ALTERNATING PROXIMAL GRADIENT METHOD FOR SPARSE NONNEGATIVE TUCKER DECOMPOSITION

YANGYANG XU*

Abstract. Multi-way data arises in many applications such as electroencephal-ography (EEG) classification, face recognition, text mining and hyperspectral data analysis. Tensor decomposition has been commonly used to find the hidden factors and elicit the intrinsic structures of the multi-way data. This paper considers sparse nonnegative Tucker decomposition (NTD), which is to decompose a given tensor into the product of a core tensor and several factor matrices with sparsity and nonnegativity constraints. An alternating proximal gradient method (APG) is applied to solve the problem. The algorithm is then modified to sparse NTD with missing values. Per-iteration cost of the algorithm is estimated scalable about the data size, and global convergence is established under fairly loose conditions. Numerical experiments on both synthetic and real world data demonstrate its superiority over a few state-of-the-art methods for (sparse) NTD from partial and/or full observations. The MATLAB code along with demos are accessible from the author's homepage.

Key words. multi-way data, sparse nonnegative Tucker decomposition, alternating proximal gradient method, non-convex optimization, sparse optimization

1. Introduction. A *tensor* is a multi-dimensional array. For example, a *vector* is a first-order tensor, and a *matrix* is a second-order tensor. The order of a tensor is the number of dimensions, also called *way* or *mode*. Tensors naturally arise in the applications that collect data along multiple dimensions, including space, time, and spectrum, from different subjects (e.g., patients), under varying conditions, and in different modalities. They can also be created by tensorization of lower dimensional data [6]. Examples include medical data (CT, MRI, EEG), text data and hyperspectral images. An efficient approach to elicit the intrinsic structure of multi-dimensional data is tensor decomposition. Two commonly used tensor decompositions are CANDECOMP/PARAFAC decomposition (CPD) [5, 13] and Tucker decomposition (TD) [33]. CPD decomposes an Nth-order tensor $\mathcal{M}$ into the product of N factor matrices $\mathbf{A}_1, \cdots, \mathbf{A}_N$, and TD decomposes $\mathcal{M}$ into the product of a core tensor $\mathcal{C}$ and N factor matrices $\mathbf{A}_1, \cdots, \mathbf{A}_N$.

This paper focuses on sparse nonnegative Tucker decomposition (NTD) [18], which imposes nonnegativity and uses ℓ_1-regularization terms to promote sparsity structure on the core tensor and/or factor matrices. Nonnegativity allows only additivity, so the solutions are often intuitive to understand and explain. Promoting the sparsity of the core tensor aims at improving the interpretability of the solutions. Roughly speaking, the core tensor interacts with all the factor matrices, and a simple one is often preferred [15]. Consider a three-way tensor, for example. The $(1, 1, 1)$-th component of the core tensor couples the first columns of three factor matrices together. If it is not *zero*, then the three columns interacts with each other. Otherwise, they have no or only weak relations. Forcing the core tensor to be sparse can often keep strong interactions between the factor matrices and remove the weak ones. Sparse factor matrices make the decomposed parts more meaningful and can enhance uniqueness as explained in [26]. Sparse NTD has found a large number of applications such as in EEG classification [8], hyperspectral data analysis [38], text mining [26], face recoginition [37], and so on.

1.1. Related work. NTD is a highly non-convex problem, and sparse regularizers make the problem even harder. A natural and often efficient way to solve the problem is to alternatingly update the core tensor and factor matrices. It includes, but not limited to, alternating least squares method (ALS) [12], column-wise coordinate descent (CCD) [24], higher-order multiplicative update (HONMF) [26], and hierarchical alternating least squares (HALS) [28]. ALS alternatingly updates the core tensor and factor matrices by solving a sequence of nonnegative least squares (NLS) problems, which requires to calculate matrix inverse

---

and make ALS unsuitable for large-scale problems[1]. For this reason, [12] simply restricts the core tensor to be super-diagonal in its numerical tests. CCD has closed form update for each column of a factor matrix. However, to update the core tensor, it still requires to solve a big NLS problem, which makes CCD unsuitable for large-scale problems either. HONMF is an extension of the multiplicative update method in [21] for nonnegative matrix factorization [20, 27] and has a relatively low per-iteration cost. At each iteration, it only needs some tensor-matrix multiplications and component-wise divisions. The drawback of HONMF is its slow convergence, which makes the algorithm often run a large number of iterations to reach an acceptable data fitting. Like ALS, HALS needs to solve a sequence of NLS problems, but it updates factor matrices in a column-wise way and the core tensor component-wisely, which enables closed form solutions for all subproblems. In addition, HALS often converges faster than HONMF. However, as shown in [29], the convergence speed of HALS is still not satisfying.

