Rayleigh-Ritz based updates of the Multilinear Singular Value Decomposition

Vassilis Kalantzis and Panagiotis Traganitis

October 2023

EPrint ID: 2023.4

Rayleigh-Ritz based updates of the Multilinear Singular Value Decomposition

Vasileios Kalantzis* *and Panagiotis A. Traganitis*†
$\star$IBM Research, Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA
†Dept. of ECE, Michigan State University, East Lansing, MI 48824, USA

***Abstract*—Multilinear singular value decomposition (MLSVD), also known as Higher-order SVD (HOSVD), is a popular method for approximating a tensor of order ≥ 3 via a smaller core tensor and corresponding factor matrices. While MLSVD has found numerous applications, it is not designed to handle tensors that vary over time. In this work we propose an algorithm for updating the MLSVD of an evolving tensor via leveraging Rayleigh-Ritz matrix projection techniques. In particular, we consider the case where at each time step the dimensions of the tensor are augmented and its entries may change. Preliminary tests on synthetic and real data showcase the potential of the proposed approach.**

***Index Terms*—Tensor decomposition, online, multilinear singular value decomposition, rayleigh-ritz**

I. Introduction

Tensors, also known as higher order arrays, are mathematical constructs that extend the notion of matrices to more than 2 dimensions, and can represent a plethora of data, such as video, time-varying graphs, and recommendation datasets [1]. Tensor decompositions are invaluable tools that enable manipulation, analytics and inference from multimodal and tensor data [2], [3]. Popular decompositions include the Canonical Polyadic Decomposition (CPD), the Tucker decomposition, and the Multilinear or Higher-order Singular Value decomposition (MLSVD or HOSVD) [4], which is the focus of this work. A significant body of work has focused on creating computationally and memory efficient decomposition algorithms for static tensors. However, in some cases, such as real-time applications, the tensor of interest may change over time. If one needs to track the decomposition of such an evolving tensor, using algorithms developed for static tensors at each time step may incur prohibitive computational complexity, especially if changes occur with high frequency.

Recently, several approaches have been developed for decomposing evolving tensors. Tensor updates under the CPD model were considered in [5]. An algorithm to track the UTV decomposition of an evolving tensor was advocated for in [6]. A random sketching based approach to update the Tucker decomposition was proposed in [7]. Closer in spirit to this contribution, [8]–[10] utilize matrix subspace tracking approaches to update the MLSVD of a dynamic tensor.

In this work, we develop a novel matrix resolvent-based approach to dynamically update the MLSVD of a tensor as new slabs are added over time. Our novel approach updates the singular value decomposition of the matricizations of the tensor, via the Rayleigh-Ritz approximation procedure. The use of this judiciously designed Rayleigh-Ritz procedure enables our algorithm to accurately track the MLSVD of a time-varying tensor while maintaining low computational complexity, as it takes advantage of computations performed in previous time-steps. The proposed method is benchmarked on synthetic and real data, where it approaches the reconstruction accuracy of the baseline.

Notation. Unless otherwise noted, lowercase bold letters, $\boldsymbol{x}$, denote column vectors, uppercase bold letters, $\mathbf{X}$, represent matrices, underlined uppercase letters, $\underline{X}$, represent tensors, and calligraphic uppercase letters, $\mathcal{X}$, stand for sets. We adopt the notation of [4], [11], and the (i,j,k)-th entry of a tensor $\underline{X}$ is denoted by $\underline{X}(i,j,k)$. The (i,j)-th entry of matrix $\mathbf{X}$ is denoted by $\mathbf{X}(i,j)$; vec$(\mathbf{X})$ denotes a vector consisting of the stacked columns of $\mathbf{X}$; $\circ$ denotes the outer (tensor) product between two vectors or matrices; and $\otimes$ denotes the Kronecker product between two matrices. The Frobenius norm of a matrix $\mathbf{X}$ is denoted by $\|\mathbf{X}\|_F$. $^\top$ represents transpose; card$(\mathcal{A})$ denotes the cardinality, i.e. the number of elements, of set $\mathcal{A}$; $\mathbb{E}[\cdot]$ denotes expectation, and $\mathbb{1}(\mathcal{A})$ is the indicator function for the event $\mathcal{A}$, that takes value 1 when $\mathcal{A}$ occurs, and 0 otherwise.

