Online Rank-Revealing Block-Term Tensor Decomposition

Athanasios A. Rontogiannis
IAASARS, National Observatory of Athens
152 36 Penteli, Greece
tronto@noa.gr

Eleftherios Kofidis
Dept. of Statistics and Insurance Science
University of Piraeus
185 34 Piraeus, Greece
kofidis@unipi.gr

Paris V. Giampouras
Mathematical Inst. for Data Science
Johns Hopkins University
Baltimore, MD 21218, USA
parisg@jhu.edu

***Abstract*—The so-called block-term decomposition (BTD) tensor model, especially in its rank-$(L_r, L_r, 1)$ version, has been recently receiving increasing attention due to its enhanced ability of representing systems and signals that are composed of *block* components of rank higher than one, a scenario encountered in numerous and diverse applications. Its uniqueness and approximation have thus been thoroughly studied. The challenging problem of estimating the BTD model structure, namely the number of block terms (rank) and their individual (block) ranks, is of crucial importance in practice and has only recently started to attract significant attention. In data-streaming scenarios and/or big data applications, where the tensor dimension in one of its modes grows in time or can only be processed incrementally, it is essential to be able to perform model selection and computation in a recursive (incremental/online) manner. To date there is only one such work in the literature concerning the (general rank-(L, M, N)) BTD model, which proposes an incremental method, however with the BTD rank and block ranks assumed to be a-priori known and time invariant. In this paper, a novel approach to rank-$(L_r, L_r, 1)$ BTD model selection and tracking is proposed, based on the idea of imposing column sparsity jointly on the factors and estimating the ranks as the numbers of factor columns of nonnegligible magnitude. An online method of the alternating reweighted least squares (RLS) type is developed and shown to be computationally efficient and fast converging, also allowing the model ranks to change in time. Its time and memory efficiency are evaluated and favorably compared with those of the batch approach. Simulation results are reported that demonstrate the effectiveness of the proposed scheme in both selecting and tracking the correct BTD model.**

I. Introduction

Although [1] introduced BTD as a sum of R rank-(L_r, M_r, N_r) terms ($r = 1, 2, \ldots, R$) in general, the special case of rank-$(L_r, L_r, 1)$ BTD has attracted a lot more of attention, because of both its more frequent occurrence in a wide range of applications and the existence of more concrete and easier to check uniqueness conditions (cf. [2] for an extensive review). Note that the mode with the unit modal rank is most commonly associated with the temporal dimension, as it is the case, for example, in tensorial functional Magnetic Resonance Imaging (fMRI) [3]. This special yet very popular BTD model is at the focus of the present work and is briefly defined as follows for a 3rd-order tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{C}^{I \times J \times K}$:

$$\boldsymbol{\mathcal{X}} = \sum_{r=1}^{R} \mathbf{A}_r \mathbf{B}_r^{\mathrm{T}} \circ \mathbf{c}_r, \tag{1}$$

with the matrices $\mathbf{A}_r \in \mathbb{C}^{I \times L_r}$ and $\mathbf{B}_r \in \mathbb{C}^{J \times L_r}$ being of full column rank, L_r, $\mathbf{c}_r$ being a nonzero column K-vector and $\circ$ denoting outer product. Clearly, canonical polyadic decomposition (CPD) results as a special case with all $L_r, r = 1, 2, \ldots, R$ equal to 1.

In general, R and L_r, $r = 1, 2, \ldots, R$ are assumed *a-priori* known (and it is commonly assumed that all L_r are all equal to L, for simplicity). However, unless external information is given, there is no way to know these values beforehand. Model selection for BTD has only recently started to be studied (cf. [2] for an extensive review of heuristic approaches and techniques). The most recent contribution of this kind can be found in our work [2], which relies on a regularization of the squared approximation error function with the sum of the Frobenius norms of the factors reweighted by a diagonal weighting which jointly depends on them in two levels: the reweighted norms of $\mathbf{A} \triangleq \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 & \cdots & \mathbf{A}_R \end{bmatrix}$ and $\mathbf{B} \triangleq \begin{bmatrix} \mathbf{B}_1 & \mathbf{B}_2 & \cdots & \mathbf{B}_R \end{bmatrix}$ are combined and then coupled with the reweighted norm of $\mathbf{C} \triangleq \begin{bmatrix} \mathbf{c}_1 & \mathbf{c}_2 & \cdots & \mathbf{c}_R \end{bmatrix}$. This two-level coupling naturally matches the structure of the model in (1), making explicit the different roles of $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$. This way, column sparsity is imposed *jointly* on the factors and in a *hierarchical* manner, which allows to estimate the ranks as the numbers of factor columns of non-negligible energy. Following a block coordinate descent (BCD) solution approach, an alternating hierarchical iterative reweighted least squares (HIRLS) algorithm, called BTD-HIRLS, was developed in [2] that manages to both reveal the ranks and compute the BTD factors at a high convergence rate and low computational cost.

In practice, data may be streaming (arriving sequentially in time) or the data-generation mechanism (the model) may change with time (cf. [4] and references therein). In big data applications, the tensor to be decomposed may be too large to fit in the memory and being processed as a whole. A possible workaround is then to recursively update the decomposition model in an incremental manner. In both cases, a sequence of optimization problems results [4]. Instead of re-

P. V. Giampouras is supported by the European Union under the Horizon 2020 Marie Skłodowska-Curie Global Fellowship program: HyPPOCRATES-H2020-MSCA-IF-2018, Grant Agreement Number: 844290.

selecting and re-computing the model of the entire tensor every time new data arrive, which is computationally and memory costly (for fast streamed and/or large-scale tensors) even when the previous model is used to initialize the computations, a recursive update is more desirable. This will update the model with relatively few additional operations and will be memory efficient. The aim of such a decomposition is to track the model in (nearly) real-time with minimal memory overhead, while attaining a modeling performance comparable to that of the batch (in terms of decomposing the tensor in increasing batches or in its entirety) approach. In any case, an online approach is the only effective way to follow if the model structure (ranks) and parameters change with time as it is the case in numerous application contexts such as video surveillance, network monitoring, dynamic neuroimaging, and others.