There are also algorithms that update the core tensor and factor matrices simultaneously, such as the damped Gauss-Newton method (dGN) in [29]. It is demonstrated that dGN overwhelmingly outperforms HONMF and HALS in terms of convergence speed.

Recently, [35] proposed an alternating proximal gradient method (APG) for solving NCP, and it was observed superior to some other algorithms such as the alternating direction method of multiplier (ADMM) [39] and alternating nonnegative least squares method (ANLS) [16, 17] in both speed and solution quality. Unlike ANLS that exactly solves each subproblem, APG updates every factor matrix by solving a relaxed subproblem with a separable quadratic objective. Each relaxed subproblem has a closed form solution, which makes low per-iteration cost. Using an extrapolation technique, APG also converges very fast.

1.2. Overview of tensor. Notation. We use small letters $a, x, \cdots$ for scalars, bold small letters $\mathbf{a}, \mathbf{x}, \cdots$ for vectors, bold capital letters $\mathbf{A}, \mathbf{B}, \cdots$ for matrices and bold caligraphic letters $\mathcal{C}, \mathcal{M}, \cdots$ for tensors. The components of a tensor $\mathcal{X}$ are written in the form of $x_{i_1 i_2 \cdots i_N}$, which denotes the $(i_1, i_2, \cdots, i_N)$-th component of $\mathcal{X}$.

Before proceeding with the model, we overview some tensor related concepts. For more details, we refer the readers to the nice review paper [19].

- A *fiber* of $\mathcal{X}$ is a vector obtained by fixing all indices of $\mathcal{X}$ except one.
- The *vectorization* of $\mathcal{X}$ gives a vector, which is obtained by stacking all mode-1 fibers of $\mathcal{X}$ and denoted by $\text{vec}(\mathcal{X})$.
- The mode-n *matricization* of $\mathcal{X}$ is a matrix denoted by $\mathbf{X}_{(n)}$ whose columns are mode-n fibers of $\mathcal{X}$ in the lexicographical order.
- The mode-n product of $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ with $\mathbf{A} \in \mathbb{R}^{J \times I_n}$ is written as $\mathcal{X} \times_n \mathbf{A} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N}$, defined component-wisely by

$$(\mathcal{X} \times_n \mathbf{A})_{i_1 \cdots i_{n-1} j i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \cdots i_N} a_{j i_n}.$$

- The *inner product* of $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is $\langle \mathcal{A}, \mathcal{B} \rangle \triangleq \sum_{i_1, \cdots, i_N} a_{i_1 \cdots i_N} b_{i_1 \cdots i_N}$. The Frobenious norm of $\mathcal{X}$ is $\|\mathcal{X}\|_F \triangleq \sqrt{\langle \mathcal{X}, \mathcal{X} \rangle}$.
- Given $\mathcal{M} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, the *Tucker decomposition* of $\mathcal{M}$ is to find a core tensor $\mathcal{C} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$ with $R_n \leq I_n, \forall n$ and N factor matrices $\mathbf{A}_n \in \mathbb{R}^{I_n \times R_n}, n = 1, \cdots, N$ such that

$$\mathcal{M} \approx \mathcal{C} \times_1 \mathbf{A}_1 \cdots \times_N \mathbf{A}_N. \tag{1.1}$$

---

[1]There appears no exact definition of "large-scale". The concept can involve with the development of the computing power. Here, we roughly mean there are over millions of variables or data values.