II. Preliminaries and Problem Statement

While the analysis and results in this paper readily apply to tensors with arbitrary number of modes, for simplicity of exposition the remainder of this paper will focus on 3-order tensors. Consider a 3-order tensor $\underline{X}$ of dimension $I \times J \times K$ with entries $\underline{X}(i,j,k)$, $i = 1,\ldots,I, j = 1,\ldots,J, k = 1,\ldots,K$. The $I \times J$ matrix formed by considering only the k-th frontal slab of $\underline{X}$ is denoted as $\underline{X}(:,:,k)$. Accordingly the $I \times K$ k-th lateral and $J \times K$ horizontal slabs are denoted as $\underline{X}(:,k,:)$ and $\underline{X}(k,:,:)$ respectively. The tensor $\underline{X}$ can be represented as a matrix in three different ways, by concatenating (vectorized) slabs:

$$\begin{aligned}
\mathbf{X}_1 &= [\mathrm{vec}(\underline{X}(1,:,:)),\ldots,\mathrm{vec}(\underline{X}(I,:,:))],\\
\mathbf{X}_2 &= [\mathrm{vec}(\underline{X}(:,1,:)),\ldots,\mathrm{vec}(\underline{X}(:,J,:))],\\
\mathbf{X}_3 &= [\mathrm{vec}(\underline{X}(:,:,1)),\ldots,\mathrm{vec}(\underline{X}(:,:,K))].
\end{aligned}$$

The matrices $\mathbf{X}_1$, $\mathbf{X}_2$, and $\mathbf{X}_3$ are also called $matricizations$ of the tensor $\underline{X}$, with the matrix $\mathbf{X}_m$ representing the mode-m unfolding of $\underline{X}$. For a 3-rd order tensor $\mathbf{X}_1$ contains all the "rows" of $\underline{X}$ as columns, $\mathbf{X}_2$ contains all the columns of $\underline{X}$ as columns, and $\mathbf{X}_3$ collects all the frontal slabs of $\underline{X}$ as columns.

A. The Tucker decomposition and MLSVD

The Tucker decomposition posits that $\underline{X}$ can be written as

$$\underline{X} = \sum_{i=1}^{r_1}\sum_{j=1}^{r_2}\sum_{k=1}^{r_3} \underline{G}(i,j,k)\mathbf{a}_i \circ \mathbf{b}_j \circ \mathbf{c}_k,$$

where $\underline{G}$ is an $r_1 \times r_2 \times r_3$ "core" tensor, and $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are columns of the $I \times r_1$ matrix $\mathbf{A}$, the $J \times r_2$ matrix $\mathbf{B}$ and the $K \times r_3$ matrix $\mathbf{C}$ respectively. Thus, under the Tucker model, a 3rd-order tensor $\underline{X}$ is represented by a 3rd-order core tensor $\underline{G}$ and three so-called factor matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$. It can be shown that the matricizations of a tensor that adheres to the Tucker model take the following form

$$\begin{aligned}\mathbf{X}_1 &= (\mathbf{B} \otimes \mathbf{C})\mathbf{G}_1\mathbf{A}^\top,\\ \mathbf{X}_2 &= (\mathbf{A} \otimes \mathbf{C})\mathbf{G}_2\mathbf{B}^\top,\\ \mathbf{X}_3 &= (\mathbf{A} \otimes \mathbf{B})\mathbf{G}_3\mathbf{C}^\top \end{aligned} \quad (1)$$

with $\mathbf{G}_i$ denoting the i-th mode matricization of the core tensor $\underline{G}$. Typically, the factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and core tensor $\underline{G}$ are estimated using alternating least squares.

Multilinear or Higher-order SVD [12], [13] can be considered as a variant of the Tucker model, and provides a straightforward way, reminiscent of the matrix SVD, to compute the factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and the core tensor $\underline{G}$. Consider the rank-r_1, rank-r_2, and rank-r_3 SVDs of $\mathbf{X}_1$, $\mathbf{X}_2$ and $\mathbf{X}_3$ respectively,

$$\begin{aligned}\mathbf{X}_1 &= \mathbf{U}_1\mathbf{\Sigma}_1\mathbf{V}_1^\top,\\ \mathbf{X}_2 &= \mathbf{U}_2\mathbf{\Sigma}_2\mathbf{V}_2^\top,\\ \mathbf{X}_3 &= \mathbf{U}_3\mathbf{\Sigma}_3\mathbf{V}_3^\top.\end{aligned}$$