Our contribution lies in extending the batch approach of [2] to online tensor factorization (OTF) with possibly changing ranks. Specifically, we first propose and develop here a relaxed variant of BTD-HIRLS, which is more amenable to an online extension. The hierarchical nature of BTD-HIRLS is relaxed, considering the third mode separately from the other modes. For extending to the online setting this so-called BTD-IRLS algorithm, we expand on earlier work of ours on online rank-revealing matrix decomposition [5]. We consider exponential windowing to realize fading memory, that is, gradually forget the effect of past slices. The proposed algorithm is highly time- and memory-efficient in the sense that the involved quantities are updated recursively in an efficient manner and its computational and memory requirements do not depend on the horizon of the growing dimension. Moreover, as demonstrated via simulations, the quality of the data approximation is comparable with that of the batch version and the ranks are estimated and tracked correctly with a high probability.

The so-called *OnlineBTD* algorithm (inspired from the OnlineCPD of [6]) recently reported in [7] is to the best of our knowledge the only method for BTD that operates in an online fashion. It is shown to outperform batch alternating least squares (ALS) and nonlinear LS (NLS) in terms of time and memory efficiency while attaining a comparable approximation error. Moreover, it is made to work for general rank-(L, M, N) BTD and for tensors of higher (than 3) order as well. However, both the number of block terms and their multilinear ranks are assumed to be fixed and *a-priori* known. The authors show that their method's performance is rather robust to overestimates of these ranks. However, this may not be always the case depending on the application (see [2] and references therein) plus that one might want to have a sufficiently accurate estimate of the ranks for the purposes of interpreting the data (as in, e.g., hyperspectral imaging, where the rank signifies the number of endmembers and the block ranks stand for the ranks of the corresponding abundance maps). Moreover, it may be the case in practice that the ranks vary with time. Our online method for rank-$(L_r, L_r, 1)$ BTD overcomes these limitations, by automatically estimating the ranks and tracking them in time. [8] and [9] are rare exceptions of OTF works, where the rank is allowed to be time varying. However, they are only concerned with CPD and are not automatic. In the former, new rank estimation and subsequent CPD procedure is needed for each newly arrived tensor, which considerably complicates the algorithm. [9] has proposed to track the changing (nonnegative) rank through expressing the factors in the non-evolving modes in terms of overcomplete dictionaries and pursuing the corresponding sparse code matrices. It should be stressed that, in our method of online BTD, automatic rank estimation and tracking is directly implied by the definition of the regularization term.

II. Problem Statement

Let the $I \times J \times k$ tensor $\boldsymbol{\mathcal{Y}}^{(k)}$ grow in its 3rd mode, i.e. for increasing k, with an additional $I \times J$ frontal slice being considered (in an incremental or streaming fashion) per step. The aim is to compute the best (in the LS sense) rank-$(L_r, L_r, 1)$ approximation of $\boldsymbol{\mathcal{Y}}^{(k)}$, $\hat{\boldsymbol{\mathcal{Y}}}^{(k)} = \sum_{r=1}^{R} \mathbf{A}_r^{(k)} \mathbf{B}_r^{(k)\mathrm{T}} \circ \mathbf{c}_r^{(k)}$, in a recursive manner, that is, based on the BTD model for the $I \times J \times (k-1)$ tensor $\boldsymbol{\mathcal{Y}}^{(k-1)}$, compute the matrices $\mathbf{A}_r^{(k)} = \begin{bmatrix} \mathbf{a}_{r,1}^{(k)} & \mathbf{a}_{r,2}^{(k)} & \cdots & \mathbf{a}_{r,L_r}^{(k)} \end{bmatrix} \in \mathbb{C}^{I \times L_r}$, $\mathbf{B}_r^{(k)} = \begin{bmatrix} \mathbf{b}_{r,1}^{(k)} & \mathbf{b}_{r,2}^{(k)} & \cdots & \mathbf{b}_{r,L_r}^{(k)} \end{bmatrix} \in \mathbb{C}^{J \times L_r}$, and $\mathbf{C}^{(k)} \in \mathbb{C}^{k \times R}$, with the ranks R and L_r, $r = 1, 2, \ldots, R$ assumed *a-priori unknown and possibly varying with* k. The assumption is that the incoming data do not have a detrimental effect on the model (i.e., the BTD model keeps being valid, albeit with possibly changing ranks), affecting smoothly only its local features (in the sense of [10]). Hence (1) holds throughout, with varying factors $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and possibly R and L_rs. It is natural to assume, for a rank-$(L_r, L_r, 1)$ BTD model, that it is the 3rd dimension that grows, while the other dimensions are kept unchanged. The growing mode could correspond to, for example, time in fMRI [3] and dynamic MRI [11].