It is not difficult to verify that if $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{C}} \times_1 \mathbf{A}_1 \cdots \times_N \mathbf{A}_N$, then

$$\text{vec}(\boldsymbol{\mathcal{X}}) = \left(\otimes_{n=N}^{1} \mathbf{A}_n \right) \text{vec}(\boldsymbol{\mathcal{C}}), \tag{1.2}$$

where

$$\otimes_{n=N}^{1} \mathbf{A}_n \triangleq \mathbf{A}_N \otimes \cdots \otimes \mathbf{A}_1, \tag{1.3}$$

and $\mathbf{A} \otimes \mathbf{B}$ denotes the Kronecker product of $\mathbf{A}$ and $\mathbf{B}$. In addition,

$$\mathbf{X}_{(n)} = \mathbf{A}_n \mathbf{C}_{(n)} \left(\otimes_{\substack{i=N \\ i \neq n}}^{1} \mathbf{A}_i \right)^\top . \tag{1.4}$$

1.3. Contributions. We apply and improve the APG method proposed in [35] to the sparse NTD problem

$$\min_{\boldsymbol{\mathcal{C}}, \mathbf{A}} F(\boldsymbol{\mathcal{C}}, \mathbf{A}) \equiv \ell(\boldsymbol{\mathcal{C}}, \mathbf{A}) + \lambda_c \|\boldsymbol{\mathcal{C}}\|_1 + \sum_{n=1}^{N} \lambda_n \|\mathbf{A}_n\|_1,$$
$$\text{s.t. } \boldsymbol{\mathcal{C}} \in \mathbb{R}_+^{R_1 \times \cdots \times R_N}, \mathbf{A}_n \in \mathbb{R}_+^{I_n \times R_n}, \ n = 1, \cdots, N, \tag{1.5}$$

where $\mathbb{R}_+^{I_n \times R_n}$ contains all $I_n \times R_n$ matrices with nonnegative components, $\mathbf{A}$ denotes $(\mathbf{A}_1, \cdots, \mathbf{A}_N)$,

$$\ell(\boldsymbol{\mathcal{C}}, \mathbf{A}) = \frac{1}{2} \|\boldsymbol{\mathcal{C}} \times_1 \mathbf{A}_1 \cdots \times_N \mathbf{A}_N - \boldsymbol{\mathcal{M}}\|_F^2$$

is a data fitting term that measures the approximation in (1.1), $\boldsymbol{\mathcal{M}} \in \mathbb{R}_+^{I_1 \times \cdots \times I_N}$ is a given tensor, $\|\boldsymbol{\mathcal{C}}\|_1 \triangleq \sum_{i_1, \cdots, i_N} |c_{i_1 \cdots i_N}|$ is used to promote the sparsity of $\boldsymbol{\mathcal{C}}$, and $\lambda_c, \lambda_1, \cdots, \lambda_N$ are parameters balancing the data fitting and sparsity level.

Our algorithm iteratively updates the core tensor $\boldsymbol{\mathcal{C}}$ and factor matrices alternatingly in the order of $\boldsymbol{\mathcal{C}}, \mathbf{A}_1, \boldsymbol{\mathcal{C}}, \mathbf{A}_2, \cdots, \boldsymbol{\mathcal{C}}, \mathbf{A}_N$. We analyze the algorithm's per-iteration complexity and give its global convergence. The algorithm is modified to sparse NTD with missing values. We also consider some extensions of NTD including sparse higher-order principal component analysis [1]. Our algorithm is carefully implemented in MATLAB and compared to some state-of-the-art methods for solving (sparse) NTD from partial and/or full observations on both synthetic and real world data. Numerical results show that the proposed algorithm makes superior performance over all the compared ones in almost all cases.

1.4. Outline. The rest of the paper is organized as follows. Section 2 applies APG to sparse NTD problem. The algorithm is modified for sparse NTD with missing values in section 3, and some extensions are considered in section 4. Numerical results are shown in section 5. Finally, section 6 concludes the paper.