In particular, in MLSVD the factor matrices are chosen as orthonormal bases for the row, column and frontal slabs of $\underline{X}$. As such they can computed as the right singular vectors of the matricizations of $\mathbf{X}_1, \mathbf{X}_2$ and $\mathbf{X}_3$, that is

$$\mathbf{A} = \mathbf{V}_1, \quad \mathbf{B} = \mathbf{V}_2, \quad \mathbf{C} = \mathbf{V}_3$$

After computing the factor matrices the matricized version core tensor can be recovered in matrix form as

$$\mathbf{G}_1 = (\mathbf{B} \otimes \mathbf{C})^\top \mathbf{X}_1 \mathbf{A}.$$

The core tensor $\underline{G}$ can be recovered by appropriately reshaping $\mathbf{G}_1$. We compactly denote the MLSVD (or Tucker decomposition) of $\underline{X}$, with core tensor $\underline{G}$ and factor matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ as $\underline{X} = \{\underline{G}, \mathbf{A}, \mathbf{B}, \mathbf{C}\}$. When $r_1 = r_2 = r_3 = \ell$ the decomposition is called the rank-ℓ MLSVD of $\underline{X}$. Clearly, the complexity of MLSVD is dominated by the SVD computations required.

B. Problem statement

Suppose that the tensor $\underline{X}$ changes over time, and at each time step t, $n_t \in \mathbb{N}$ new frontal slabs are added. The newly added slabs can be collected in a $I \times J \times n_t$ tensor $\underline{E}_t$. The augmented $I \times J \times K_t$ tensor is denoted as $\underline{X}_t = [\underline{X}, \underline{E}_t]$, where $K_t = K_{t-1} + n_t$ denotes the time-varying third dimension. Note that tensor extensions along the horizontal and/or lateral directions can be treated in a similar fashion. The task of *updating the MLSVD of $\underline{X}_t$ at each time-step t* involves extracting the factor matrices $\mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t$ and core tensor $\underline{G}_t$. The next section will present our proposed method for updating the MLSVD of an evolving tensor.

III. Dynamic MLSVD

As the tensor changes with time, so do its matricizations. In particular, new rows are appended on $\mathbf{X}_1$, i.e. $\mathbf{X}_{1,t} = [\mathbf{X}_{1,t-1}^\top, \mathbf{E}_{1,t}^\top]^\top$, where $\mathbf{E}_{1,t}$ is a matrix containing the vectorization of the newly added slabs. Similarly, the mode-2 matricization of $\underline{X}_t$ can be also seen as equivalent to row augmentation of the corresponding mode-2 matricization of $\underline{X}$ up to column permutation, i.e., $\mathbf{X}_{2,t} = [\mathbf{X}_{2,t-1}^\top \mathbf{E}_{2,t}^\top]^\top$. On the other hand, the mode-3 matricization is updated via column additions as $\mathbf{X}_{3,t} = [\mathbf{X}_{3,t-1}\mathbf{E}_{3,t}]$. Nevertheless, by transposing $\mathbf{X}_{3,t-1}$, all tensor matricizations can be viewed as addition of new rows. As such for the remainder of this paper all matrix extensions are treated as additions of new rows[1] For brevity, the rest of this section will focus on a single time-step progression from step $t-1$ to step t. Also define $\underline{X} = \underline{X}_{t-1}$ and $\widehat{\underline{X}} = \underline{X}_t$.

A. Appending frontal slabs on 3-order tensors

Focusing on a single time-step update of a 3-mode tensor, Algorithm 1 sketches the general procedure for updating the MLSVD of a tensor. Here, the truncated MLSVD of the $I \times J \times K$ tensor $\underline{X}$ is updated to that of the augmented $I \times J \times (K+n)$ tensor $\widehat{\underline{X}} = [\underline{X}, \underline{E}]$, with the $I \times J \times n$ tensor $\underline{E}$ collecting the new slabs.