Recall that, in terms of its mode unfoldings, the tensor in (1) can be written as [1]

$$\mathbf{X}_{(1)}^{\mathrm{T}} = (\mathbf{B} \odot \mathbf{C})\mathbf{A}^{\mathrm{T}} \triangleq \mathbf{P}\mathbf{A}^{\mathrm{T}}, \tag{2}$$

$$\mathbf{X}_{(2)}^{\mathrm{T}} = (\mathbf{C} \odot \mathbf{A})\mathbf{B}^{\mathrm{T}} \triangleq \mathbf{Q}\mathbf{B}^{\mathrm{T}}, \tag{3}$$

$$\mathbf{X}_{(3)}^{\mathrm{T}} = \left[(\mathbf{A}_1 \odot_{\mathrm{c}} \mathbf{B}_1)\mathbf{1}_{L_1} \cdots (\mathbf{A}_R \odot_{\mathrm{c}} \mathbf{B}_R)\mathbf{1}_{L_R}\right] \mathbf{C}^{\mathrm{T}} \triangleq \mathbf{S}\mathbf{C}^{\mathrm{T}} \tag{4}$$

where, in its general (partition-wise) version, the Khatri-Rao (KR) product is denoted by $\odot$, and is $\odot_{\mathrm{c}}$ in its column-wise version. Moreover, the kth frontal slice can be expressed as $\boldsymbol{\mathcal{X}}(:,:,k) = \mathbf{A}\mathrm{blockdiag}(c_{k,1}\mathbf{I}_{L_1}, c_{k,2}\mathbf{I}_{L_2}, \ldots, c_{k,R}\mathbf{I}_{L_R})\mathbf{B}^{\mathrm{T}}$, which, for the most common case of all equal $L_r = L$, becomes $\boldsymbol{\mathcal{X}}(:,:,k) = \mathbf{A}(\mathrm{diag}(\mathbf{C}(k,:)) \otimes \mathbf{I}_L)\mathbf{B}^{\mathrm{T}}$.

III. Batch BTD-IRLS Algorithm

A simpler definition of the regularizer in [2], which relaxes the coupling between $\mathbf{A}_r, \mathbf{B}_r$ and $\mathbf{c}_r$ and greatly facilitates the development of a recursive decomposition scheme, is

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \frac{1}{2} \left\| \boldsymbol{\mathcal{Y}} - \sum_{r=1}^{R} \left(\mathbf{A}_r \mathbf{B}_r^{\mathrm{T}}\right) \circ \mathbf{c}_r \right\|_{\mathrm{F}}^2 + \tag{5}$$

$$\lambda \sum_{r=1}^{R} \sum_{l=1}^{L} \sqrt{\|\mathbf{a}_{r,l}\|_2^2 + \|\mathbf{b}_{r,l}\|_2^2 + \eta^2} + \mu \sum_{r=1}^{R} \sqrt{\|\mathbf{c}_r\|_2^2 + \eta^2},$$

Algorithm 1: BTD-IRLS algorithm

Input: $\mathcal{Y}$,$\lambda, \mu, R_{\text{ini}}, L_{\text{ini}}$
Output: Best (in the sense of (5)) BTD approximation of $\mathcal{Y}$
Initialize: $\mathbf{A}^{(0)}, \mathbf{B}^{(0)}, \mathbf{C}^{(0)}$
$n \leftarrow 0$
repeat
  Compute $\mathbf{D}_1^{(n)}, \mathbf{D}_2^{(n)}$
  $\mathbf{P}^{(n)} \leftarrow \mathbf{B}^{(n)} \odot \mathbf{C}^{(n)}$
  $\mathbf{A}^{(n+1)} \leftarrow \mathbf{Y}_{(1)}\mathbf{P}^{(n)} \left(\mathbf{P}^{(n)\mathrm{T}}\mathbf{P}^{(n)} + \lambda \mathbf{D}_2^{(n)}\right)^{-1}$
  $\mathbf{Q}^{(n)} \leftarrow \mathbf{C}^{(n)} \odot \mathbf{A}^{(n)}$
  $\mathbf{B}^{(n+1)} \leftarrow \mathbf{Y}_{(2)}\mathbf{Q}^{(n)} \left(\mathbf{Q}^{(n)\mathrm{T}}\mathbf{Q}^{(n)} + \lambda \mathbf{D}_2^{(n)}\right)^{-1}$
  $\mathbf{S}^{(n)} \leftarrow \left[\begin{array}{ccc} \left(\mathbf{A}_1^{(n)} \odot_{\mathrm{c}} \mathbf{B}_1^{(n)}\right) \mathbf{1}_L & \cdots & \left(\mathbf{A}_R^{(n)} \odot_{\mathrm{c}} \mathbf{B}_R^{(n)}\right) \mathbf{1}_L \end{array} \right]$
  $\mathbf{C}^{(n+1)} \leftarrow \mathbf{Y}_{(3)}\mathbf{S}^{(n)}(\mathbf{S}^{(n)\mathrm{T}}\mathbf{S}^{(n)} + \mu \mathbf{D}_1^{(n)})^{-1}$
  $n \leftarrow n+1$
until *convergence*