2. Sparse nonnegative Tucker decomposition.

2.1. Bound constraints for well-definedness. Note that for any positive scalars $s_c, s_1, \cdots, s_N$ such that their product equals *one*, $(s_c \boldsymbol{\mathcal{C}}, s_1 \mathbf{A}_1, \cdots, s_N \mathbf{A}_N)$ does not change the value of ℓ. Hence, if some λ's vanish, the corresponding variables would be unbounded such that the variables with positive λ's would approach to *zero*, and (1.5) may not admit a solution. To tackle this problem, if $\lambda_n = 0$, we add

$$\mathbf{A}_n \leq \max(1, \|\boldsymbol{\mathcal{M}}\|_\infty) \tag{2.1}$$

to bound $\mathbf{A}_n$, where $\|\boldsymbol{\mathcal{M}}\|_\infty$ denotes the maximum component of $\boldsymbol{\mathcal{M}}$. If $\lambda_c = 0$, we add

$$\boldsymbol{\mathcal{C}} \leq \max(1, \|\boldsymbol{\mathcal{M}}\|_\infty) \tag{2.2}$$

to bound $\boldsymbol{\mathcal{C}}$. The constraints in (2.1) and (2.2) are reasonable according to the following proposition, which is not difficult to show.

PROPOSITION 2.1. *If $\boldsymbol{\mathcal{M}} = \tilde{\boldsymbol{\mathcal{C}}} \times_1 \tilde{\mathbf{A}}_1 \cdots \times_N \tilde{\mathbf{A}}_N$ for some $(\tilde{\boldsymbol{\mathcal{C}}}, \tilde{\mathbf{A}}_1, \cdots, \tilde{\mathbf{A}}_N)$, then there exists some $(\boldsymbol{\mathcal{C}}, \mathbf{A}_1, \cdots, \mathbf{A}_N)$ satisfying (2.1) and (2.2) such that $\boldsymbol{\mathcal{M}} = \boldsymbol{\mathcal{C}} \times \mathbf{A}_1 \cdots \times_N \mathbf{A}_N$ and $(\boldsymbol{\mathcal{C}}, \mathbf{A}_1, \cdots, \mathbf{A}_N)$ has the same sparsity as that of $(\tilde{\boldsymbol{\mathcal{C}}}, \tilde{\mathbf{A}}_1, \cdots, \tilde{\mathbf{A}}_N)$.*

REMARK 2.1. *If $\tilde{\boldsymbol{\mathcal{C}}} \times_1 \tilde{\mathbf{A}}_1 \cdots \times_N \tilde{\mathbf{A}}_N$ is not exactly equal but close to $\boldsymbol{\mathcal{M}}$, one can magnify the bounds in (2.1) and (2.2) by multiplying some $\tau > 1$.*

2.2. APG for sparse NTD. For convenience, we assume all λ's to be positive in the derivation of our algorithm, so there are no constraints as in (2.1) and (2.2) present. Our algorithm is based on the APG method proposed in [35]. Suppose the current iterate is $(\tilde{\boldsymbol{\mathcal{C}}}, \tilde{\mathbf{A}})$. We update $\boldsymbol{\mathcal{C}}$ by

$$\boldsymbol{\mathcal{C}}_{\text{new}} = \underset{\boldsymbol{\mathcal{C}} \geq 0}{\arg\min} \langle \nabla_{\boldsymbol{\mathcal{C}}} \ell(\hat{\boldsymbol{\mathcal{C}}}, \tilde{\mathbf{A}}), \boldsymbol{\mathcal{C}} - \hat{\boldsymbol{\mathcal{C}}} \rangle + \frac{L_c}{2} \|\boldsymbol{\mathcal{C}} - \hat{\boldsymbol{\mathcal{C}}}\|_F^2 + \lambda_c \|\boldsymbol{\mathcal{C}}\|_1, \tag{2.3}$$

$$= \max \left(0, \hat{\boldsymbol{\mathcal{C}}} - \frac{1}{L_c} \nabla_{\boldsymbol{\mathcal{C}}} \ell(\hat{\boldsymbol{\mathcal{C}}}, \tilde{\mathbf{A}}) - \frac{\lambda_c}{L_c} \right), \tag{2.4}$$

where L_c is a Lipschitz constant of $\nabla_{\boldsymbol{\mathcal{C}}} \ell(\boldsymbol{\mathcal{C}}, \tilde{\mathbf{A}})$ with respect to $\boldsymbol{\mathcal{C}}$, namely,

