section, we demonstrate the effectiveness and efficiency of our algorithm, ROLCP, on both synthetic and real data. We also compare ROLCP with the state-of-the-art adaptive (online) CP algorithms, including PARAFAC-SDT [5], PARAFAC-RLST [5], OLCP [9], SOAP [7] and OLSTEC [11]. Default parameters of these algorithms are kept to have a fair comparison. Note that the first four algorithms require a batch initialization, while OLSTEC is initialized randomly. All experiments are implemented in MATLAB using a computer with an Intel core $i5$ and 16GB of RAM. Our MATLAB codes are available online at `https://github.com/thanhtbt/ROLCP/`.

4.1. Synthetic Data

Following the experimental framework in [7], at each time t, synthetic tensor data are generated under the model:

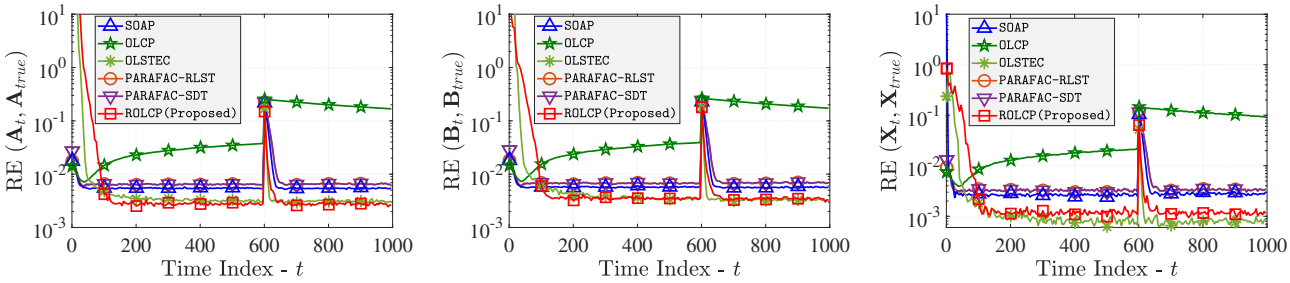
$$\mathbf{X}_t = \mathbf{A}_t\,\mathrm{diag}(\mathbf{c}_t)\mathbf{B}_t^\top + \sigma_N\mathbf{N}_X,$$

where $\mathbf{X}_t \in \mathbb{R}^{I\times J}$ is the t-th slice of $\boldsymbol{\mathcal{X}}_t$, $\mathbf{c}_t$ is a random vector living on $\mathbb{R}^r$ space, and σ_N is to control the Gaussian noise $\mathbf{N}_X \in \mathbb{R}^{I\times J}$. The two factors $\mathbf{A}_t \in \mathbb{R}^{I\times r}$, $\mathbf{B}_t \in \mathbb{R}^{J\times r}$ are defined by

$$\mathbf{A}_t = (1-\epsilon_A)\mathbf{A}_{t-1} + \epsilon_A\mathbf{N}_A, \;\; \mathbf{B}_t = (1-\epsilon_B)\mathbf{B}_{t-1} + \epsilon_B\mathbf{N}_B,$$

where ϵ_A and ϵ_B are parameters chosen to control the variation of the two factors between two consecutive instances and $\mathbf{N}_A, \mathbf{N}_B$ are two random noise matrices with Gaussian entries i.i.d of pdf $\mathcal{N}(0,1)$. In all experiments, the values of σ_N, ϵ_A,

[2] In the presence of highly coherent factors, a preconditioning (mixing) step is necessary to guarantee the incoherence. For instance, the subsampled randomized Hadamard transform is a good candidate which can yield a transformed matrix whose rows have (almost) uniform leverage scores, while the error bound (8) is still guaranteed [19].



(a) Loading factor $\mathbf{A}_t$ (b) Loading factor $\mathbf{B}_t$ (c) Observation $\mathbf{X}_t$

Fig. 2: Performance of six adaptive CP algorithms on a synthetic tensor of rank 10 and size $100 \times 150 \times 1000$.