Algorithm 1 Rank ℓ MLSVD update for a 3-mode tensor.

```
1: Input: Rank $\ell$ MLSVD of the $I \times J \times K$ tensor $\underline{X} = \{\mathcal{G}, \mathbf{A}, \mathbf{B}, \mathbf{C}\}$, Rank $\ell$ SVDs of matricizations $\mathbf{X}_m = \mathbf{U}_m\mathbf{\Sigma}_m\mathbf{V}_m^\top, m = 1,2,3$, $I \times J \times n$ extension tensor $\underline{E}$, $\ell \in \mathbb{N}$
2: Output: Rank $\ell$ MLSVD of $\widehat{\underline{X}} = [\underline{X}, \underline{E}] = \{\widehat{\mathcal{G}}, \widehat{\mathbf{U}}_1, \widehat{\mathbf{U}}_2, \widehat{\mathbf{U}}_3\}$
3: For $m \in \{1,2,3\}$ Form the $m$-th matricization $\mathbf{X}_m$ of $\underline{X}$. Append to it the corresponding matricization $\mathbf{E}_m$ of $\underline{E}$.
4: For $m \in \{1,2,3\}$ set $\widehat{\mathbf{X}}_m = \begin{bmatrix}\mathbf{X}_m \\ \mathbf{E}_m\end{bmatrix}$.
5: For $m \in \{1,2,3\}$, approximate the updated rank-$\ell$ truncated SVD of the matrix $\widehat{\mathbf{X}}_m$ via Alg. 2
6: Recover $\widehat{\underline{G}}$ using $\widehat{\underline{X}}$ and $\widehat{\mathbf{U}}_1, \widehat{\mathbf{U}}_2, \widehat{\mathbf{U}}_3$.
```

The key step of Algorithm 1 is the extension of the matrix formed by the ℓ dominant left singular vectors of the m-th matricization of $\underline{X}$, $\mathbf{U}_m$ to the ℓ dominant left singular vectors of the i-th matricization $\widehat{\mathbf{X}}_m$ of $\widehat{\underline{X}}$, $\widehat{\mathbf{U}}_m$, $m = 1,2,3$. The standard approach to achieve this is to compute $\widehat{\mathbf{U}}_m$ from scratch per matricization of $\widehat{\underline{X}}$. Nonetheless, $\widehat{\mathbf{X}}_m$ contains $IJ(K+n)$ entries, of which IJK entries also appear in the

[1]Since we are dealing with row additions, it is convenient to work with the forward cyclic matricization of [14].

m-th matricization $\mathbf{X}_m$ of the tensor $\underline{X}$ for which ℓ dominant left singular vectors $\mathbf{U}_m$ are considered available. Following the above discussion, extending a $I \times J \times K$ tensor $\underline{X}$ by n frontal slabs requires the computation of the ℓ dominant left singular vectors of a sequence of matrices where each said matrix is a row/column extension of a matrix for which the ℓ dominant left singular vectors are available. Thus, one can think of re-using the information in $\mathbf{U}_m$ to "warm-start" the approximation of $\widehat{\mathbf{U}}_m$.

This problem is known as *SVD updating* and is a well-studied topic in computational linear algebra with applications in latent semantic indexing, recommender systems, and vision, to name a few [15], [16]. In this paper we exploit the approach outlined in [17] (see also [18]) and approximate $\widehat{\mathbf{U}}_m$ via a projection onto a judiciously crafted subspace. The projection should reduce the dimension of the problem significantly to reduce computational complexity, while capturing the critical subspace contained in $\widehat{\mathbf{X}}_m$. Note, the approach in [17] can be applied to both row and column update problems.

B. *Updating the SVD of a matricization*

To illustrate how the proposed algorithm works, we focus on the row augmentation scenario for a single matricization and a single time-step. Here, a $p \times n$ matrix $\mathbf{X}$, corresponding to a matricization of $\underline{X}$, is augmented to a $(p+s) \times n$ matrix $\widehat{\mathbf{X}}$ for $s > 0$. The case of column augmentation can be treated in the same way via transposition. Let $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ denote the rank-ℓ truncated SVD of $\mathbf{X}$ and let $\mathbf{E}$ be a row extension matrix collecting the data to be appended to $\mathbf{X}$. Then $\widehat{\mathbf{X}}^\top = [\mathbf{X}^\top \mathbf{E}^\top]^\top$.