where η^2 is a very small positive constant that ensures smoothness at zero and R and L here stand for the initial *(over)estimates* of the model rank parameters. Note that the regularization parameters of the terms associated with $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$, namely λ and μ, may in general differ. It should be noted that, ignoring η^2, the two regularization terms above coincide with the $\ell_{1,2}$ norms of $\left[\mathbf{A}^{\mathrm{T}}\ \mathbf{B}^{\mathrm{T}}\right]^{\mathrm{T}}$ and $\mathbf{C}$, respectively. Hence, as detailed and demonstrated in [2], it is expected that this regularization scheme will also promote column sparsity simultaneously on the factors $\mathbf{A}, \mathbf{B}$ (jointly) and $\mathbf{C}$, allowing the actual ranks R and L_rs to be recovered. Following a similar methodology as in [2], namely employing a BCD scheme (with the blocks being the factors $\mathbf{A}, \mathbf{B}, \mathbf{C}$) and relying on majorization-minimization (MM) [12] for each block, we can develop an IRLS-type algorithm for solving (5). The algorithm is tabulated as Algorithm 1 and is seen to only involve closed-form matrix-wise operations for $\mathbf{A}, \mathbf{B}, \mathbf{C}$ at each iteration. The ranks are over-estimated as $R = R_{\text{ini}}$ and $L_r = L_{\text{ini}}, r = 1, 2, \ldots, R$ and, provided that λ, μ are appropriately selected, their true values are recovered as the numbers of columns of non-negligible magnitude of $\mathbf{C}$ and the corresponding $\mathbf{A}_r, \mathbf{B}_r$ blocks, respectively. This can be done either after convergence or in the course of the procedure, accompanied by the respective column pruning. BTD-IRLS differs from the BTD-HIRLS scheme of [2] in that reweighting is done separately for $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ (reflecting the 'de-coupled' nature of the regularizer) and not in a two-level hierarchy as in BTD-HIRLS, hence the name of the new algorithm. The reweighting diagonal matrices $\mathbf{D}_1^{(n)}$ and $\mathbf{D}_2^{(n)}$, of order R and LR, respectively, are given by $\mathbf{D}_1^{(n)}(r,r) = \left(\|\mathbf{c}_r^{(n)}\|_2^2 + \eta^2\right)^{-1/2}$ and $\mathbf{D}_2^{(n)}\left((r-1)L+l, (r-1)L+l\right) = \left(\|\mathbf{a}_{r,l}^{(n)}\|_2^2 + \|\mathbf{b}_{r,l}^{(n)}\|_2^2 + \eta^2\right)^{-1/2}$ and are applied for reweighting $\mathbf{C}$ and $\mathbf{A}, \mathbf{B}$, respectively. As demonstrated through simulations, the above algorithm shares the rank-revelation ability of its counterpart of [2] and is also fast converging. An online version of it is developed next.

IV. A Rank-Revealing Online BTD Algorithm

In order to incorporate in the previous method the ability to track time-varying BTD models, we define an exponentially windowed version of the objective function in (5) by formulating the optimization problem at time step k as

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \frac{1}{2}\sum_{\kappa=1}^{k} \xi^{k-\kappa} \left\| \mathbf{Y}^{(\kappa)} - \mathbf{A}\left(\mathrm{diag}(\boldsymbol{\gamma}^{(\kappa)}) \otimes \mathbf{I}_L\right)\mathbf{B}^{\mathrm{T}} \right\|_{\mathrm{F}}^2 + \quad (6)$$
$$\lambda \sum_{r=1}^{R}\sum_{l=1}^{L} \sqrt{\|\mathbf{a}_{r,l}\|_2^2 + \|\mathbf{b}_{r,l}\|_2^2 + \eta^2} + \mu \sum_{r=1}^{R} \sqrt{\|\boldsymbol{\Xi}^{(k)\frac{1}{2}}\mathbf{c}_r\|_2^2 + \eta^2},$$

where $\mathbf{Y}^{(\kappa)}$ is the κth $I \times J$ slice of $\mathcal{Y}^{(k)}$, $\boldsymbol{\gamma}^{(\kappa)\mathrm{T}} \triangleq \mathbf{C}(\kappa, :)$ is the κth row of $\mathbf{C}$, and $\boldsymbol{\Xi}^{(k)} \triangleq \mathrm{diag}\left(\xi^{k-1}, \ldots, \xi, 1\right)$, with $0 < \xi \leq 1$ being the forgetting factor. The aim is, given the model for the tensor consisting of the first $k-1$ slices and the new, kth slice $\mathbf{Y}^{(k)}$, to update the model parameters $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and the model orders R and $L_r, r = 1, 2, \ldots, R$ based on their values at $\kappa = k-1$, in a time- and memory-efficient way. As previously, R and L are the over-estimates of the rank parameters, with all L_r being over-estimated as L, and we have thus employed the corresponding expression for the frontal slice. It must be emphasized that these parameters are not only *a-priori* unknown but also potentially changing with k. Exponential time-weighting is also applied on the columns of $\mathbf{C}$, while no weighting is required in the term involving the factors $\mathbf{A}$ and $\mathbf{B}$ since these do not change in size with k. As it is common in the OTF literature, we will make the assumption that the latter factors, that is those corresponding to the non-evolving modes, are only changing slowly.

Let $\mathbf{Y}_{(1)}^{(k-1)}, \mathbf{Y}_{(2)}^{(k-1)}$ and $\mathbf{Y}_{(3)}^{(k-1)}$ be the mode unfoldings of the $I \times J \times (k-1)$ tensor $\mathcal{Y}^{(k-1)}$, available at step $k-1$, and $\mathbf{y}^{(k)} \triangleq \mathrm{vec}(\mathbf{Y}^{(k)\mathrm{T}})$ the row vectorization of $\mathbf{Y}^{(k)}$, the new $I \times J$ slice that is included at step k. Then (cf. [13, Fig. 2]) the mode unfoldings of the incremented tensor $\mathcal{Y}^{(k)}$ can be expressed as follows:

$$\mathbf{Y}_{(1)}^{(k)} = \left[\begin{array}{cc} \mathbf{Y}_{(1)}^{(k-1)} & \mathbf{Y}^{(k)} \end{array} \right] \mathbf{U}^{(k)}, \quad (7)$$