Dataset	Highway		Hall		Lobby		Park	
Tensor size	$320 \times 240 \times 1700$		$174 \times 144 \times 3584$		$128 \times 160 \times 1546$		$288 \times 352 \times 600$	
Evaluation metric	Time(s)	RE($\boldsymbol{\mathcal{X}}$)	Time(s)	RE($\boldsymbol{\mathcal{X}}$)	Time(s)	RE($\boldsymbol{\mathcal{X}}$)	Time(s)	RE($\boldsymbol{\mathcal{X}}$)
SOAP	56.27	8.96	35.30	7.61	11.84	6.71	23.78	1.85
OLCP	41.96	4.97	27.25	6.56	8.51	5.24	**23.19**	1.01
OLSTEC	226.49	6.03	126.83	5.91	42.64	3.06	88.00	1.05
PARAFAC-RLST	406.68	8.55	187.1	7.63	55.58	8.16	**215.48**	2.16
PARAFAC-SDT	70.78	36.81	35.31	45.52	12.67	36.95	48.11	23.88
ROLCP (Proposed)	**9.63**	5.85	**6.96**	5.49	**3.31**	3.92	**3.78**	1.06

Table 1: Performance of adaptive CP algorithms on real data.

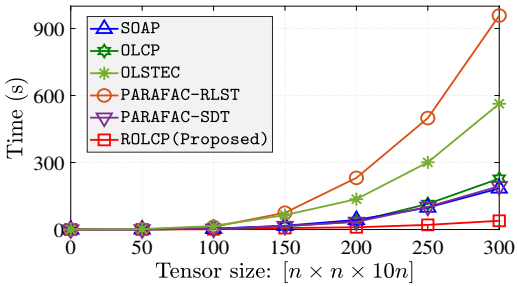


Fig. 3: Average running time of adaptive algorithms on different synthetic tensors.

and ϵ_B are set to 10^{-3}, while the forgetting factor λ is fixed at 0.9. We set $|\Omega_t| = 10r \log r$ for reasonable performance.

In order to evaluate the estimation accuracy, we use the relative error (RE) metric defined by

$$\mathrm{RE}(\mathbf{U}_{est}, \mathbf{U}_{true}) = \|\mathbf{U}_{true} - \mathbf{U}_{est}\|_F \,/\, \|\mathbf{U}_{true}\|_F,$$

where $\mathbf{U}_{true}$(resp. $\mathbf{U}_{est}$) refers to the ground truth (resp. estimation).

We use a simulated tensor whose size is $100 \times 150 \times 1000$ and its rank $r = 10$ to illustrate the effectiveness of our algorithm. At time instant $t = 600$, we set ϵ_A and ϵ_B to 10^{-1} aiming to create a significant change in the data model. The results are shown in Fig. 2. As can be seen, ROLCP provides a competitive performance as compared to OLSTEC, better than SOAP, PARAFAC-SDT and PARAFAC-RLST, while OLCP does not work well in this scenario.

The running times of these algorithms are reported in Fig. 3. We here use a sequence of simulated tensors with size of $n \times n \times 10n$, rank of $0.1n$, $n \in [10, 300]$ for this task. The result indicates that ROLCP is the fastest CP algorithm, several times faster than the second best.

4.2. Real Data

In order to demonstrate the effectiveness of ROLCP on real data, four real surveillance video sequences are used, including `Highway`, `Hall`, `Lobby` and `Park`[3]. Specifically, `Highway` contains 1700 frames of size 320×240 pixels. `Hall` has 3584 frames of size 174×144 pixels. `Lobby` consists of 1546 frames of size 128×160 pixels. `Park` includes 600 frames of size 288×352 pixels. We fix the rank at $r = 10$ for all video tensors. To have a good initialization for SOAP, OLCP, PARAFAC-RLST and PARAFAC-SDT, training slices are the 100 first video frames.