Algorithm 2 Truncated SVD update (row extension).

```
1: Input: $\mathbf{X} \in \mathbb{R}^{p\times n}$, $\mathbf{E} \in \mathbb{R}^{s\times n}$, $\mathbf{U} \in \mathbb{R}^{r\times \ell}$, $\mathbf{\Sigma} \in \mathbb{R}^{\ell\times\ell}$, $\mathbf{V} \in \mathbb{R}^{n\times\ell}$, $\lambda \in \mathbb{R}$
2: Set $\mathbf{Z} = \begin{bmatrix} \mathbf{U} & \mathbf{Q} & \\ & & \mathbf{I} \end{bmatrix}$, where $\mathbf{Q}$ denotes an orthonormal basis of $(\mathbf{I} - \mathbf{U}\mathbf{U}^\top)(\lambda\mathbf{I} - \mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{E}^\top$
3: Set $\widehat{\mathbf{X}} = \begin{bmatrix} \mathbf{X} \\ \mathbf{E} \end{bmatrix}$
4: Compute the rank-$\ell$ truncated SVD of $\mathbf{Z}^\top\widehat{\mathbf{X}} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^\top$
5: Set $\widehat{\mathbf{U}} = \mathbf{Z}\tilde{\mathbf{U}}$
6: Set $\widehat{\mathbf{\Sigma}} = \tilde{\mathbf{\Sigma}}$
7: Set $\widehat{\mathbf{V}} = \widehat{\mathbf{X}}^\top\widehat{\mathbf{U}}\widehat{\mathbf{\Sigma}}^{-1}$
8: Output: $\widehat{\mathbf{U}}, \widehat{\mathbf{\Sigma}}, \widehat{\mathbf{V}}$
```

The proposed truncated SVD updating mechanism for each matricization is based on the Rayleigh-Ritz approximation [19] and is outlined in Algorithm 2 (see also [17]). Per the Rayleigh-Ritz procedure, the rank ℓ SVD of $\widehat{\mathbf{X}}$ can be approximated via the rank ℓ SVD of the dimensionality reduced matrix $\mathbf{Z}^\top\widehat{\mathbf{X}}$, for an appropriate choice of $\mathbf{Z}$. Denote the rank-ℓ SVD of the reduced matrix as $\mathbf{Z}^\top\widehat{\mathbf{X}} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^\top$. A proper choice for the projection matrix $\mathbf{Z}$ is

$$\mathbf{Z} = \begin{bmatrix} \mathbf{U} & \mathbf{Q} & \\ & & \mathbf{I} \end{bmatrix}, \tag{2}$$

where $\mathbf{Q}$ is an orthonormal basis of $(\mathbf{I} - \mathbf{U}\mathbf{U}^\top)(\lambda\mathbf{I} - \mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{E}^\top$, $\mathbf{I}$ is the identity matrix of dimension $s \times s$ and λ is a scalar that is larger than the square of the leading singular value of the matrix $\widehat{\mathbf{X}}$ [18]. Such a λ can be obtained via Gershgorin's circle theorem [19].

To see why $\mathbf{Z}$ is a prudent choice for projection basis, let $(\widehat{\sigma}, \widehat{\mathbf{u}})$ denote one of the ℓ dominant singular values and associated left singular vector of the matrix $\widehat{\mathbf{X}}$, where $\widehat{\mathbf{u}} = [\widehat{\mathbf{r}}^\top, \widehat{\mathbf{y}}^\top]^\top, \widehat{\mathbf{r}} \in \mathbb{R}^p,\ \widehat{\mathbf{y}} \in \mathbb{R}^s$. Then, one can write

$$\begin{bmatrix} \mathbf{X}\mathbf{X}^\top - \widehat{\sigma}\mathbf{I} & \mathbf{X}\mathbf{E}^\top \\ \mathbf{E}\mathbf{X}^\top & \mathbf{E}\mathbf{E}^\top - \widehat{\sigma}\mathbf{I} \end{bmatrix} \begin{bmatrix} \widehat{\mathbf{r}} \\ \widehat{\mathbf{y}} \end{bmatrix} = \mathbf{0} \tag{3}$$

from which it follows $\widehat{\mathbf{r}} = (\widehat{\sigma}\mathbf{I} - \mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{E}^\top\widehat{\mathbf{y}}$, and thus

$$\begin{aligned} \widehat{\mathbf{u}} = \begin{bmatrix} \widehat{\mathbf{r}} \\ \widehat{\mathbf{y}} \end{bmatrix} &\in \operatorname{Ran}\left(\begin{bmatrix} (\widehat{\sigma}\mathbf{I} - \mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{E}^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \right) \\ &\in \operatorname{Ran}\left(\begin{bmatrix} \mathbf{U} & (\mathbf{I} - \mathbf{U}\mathbf{U}^\top)(\widehat{\sigma}\mathbf{I} - \mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{E}^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \right). \end{aligned}$$