$$\mathbf{Y}_{(2)}^{(k)} = \left[\begin{array}{cc} \mathbf{Y}_{(2)}^{(k-1)} & \mathbf{Y}^{(k)\mathrm{T}} \end{array} \right], \quad (8)$$

$$\mathbf{Y}_{(3)}^{(k)} = \left[\begin{array}{cc} \mathbf{Y}_{(3)}^{(k-1)\mathrm{T}} & \mathbf{y}^{(k)} \end{array} \right]^{\mathrm{T}}, \quad (9)$$

where $\mathbf{U}^{(k)}$ is the $kJ \times kJ$ permutation matrix that moves the jth column of $\mathbf{Y}^{(k)}$ to the jkth position of the resulting matrix. It follows from (4) and (9) and the assumption of slowly-varying $\mathbf{A}, \mathbf{B}$ that the factor $\mathbf{C}$ will only change in its new row at each step k, that is,

$$\mathbf{C}^{(k)} = \left[\begin{array}{cc} \mathbf{C}^{(k-1)\mathrm{T}} & \boldsymbol{\gamma}^{(k)} \end{array} \right]^{\mathrm{T}}. \quad (10)$$

Hence the problem of estimating $\mathbf{C}$ at the kth step can be cast from (6) in terms of its last row and using (4) can be written as:

$$\boldsymbol{\gamma}^{(k)} = \arg\min_{\boldsymbol{\gamma}} \frac{1}{2}\left\| \mathbf{y}^{(k)} - \mathbf{S}^{(k-1)}\boldsymbol{\gamma} \right\|_2^2 + \quad (11)$$
$$\mu \sum_{r=1}^{R} \sqrt{\xi \left\| \boldsymbol{\Xi}^{(k-1)\frac{1}{2}}\mathbf{c}_r^{(k-1)} \right\|_2^2 + \gamma_r^2 + \eta^2},$$

where $\mathbf{c}_r^{(k-1)}$ is the rth column of $\mathbf{C}^{(k-1)}$ and γ_r is the rth element of the optimization variable $\boldsymbol{\gamma}$ while $\mathbf{S}^{(k-1)}$ is defined according to (4). This sub-problem can be solved via MM [14], leading to the following closed-form solution for $\boldsymbol{\gamma}^{(k)}$:

$$\boldsymbol{\gamma}^{(k)} = \left(\mathbf{S}^{(k-1)\mathrm{T}}\mathbf{S}^{(k-1)} + \mu\mathbf{D}_1^{(k-1)}\right)^{-1}\mathbf{S}^{(k-1)\mathrm{T}}\mathbf{y}^{(k)}, \quad (12)$$

where $\mathbf{D}_1^{(k-1)}$ is the $R \times R$ diagonal matrix with diagonal entries

$$\mathbf{D}_1^{(k-1)}(r,r) = \left(\xi\left\|\boldsymbol{\Xi}^{(k-1)\frac{1}{2}}\mathbf{c}_r^{(k-1)}\right\|_2^2 + \eta^2\right)^{-1/2}. \quad (13)$$

Observe that the squared norm above can also be recursively computed.

$\mathbf{C}$ is updated row-by-row from (12). On the contrary, each of the factors $\mathbf{A}$ and $\mathbf{B}$ must be updated as a whole upon the arrival of a new slice. Writing (6) in terms of the mode-1 unfoldings and making use of (2) we obtain the $\mathbf{A}$ sub-problem in the form

$$\mathbf{A}^{(k)} = \arg\min_{\mathbf{A}} \frac{1}{2}\left\|{}_J\boldsymbol{\Xi}^{(k)\frac{1}{2}}\left(\mathbf{Y}_{(1)}^{(k)\mathrm{T}} - \mathbf{P}^{(k)}\mathbf{A}^{\mathrm{T}}\right)\right\|_{\mathrm{F}}^2 + \lambda\sum_{r=1}^{R}\sum_{l=1}^{L}\sqrt{\|\mathbf{a}_{r,l}\|_2^2 + \|\mathbf{b}_{r,l}^{(k-1)}\|_2^2 + \eta^2}, \quad (14)$$

where $\mathbf{P}^{(k)} = \mathbf{B}^{(k)} \odot \mathbf{C}^{(k)}$ (cf. (2)) and ${}_J\boldsymbol{\Xi}^{(k)}$ stands for $\mathbf{I}_J \otimes \boldsymbol{\Xi}^{(k)}$. Solving (14) via MM [14] leads to the following closed-form expression for $\mathbf{A}^{(k)}$:

$$\mathbf{A}^{(k)} = \mathbf{Y}_{(1)}^{(k)}{}_J\boldsymbol{\Xi}^{(k)}\mathbf{P}^{(k)}\left(\mathbf{P}^{(k)\mathrm{T}}{}_J\boldsymbol{\Xi}^{(k)}\mathbf{P}^{(k)} + \lambda\mathbf{D}_2^{(k-1)}\right)^{-1}, \quad (15)$$

where $\mathbf{D}_2^{(k-1)}$ is the $LR \times LR$ diagonal matrix with

$$\mathbf{D}_2^{(k-1)}\left((r-1)L+l,(r-1)L+l\right) = \left(\|\mathbf{a}_{r,l}^{(k-1)}\|_2^2 + \|\mathbf{b}_{r,l}^{(k-1)}\|_2^2 + \eta^2\right)^{-1/2}. \quad (16)$$