Results are shown statistically in Table 1. Clearly, our algorithm is the fastest adaptive CP decomposition. For instance, when decomposing the `Park` tensor, our running time is 3.78 seconds, 6 times faster than OLCP. The worst computation time is 215.48 seconds belonging to PARAFAC-RLST. Besides, ROLCP also provides good estimation accuracy on these data as compared to others, i.e., ROLCP usually yields reasonable RE values.

5. CONCLUSIONS

In this paper, we proposed a fast adaptive algorithm for CP decomposition based on the alternating minimization framework. ROLCP estimates a low rank approximation of tensors from noisy and high dimensional data with high accuracy, even when the model may be time-varying. Thanks to the randomized sampling techniques, ROLCP is shown to be one of the fastest adaptive CP algorithms, several times faster than SOAP and OLCP in both synthetic and real data.

[3]Data: http://jacarini.dinf.usherbrooke.ca/

6. REFERENCES

[1] Min Chen, Shiwen Mao, and Yunhao Liu, "Big data: A survey," *Mobile Netw. Appl.*, vol. 19, no. 2, pp. 171–209, 2014.

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

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

[4] Taiwo Kolajo, Olawande Daramola, and Ayodele Adebiyi, "Big data stream analysis: A systematic literature review," *J. Big Data*, vol. 6, no. 1, pp. 1–30, 2019.

[5] 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.

[6] T. M. Chinh, V. D. Nguyen, N. L. Trung, and K. Abed-Meraim, "Adaptive PARAFAC decomposition for third-order tensor completion," in *IEEE Int. Conf. Commun. Elect.*, 2016, pp. 297–301.

[7] V. D. Nguyen, K. Abed-Meraim, and N. L. Trung, "Second-order optimization based adaptive PARAFAC decomposition of three-way tensors," *Digit. Signal Process.*, vol. 63, pp. 100–111, 2017.

[8] M. Mardani, G. Mateos, and G. B. Giannakis, "Subspace learning and imputation for streaming big data matrices and tensors," *IEEE Trans. Signal Process.*, vol. 63, no. 10, pp. 2663–2677, 2015.

[9] Shuo Zhou, Nguyen Xuan Vinh, James Bailey, Yunzhe Jia, and Ian Davidson, "Accelerating online CP decompositions for higher order tensors," in *ACM Int. Conf. Knowl. Discover. Data Min.*, 2016, pp. 1375–1384.

[10] Shaden Smith, Kejun Huang, Nicholas D Sidiropoulos, and George Karypis, "Streaming tensor factorization for infinite data sources," in *SIAM Int. Conf. Data Min.*, 2018, pp. 81–89.

[11] Hiroyuki Kasai, "Fast online low-rank tensor subspace tracking by CP decomposition using recursive least squares from incomplete observations," *Neurocomput.*, vol. 347, pp. 177–190, 2019.

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

[13] Yining Wang, Hsiao-Yu Tung, Alexander J Smola, and Anima Anandkumar, "Fast and guaranteed tensor decomposition via sketching," in *Adv. Neural Inf. Process. Syst.*, 2015, pp. 991–999.

[14] Zhao Song, David Woodruff, and Huan Zhang, "Sublinear time orthogonal tensor decomposition," in *Adv. Neural Inf. Process. Syst.*, 2016, pp. 793–801.

[15] Casey Battaglino, Grey Ballard, and Tamara G Kolda, "A practical randomized CP tensor decomposition," *SIAM J. Matrix Analy. Appl.*, vol. 39, no. 2, pp. 876–901, 2018.

[16] C. Ma, X. Yang, and H. Wang, "Randomized online CP decomposition," in *Int. Conf. Adv. Comput. Intell.*, 2018, pp. 414–419.

[17] Garvesh Raskutti and Michael W Mahoney, "A statistical perspective on randomized sketching for ordinary least-squares," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 7508–7538, 2016.

[18] Yudong Chen, "Incoherence-optimal matrix completion," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2909–2923, 2015.

[19] Joel A Tropp, "Improved analysis of the subsampled randomized Hadamard transform," *Adv. Adapt Data Anal.*, vol. 3, no. 01n02, pp. 115–126, 2011.