The particular choice of the matrix $\mathbf{Z}$ corresponds to a zero-order approximation of $(\mathbf{I} - \mathbf{U}\mathbf{U}^\top)(\widehat{\sigma}\mathbf{I} - \mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{E}^\top$ around a scalar $\lambda > \widehat{\sigma}$.

Finally, the approximate rank-ℓ truncated SVD of the matrix $\widehat{\mathbf{X}}$ is recovered by computing the ℓ dominant left and singular vectors as $\widehat{\mathbf{U}} = \mathbf{Z}\tilde{\mathbf{U}}$ and $\widehat{\mathbf{V}} = \widehat{\mathbf{X}}^\top\widehat{\mathbf{U}}\widehat{\mathbf{\Sigma}}^{-1}$. The entire procedure is outlined in Alg. 2. Alg. 2 receives as inputs a matricization $\mathbf{X}$, along with its rank-ℓ truncated SVD $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, the row extension matrix $\mathbf{E}$ and an estimate λ. Note, Alg. 2 is performed on all matricizations of a tensor $\underline{X}$. Alg. 1 is repeated for each update that occurs on the tensor of interest.

IV. Numerical tests

The performance of the proposed MLSVD update scheme is evaluated in this section. The tensors considered are updated by appending new frontal slabs to them at each time-step. All experiments are conducted in a MATLAB environment [11], using 64-bit arithmetic, on a single core of a computing system equipped with an 2.3 GHz 8-Core Intel Core i9 processor and 64 GB of DDR4 system memory.

Two datasets are considered: a synthetic and a real one. The first dataset is a synthetic 3-mode tensor $\underline{X}$ of size $60 \times 60 \times 50$ with random entries. The tensor $\underline{X}$ is extended to the tensor $\underline{\widehat{X}} = [\underline{X}, \underline{E}]$ of size $60 \times 60 \times 60$, by augmenting the third dimension of $\underline{X}$ with ten random matrices of size 60×60. The second dataset consists of subset of a hyperspectral scenes gathered by Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) sensor [2] across a wavelength range of 400 to 2500 nm over the Indian Pines test site in North-western Indiana [20]. This dataset consists of 220 145×145 images, each captured in a different wavelength by the AVIRIS sensor. A visualization of the dataset can be seen in Figure 1. The initial tensor $\underline{X}$ is created by retaining the $145 \times 145 \times 200$ subtensor of the original dataset and setting the remaining $145 \times 145 \times 20$ as the update tensor $\underline{E}$.

[2]https://aviris.jpl.nasa.gov/

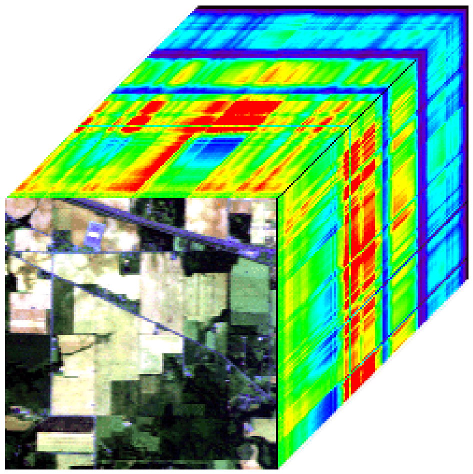

Fig. 1: Hyperspectral Image of the Indian Pines dataset [20].