With $\mathbf{V}_{\mathbf{A}}^{(k)} \triangleq \mathbf{P}^{(k)\mathrm{T}}{}_J\boldsymbol{\Xi}^{(k)}\mathbf{P}^{(k)}$ and $\mathbf{G}_{\mathbf{A}}^{(k)} \triangleq \mathbf{Y}_{(1)}^{(k)}{}_J\boldsymbol{\Xi}^{(k)}\mathbf{P}^{(k)}$, (15) is more compactly written as

$$\mathbf{A}^{(k)} = \mathbf{G}_{\mathbf{A}}^{(k)}\left(\mathbf{V}_{\mathbf{A}}^{(k)} + \lambda\mathbf{D}_2^{(k-1)}\right)^{-1}. \quad (17)$$

In addition, making use of the assumption that $\mathbf{B}^{(k)} \approx \mathbf{B}^{(k-1)}$ and recalling the definition of the permutation matrix $\mathbf{U}^{(k)}$, $\mathbf{P}^{(k)}$ can be approximately written as

$$\mathbf{P}^{(k)} \approx \mathbf{U}^{(k)\mathrm{T}}\begin{bmatrix}\mathbf{P}^{(k-1)} \\ \mathbf{B}^{(k-1)} \odot \boldsymbol{\gamma}^{(k)\mathrm{T}}\end{bmatrix}. \quad (18)$$

Since $\boldsymbol{\gamma}^{(k)}$ is a vector, the latter KR product can be equivalently written as $\mathbf{B}^{(k-1)} \odot \boldsymbol{\gamma}^{(k)\mathrm{T}} = \mathbf{B}^{(k-1)}(\mathrm{diag}(\boldsymbol{\gamma}^{(k)}) \otimes \mathbf{I}_L)$. Using the obvious order-recursive nature of $\boldsymbol{\Xi}$ and the definition of $\mathbf{U}^{(k)}$ readily leads to the following relation

$$\mathbf{U}^{(k)\mathrm{T}}{}_J\boldsymbol{\Xi}^{(k)}\mathbf{U}^{(k)} = \begin{bmatrix}\xi\cdot{}_J\boldsymbol{\Xi}^{(k-1)} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_J\end{bmatrix}. \quad (19)$$

From (18) and (7) and making use of (19) we can easily arrive at the following recursive formulas for updating $\mathbf{V}_{\mathbf{A}}^{(k)}$ and $\mathbf{G}_{\mathbf{A}}^{(k)}$:

$$\mathbf{V}_{\mathbf{A}}^{(k)} = \xi\mathbf{V}_{\mathbf{A}}^{(k-1)} + \left(\mathbf{B}^{(k-1)} \odot \boldsymbol{\gamma}^{(k)\mathrm{T}}\right)^{\mathrm{T}}\left(\mathbf{B}^{(k-1)} \odot \boldsymbol{\gamma}^{(k)\mathrm{T}}\right) \quad (20)$$

Algorithm 2: The O-BTD-RLS algorithm

Input: $\boldsymbol{\mathcal{Y}}$ in a streaming manner, $\xi, \lambda, \mu, R_{\mathrm{ini}}, L_{\mathrm{ini}}$
Output: Best (in the sense of (6)) BTD approximation of $\boldsymbol{\mathcal{Y}}^{(k)}$
Initialize $\mathbf{A}^{(0)}, \mathbf{B}^{(0)}, \mathbf{C}^{(0)}, \mathbf{V}_{\mathbf{A}}^{(0)}, \mathbf{V}_{\mathbf{B}}^{(0)}, \mathbf{G}_{\mathbf{A}}^{(0)}, \mathbf{G}_{\mathbf{B}}^{(0)}$ from Algorithm 1
for $k = 1, 2, \ldots$
  Compute $\mathbf{D}_1^{(k-1)}, \mathbf{D}_2^{(k-1)}$ from (13) and (16)
  Compute $\mathbf{S}^{(k-1)}$ from $\mathbf{A}^{(k-1)}$ and $\mathbf{B}^{(k-1)}$ (cf. (4))
  $\boldsymbol{\gamma}^{(k)} \leftarrow \left(\mathbf{S}^{(k-1)\mathrm{T}}\mathbf{S}^{(k-1)} + \mu\mathbf{D}_1^{(k-1)}\right)^{-1}\mathbf{S}^{(k-1)\mathrm{T}}\mathbf{y}^{(k)}$
  $\mathbf{V}_{\mathbf{A}}^{(k)} \leftarrow \xi\mathbf{V}_{\mathbf{A}}^{(k-1)} + \left(\mathbf{B}^{(k-1)} \odot \boldsymbol{\gamma}^{(k)\mathrm{T}}\right)^{\mathrm{T}}\left(\mathbf{B}^{(k-1)} \odot \boldsymbol{\gamma}^{(k)\mathrm{T}}\right)$
  $\mathbf{G}_{\mathbf{A}}^{(k)} \leftarrow \xi\mathbf{G}_{\mathbf{A}}^{(k-1)} + \mathbf{Y}^{(k)}\left(\mathbf{B}^{(k-1)} \odot \boldsymbol{\gamma}^{(k)\mathrm{T}}\right)$
  $\mathbf{A}^{(k)} \leftarrow \mathbf{G}_{\mathbf{A}}^{(k)}\left(\mathbf{V}_{\mathbf{A}}^{(k)} + \lambda\mathbf{D}_2^{(k-1)}\right)^{-1}$
  $\mathbf{V}_{\mathbf{B}}^{(k)} \leftarrow \xi\mathbf{V}_{\mathbf{B}}^{(k-1)} + \left(\boldsymbol{\gamma}^{(k)\mathrm{T}} \odot \mathbf{A}^{(k)}\right)^{\mathrm{T}}\left(\boldsymbol{\gamma}^{(k)\mathrm{T}} \odot \mathbf{A}^{(k)}\right)$
  $\mathbf{G}_{\mathbf{B}}^{(k)} \leftarrow \xi\mathbf{G}_{\mathbf{B}}^{(k-1)} + \mathbf{Y}^{(k)\mathrm{T}}\left(\boldsymbol{\gamma}^{(k)\mathrm{T}} \odot \mathbf{A}^{(k)}\right)$
  $\mathbf{B}^{(k)} \leftarrow \mathbf{G}_{\mathbf{B}}^{(k)}\left(\mathbf{V}_{\mathbf{B}}^{(k)} + \lambda\mathbf{D}_2^{(k-1)}\right)^{-1}$
end