$$\|\nabla_{\boldsymbol{\mathcal{C}}} \ell(\boldsymbol{\mathcal{C}}_1, \tilde{\mathbf{A}}) - \nabla_{\boldsymbol{\mathcal{C}}} \ell(\boldsymbol{\mathcal{C}}_2, \tilde{\mathbf{A}})\|_F \leq L_c \|\boldsymbol{\mathcal{C}}_1 - \boldsymbol{\mathcal{C}}_2\|_F, \ \forall \ \boldsymbol{\mathcal{C}}_1, \boldsymbol{\mathcal{C}}_2,$$

and $\hat{\boldsymbol{\mathcal{C}}}$ is an extrapolated point. Similarly, if the current iterate is $(\tilde{\boldsymbol{\mathcal{C}}}, \tilde{\mathbf{A}})$, a factor matrix $\mathbf{A}_n$ is updated by

$$(\mathbf{A}_n)_{\text{new}} = \underset{\mathbf{A}_n \geq 0}{\arg\min} \langle \nabla_{\mathbf{A}_n} \ell(\tilde{\boldsymbol{\mathcal{C}}}, \tilde{\mathbf{A}}_{j<n}, \hat{\mathbf{A}}_n, \tilde{\mathbf{A}}_{j>n}), \mathbf{A}_n - \hat{\mathbf{A}}_n \rangle \tag{2.5}$$

$$+ \frac{L_n}{2} \|\mathbf{A}_n - \hat{\mathbf{A}}_n\|_F^2 + \lambda_n \|\mathbf{A}_n\|_1,$$

$$= \max \left(0, \hat{\mathbf{A}}_n - \frac{1}{L_n} \nabla_{\mathbf{A}_n} \ell(\tilde{\boldsymbol{\mathcal{C}}}, \tilde{\mathbf{A}}_{j<n}, \hat{\mathbf{A}}_n, \tilde{\mathbf{A}}_{j>n}) - \frac{\lambda_n}{L_n} \right), \tag{2.6}$$

where L_n is a Lipschitz constant of $\nabla_{\mathbf{A}_n} \ell(\tilde{\boldsymbol{\mathcal{C}}}, \tilde{\mathbf{A}}_{j<n}, \mathbf{A}_n, \tilde{\mathbf{A}}_{j>n})$ with respect to $\mathbf{A}_n$, and $\hat{\mathbf{A}}_n$ is an extrapolated point.
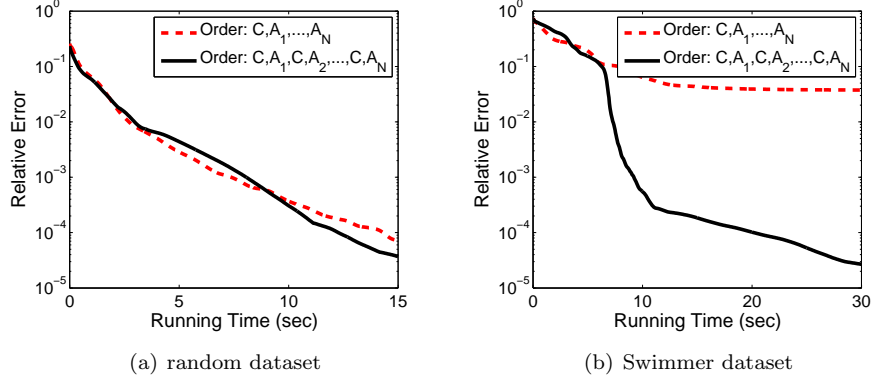
One can perform (2.4) and (2.6) to update $\boldsymbol{\mathcal{C}}$ and $\mathbf{A}$ in different manners. Directly applying the APG method proposed in [35] leads to the order of $\boldsymbol{\mathcal{C}}, \mathbf{A}_1, \cdots, \mathbf{A}_N$. However, since the core tensor $\boldsymbol{\mathcal{C}}$ interacts with all $\mathbf{A}_n$'s, updating it more frequently is expected to speed up the convergence of the algorithm. Hence, a more efficient way would be to update the variables in the order of $\boldsymbol{\mathcal{C}}, \mathbf{A}_1, \boldsymbol{\mathcal{C}}, \mathbf{A}_2, \cdots, \boldsymbol{\mathcal{C}}, \mathbf{A}_N$. Figure 2.1 shows the convergence behavior of APG with two different updating orders on a synthetic tensor and the Swimmer dataset [11]. From the figure, we see that APG with the updating order $\boldsymbol{\mathcal{C}}, \mathbf{A}_1, \cdots, \mathbf{A}_N$ performs comparably well as that with the order $\boldsymbol{\mathcal{C}}, \mathbf{A}_1, \boldsymbol{\mathcal{C}}, \mathbf{A}_2, \cdots, \boldsymbol{\mathcal{C}}, \mathbf{A}_N$ on the randomly generated data. However, the former behaves much worse than the latter on the Swimmer dataset. For this reason, we only consider the latter one, whose pseudocode is shown in Algorithm 1.