For both datasets, our goal is to exploit the rank-ℓ MLSVD of $\underline{X}$ to approximate a rank-ℓ MLSVD of $\underline{\widehat{X}}$ without computing the individual rank-ℓ truncated SVDs of each different matricization of $\underline{\widehat{X}}$ from scratch. The figure of merit for both datasets is thee relative approximation error $\|\underline{\widehat{X}} - \underline{T}\|_F / \|\underline{\widehat{X}}\|_F$ where $\underline{T}$ denotes the rank-ℓ MLSVD of $\underline{\widehat{X}}$. The approximation $\underline{T}$ is formed using three different approaches:

- **A1**: Computing the rank-ℓ MLSVD of $\underline{X}$ from scratch at each timestep,
- **A2**: dynamically update the rank-ℓ MLSVD of $\underline{X}$ via Algorithm 2
- **A3**: dynamically update the rank-ℓ MLSVD of $\underline{X}$ via Algorithm 2 with $\mathbf{Z} = \begin{bmatrix} \mathbf{U} & \\ & \mathbf{I} \end{bmatrix}$.

Figures 2 and 3 plot the approximation error for synthetic and real data, respectively. As expected, **A1** enjoys the highest accuracy since it computes the exact rank-ℓ MLSVD at each time step. On the other hand, **A2** and **A3** result to a less accurate approximation, which, however, can be obtained with markedly reduced complexity, especially when the initial tensor $\underline{X}$ is similar in size to $\underline{\widehat{X}}$. Among the two inexact approaches, the resolvent-based approach **A2** leads to higher accuracy.

V. Conclusions

In this paper we considered the problem of updating the truncated MLSVD of a tensor subject to the addition of frontal slabs. In this setting, the computation of the MLSVD of the augmented tensor requires the computation of the truncated SVD of all possible matricizations of the tensor and each such matricization is by itself a row/column extension of the matricizations prior to the tensor update. Our algorithm presented a framework to exploit the truncated SVD of the matricizations prior to the tensor extension in order to approximate the truncated SVD of the updated matricizations. Our framework is based on Rayleigh-Ritz projections where the truncated SVD of a matrix is updated via projection onto a low-dimensional subspace. Several options to set the projection subspace were presented, including a scheme based on matrix resolvent expansions. Numerical experiments confirm that the proposed algorithm can track the error of the baseline approach closely.

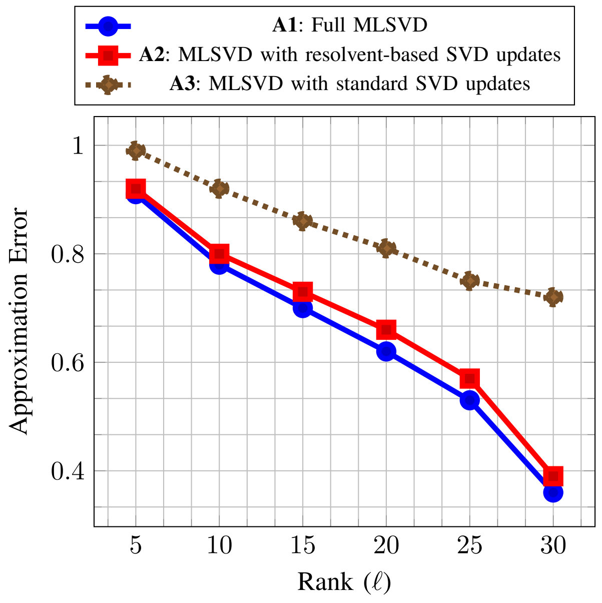

Fig. 2: Approximation error of a synthetic tensor of size $60 \times 60 \times 60$ by a rank-ℓ MLSVD approximation.

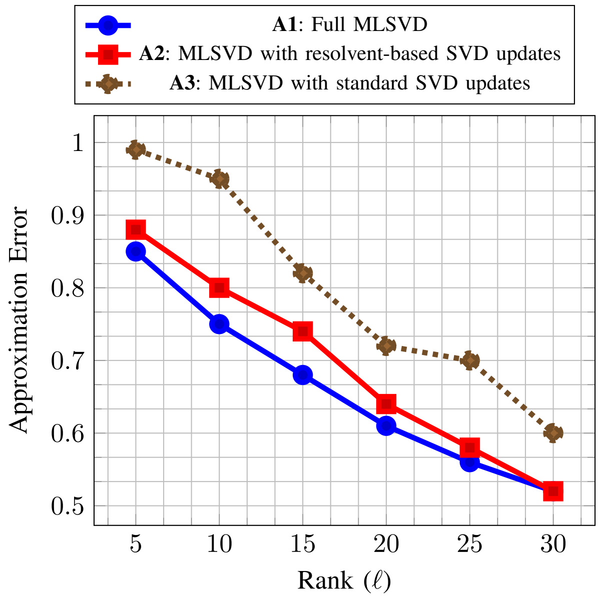

Fig. 3: Approximation error of the Indian Pines dataset by a rank-ℓ approximation.