$$\mathbf{G}_{\mathbf{A}}^{(k)} = \xi\mathbf{G}_{\mathbf{A}}^{(k-1)} + \mathbf{Y}^{(k)}\left(\mathbf{B}^{(k-1)} \odot \boldsymbol{\gamma}^{(k)\mathrm{T}}\right). \quad (21)$$

The update of $\mathbf{B}$ can be derived in an analogous manner (see [14] for details).

The resulting algorithm, called *Online BTD Reweighted Least Squares (RLS) (O-BTD-RLS)*, is tabulated as Algorithm 2. The common in the OTF literature initialization practice, namely initializing the online scheme with the result of applying the batch algorithm in the first few slices, can also be employed here. It should be stressed that, in the proposed algorithm, we respect the BTD model structure throughout, in contrast to works like [15] where the KR product structure of the slowly-varying part is only taken into account at the end of each recursion. One can readily verify that, in the most practical case of big low-rank tensors, that is $I, J \gg R, L$, the method requires a total storage of roughly $\mathcal{O}(IJR)$. while its per-step computational cost can be estimated to $\mathcal{O}(IJRL)$, that is, roughly the cost of analyzing the entire tensor in one batch divided by K.

V. Simulation Results

In this section, we evaluate the performance of the O-BTD-RLS algorithm in selecting and computing/tracking the correct BTD model for a given tensor. Comparisons with its batch counterpart are included, in terms of both approximation accuracy and time efficiency. We consider an $I \times J \times K$ tensor $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{X}} + \sigma\boldsymbol{\mathcal{N}}$, where $\boldsymbol{\mathcal{X}}$ is built as in (1), $\boldsymbol{\mathcal{N}}$ contains zero-mean, independent and identically distributed (i.i.d.) Gaussian entries of unit variance, and σ is set so that we get a given signal-to-noise ratio (SNR), with SNR in dB defined as $\mathrm{SNR} = 10\log_{10}\|\boldsymbol{\mathcal{X}}\|_{\mathrm{F}}^2/(\sigma^2\|\boldsymbol{\mathcal{N}}\|_{\mathrm{F}}^2)$. The entries of $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are also i.i.d. samples from the standard Gaussian distribution.

Experiment 1: We set $I = 40$, $J = 35$ and $K = 1250$. The true R is 5 and all L_rs are equal to 4. BTD-IRLS was run with overestimates of the true ranks, namely $R_{\mathrm{ini}} = 10$ and $L_{\mathrm{ini}} = 10$ for all blocks terms, using random initialization.[1]

[1] In the light of the robustness to initialization of BTD-HIRLS demonstrated in [2], we expect that BTD-IRLS will also enjoy such a desirable property. Further experimentation is required to confirm this.

TABLE I
COMPARISON OF BTD-IRLS AND O-BTD-RLS

	SNR (dB)						Average run-time (s)
	5		10		15		
	NMSE ($\times 10^{-3}$)	RE	NMSE ($\times 10^{-3}$)	RE	NMSE ($\times 10^{-3}$)	RE	
BTD-IRLS	0.5	0.2703	0.15	0.1560	0.05	0.085	10.8
O-BTD-RLS	1.2	0.2945	0.4	0.16	0.2	0.0886	2.8

O-BTD-RLS was initialized with the result of BTD-IRLS on the first 50 frontal slices and incrementally processed the remaining 1200 slices, with $\xi = 1$. For BTD-IRLS, λ was selected as $L(I+J)\hat{\sigma}$, where $\hat{\sigma}$ is an estimate of the noise standard deviation (in our experiments taken equal to the true σ), and either $\mu = 0.75KR\hat{\sigma}$ (at SNR=5 dB) or $\mu = 2KR\hat{\sigma}$ (at SNR=10 and 15 dB). For the example of SNR=10 dB, R was correctly recovered in 80% of the cases, while it was overestimated to only 6 in the rest. L was found correctly in all realizations. The regularization parameters of O-BTD-RLS were selected for simplicity equal to each other, $\lambda = \mu = L(I+J)\hat{\sigma}$. Table I shows the Relative Error (RE), $\left\|\boldsymbol{\mathcal{Y}} - \hat{\boldsymbol{\mathcal{X}}}\right\|_{\mathrm{F}} / \|\boldsymbol{\mathcal{Y}}\|_{\mathrm{F}}$, and the normalized mean squared error (NMSE) over the blocks, $\frac{1}{R}\sum_{r=1}^{R} \frac{\|\mathbf{A}_r\mathbf{B}_r^{\mathrm{T}} \circ \mathbf{c}_r - \hat{\mathbf{A}}_r\hat{\mathbf{B}}_r^{\mathrm{T}} \circ \hat{\mathbf{c}}_r\|_{\mathrm{F}}^2}{\|\mathbf{A}_r\mathbf{B}_r^{\mathrm{T}} \circ \mathbf{c}_r\|_{\mathrm{F}}^2}$, at different SNR values, for both the batch and incremental cases.

For computing the NMSE, the Hungarian algorithm was employed to resolve permutation ambiguities (as in [2]). In each case, the median over 50 independent realizations of the experiment is given. Clearly, the online scheme achieves an approximation accuracy close to that of the batch one (especially in terms of the RE), and the performance gap is diminished as SNR increases. In addition, this is achieved much more efficiently in terms of the average run-time. All simulations were run in a MacBook Pro, 2.6 GHz 6-Core Intel Core i7, 16 GB 2667 MHz DDR4 using Matlab v2019b.

Experiment 2: We consider a $40 \times 35 \times 5000$ tensor resulting by concatenating two smaller tensors with 3rd mode dimensions 2000 and 3000. The true ranks R and L_rs of the first tensor are 5 and 4 and those of the second one are 4 and 2. This implies an abrupt change of the underlying BTD model at the $k = 2001$ step. Noise is added for an SNR of 10 dB. Since the model is now time varying, a fading memory effect is simulated by setting ξ to 0.985 in O-BTD-RLS. All other specifications are as in the previous experiment. The normalized squared error (NSE) per frontal slice of one run of the online algorithm is plotted in Fig. 1 as a function of the number of update steps. Note that the algorithm immediately recognizes the model change and, after a number of steps required to collect information about the new model, re-adapts back to the new BTD model.

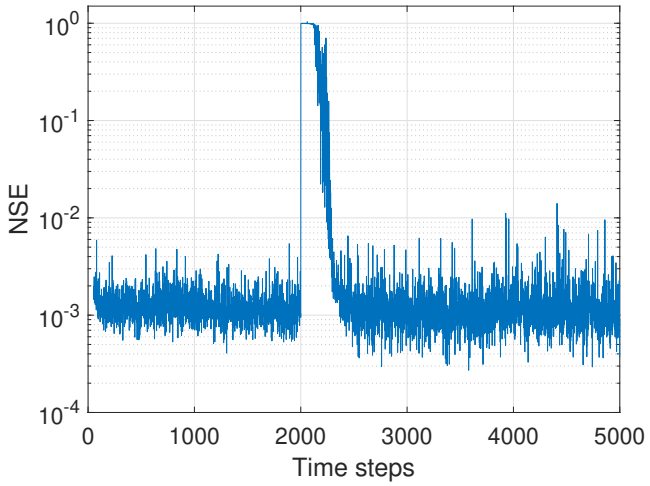

Fig. 1. Normalized squared error per frontal slice of O-BTD-RLS vs. time steps, at SNR=10 dB. The model changes abruptly at the $k = 2001$ step.

REFERENCES

[1] 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.

[2] A. A. Rontogiannis, E. Kofidis, and P. V. Giampouras, "Block-term tensor decomposition: Model selection and computation," *IEEE J. Sel. Topics Signal Process.*, Apr. 2021.

[3] C. Chatzichristos *et al.*, "Blind fMRI source unmixing via higher-order tensor decompositions," *J. Neurosci. Methods*, Mar. 2019.

[4] E. Dall'Anese *et al.*, "Optimization and learning with information streams," *IEEE Signal Process. Mag.*, May 2020.

[5] A. A. Rontogiannis, P. V. Giampouras, and K. D. Koutroumbas, "Online reweighted least squares robust PCA," *IEEE Signal Process. Lett.*, 2020.

[6] S. Zhou *et al.*, "Accelerating online CP decompositions for higher order tensors," in *Proc. KDD-2016*, San Francisco, CA, Aug. 2016.

[7] E. Gujral and E. E. Papalexakis, "OnlineBTD: Streaming algorithms to track the block term decomposition of large tensors," in *Proc. DSAA-2020*, Porto, Portugal, Oct. 2020.

[8] R. Pasricha, E. Gujral, and E. E. Papalexakis, "Identifying and alleviating concept drift in streaming tensor decomposition," in *Proc. ECML PKDD-2018*, Dublin, Ireland, Sep. 2018.

[9] I. W. Sanou *et al.*, "Online nonnegative canonical polyadic decomposition: Algorithms and application," in *Proc. EUSIPCO-2021*, Dublin, Ireland, Aug. 2021.

[10] S. Zhou, S. M. Erfani, and J. Bailey, "Online CP decomposition for sparse tensors," in *Proc. ICDM-2018*, Singapore, Nov. 2018.

[11] M. Mardani, G. Mateos, and G. B. Giannakis, "Subspace learning and imputation for streaming big data matrices and tensors," *IEEE Trans. Signal Process.*, May 2015.

[12] M. Hong *et al.*, "A unified algorithmic framework for block-structured optimization involving big data," *IEEE Signal Process. Mag.*, Jan. 2016.

[13] W. Hu *et al.*, "Incremental tensor subspace learning and its applications to foreground segmentation and tracking," *Int. J. Comput. Vis.*, 2011.

[14] A. A. Rontogiannis, E. Kofidis, and P. V. Giampouras, "Online rank-revealing block-term tensor decomposition," arXiv:2106.10755v2 [math.NA], Jun. 2021.

[15] D. Nion and N. D. Sidiropoulos, "Adaptive algorithms to track the PARAFAC decomposition of a third-order tensor," *IEEE Trans. Signal Process.*, Jun. 2009.