References

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

[2] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE Signal Processing Magazine*, vol. 32, no. 2, pp. 145–163, 2015.

[3] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky, “Tensor decompositions for learning latent variable models,” *Journal of Machine Learning Research*, vol. 15, no. 80, pp. 2773–2832, 2014. [Online]. Available: http://jmlr.org/papers/v15/anandkumar14b.html

[4] 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 Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, 2017.

[5] D. Nion and N. D. Sidiropoulos, “Adaptive algorithms to track the parafac decomposition of a third-order tensor,” *IEEE Transactions on Signal Processing*, vol. 57, no. 6, pp. 2299–2310, 2009.

[6] M. Vandecappelle and L. De Lathauwer, “Updating the multilinear utv decomposition,” *IEEE Transactions on Signal Processing*, vol. 70, pp. 3551–3565, 2022.

[7] Y. Sun, Y. Guo, C. Luo, J. Tropp, and M. Udell, “Low-rank tucker approximation of a tensor from streaming data,” *SIAM Journal on Mathematics of Data Science*, vol. 2, no. 4, pp. 1123–1150, 2020. [Online]. Available: https://doi.org/10.1137/19M1257718

[8] W. Hu, X. Li, X. Zhang, X. Shi, S. Maybank, and Z. Zhang, “Incremental tensor subspace learning and its applications toâ foreground segmentation and tracking,” *International Journal of Computer Vision*, vol. 91, no. 3, pp. 303–327, Feb 2011. [Online]. Available: https://doi.org/10.1007/s11263-010-0399-6

[9] A. Sobral, C. G. Baker, T. Bouwmans, and E.-h. Zahzah, “Incremental and multi-feature tensor subspace learning applied for background modeling and subtraction,” in *Image Analysis and Recognition*, A. Campilho and M. Kamel, Eds. Springer International Publishing, 2014, pp. 94–103.

[10] Y. Cheng, F. Roemer, O. Khatib, and M. Haardt, “Tensor subspace tracking via kronecker structured projections (tetrakron) for time-varying multidimensional harmonic retrieval,” *EURASIP Journal on Advances in Signal Processing*, vol. 2014, no. 1, p. 123, Aug 2014. [Online]. Available: https://doi.org/10.1186/1687-6180-2014-123

[11] MATLAB, *version 9.14.0 (R2023a)*. Natick, Massachusetts: The MathWorks Inc., 2023.

[12] L. De Lathauwer, B. De Moor, and J. Vandewalle, “A multilinear singular value decomposition,” *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2000. [Online]. Available: https://doi.org/10.1137/S0895479896305696

[13] G. Bergqvist and E. G. Larsson, “The higher-order singular value decomposition: Theory and an application [lecture notes],” *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 151–154, 2010.

[14] B. W. Bader and T. G. Kolda, “Algorithm 862: Matlab tensor classes for fast algorithm prototyping,” *ACM Transactions on Mathematical Software (TOMS)*, vol. 32, no. 4, pp. 635–653, 2006.

[15] H. Zha and H. D. Simon, “On updating problems in latent semantic indexing,” *SIAM Journal on Scientific Computing*, vol. 21, no. 2, pp. 782–791, 1999.

[16] A. N. Nikolakopoulos, V. Kalantzis, E. Gallopoulos, and J. D. Garofalakis, “Eigenrec: generalizing PureSVD for effective and efficient top-N recommendations,” *Knowledge and Information Systems*, vol. 58, no. 1, pp. 59–81, 2019.

[17] V. Kalantzis, G. Kollias, S. Ubaru, A. N. Nikolakopoulos, L. Horesh, and K. Clarkson, “Projection techniques to update the truncated svd of evolving matrices with applications,” in *International Conference on Machine Learning*. PMLR, 2021, pp. 5236–5246.

[18] V. Kalantzis and P. A. Traganitis, “Matrix resolvent eigenembeddings for dynamic graphs,” in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.

[19] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 2013.

[20] B. Rasti, “Sparse hyperspectral image modeling and restoration,” Ph.D. dissertation, Department of Electrical and Computer Engineering, University of Iceland, 2014.