REMARK 2.2. *We do re-update in Line **ReDo** to make the objective nonincreasing. The monotonicity of the objective is important since the algorithm may perform unstably without the re-update. The computational cost of one objective evaluation is much cheaper than, actually not in the same order as, one gradient computation. Detailed complexity analysis is listed in Appendix B. Moreover, in each one of our experiments, the re-update occurs only a few times (often less than 10), so it needs only a little more computations.*

If some λ_n and/or λ_c vanish, we further do projections

$$\boldsymbol{\mathcal{C}}^{k,n} = \min \left(\max(1, \|\boldsymbol{\mathcal{M}}\|_\infty), \boldsymbol{\mathcal{C}}^{k,n} \right) \tag{2.9}$$

4

FIG. 2.1. *Results by APG with two different orders of updating the core tensor and factor matrices. (a). APG on a Gaussian random $20 \times 20 \times 20 \times 20$ tensor $\boldsymbol{\mathcal{M}}$ with core size $5 \times 5 \times 5 \times 5$; (b). APG on the $32 \times 32 \times 256$ Swimmer dataset [11] with core size $24 \times 20 \times 20$.*

(a) random dataset   (b) Swimmer dataset

Algorithm 1: Alternating proximal gradient for sparse NTD

Data: tensor $\boldsymbol{\mathcal{M}}$, core dimension $(R_1, \cdots, R_N)$, parameters $\lambda_c, \lambda_1, \cdots, \lambda_N \geq 0$, and $(\boldsymbol{\mathcal{C}}^{-1}, \mathbf{A}^{-1}) = (\boldsymbol{\mathcal{C}}^0, \mathbf{A}^0)$.

for $k = 1, 2, \cdots$ **do**

    Set $\boldsymbol{\mathcal{C}}^{k,-1} = \boldsymbol{\mathcal{C}}^{k,0} = \boldsymbol{\mathcal{C}}^0$ if $k = 1$ and $\boldsymbol{\mathcal{C}}^{k,-1} = \boldsymbol{\mathcal{C}}^{k-1,N-1}$, $\boldsymbol{\mathcal{C}}^{k,0} = \boldsymbol{\mathcal{C}}^{k-1,N}$ otherwise.

    for $n = 1, \cdots, N$ **do**

        Choose $L_c^{k,n}$ to be a Lipschitz constant of $\nabla_{\boldsymbol{\mathcal{C}}} \ell(\boldsymbol{\mathcal{C}}, \mathbf{A}_{j<n}^k, \mathbf{A}_{j\geq n}^{k-1})$ about $\boldsymbol{\mathcal{C}}$.

        Choose $\omega_c^{k,n} \geq 0$ and set $\hat{\boldsymbol{\mathcal{C}}}^{k,n} = \boldsymbol{\mathcal{C}}^{k,n-1} + \omega_c^{k,n}(\boldsymbol{\mathcal{C}}^{k,n-1} - \boldsymbol{\mathcal{C}}^{k,n-2})$.

        Update $\boldsymbol{\mathcal{C}}$ by

$$\boldsymbol{\mathcal{C}}^{k,n} = \max\left(0, \hat{\boldsymbol{\mathcal{C}}}^{k,n} - \frac{1}{L_c^{k,n}} \nabla_{\boldsymbol{\mathcal{C}}} \ell(\hat{\boldsymbol{\mathcal{C}}}^{k,n}, \mathbf{A}_{j<n}^k, \mathbf{A}_{j\geq n}^{k-1}) - \frac{\lambda_c}{L_c^{k,n}}\right); \tag{2.7}$$

        Choose L_n^k to be a Lipschitz constant of $\nabla_{\mathbf{A}_n} \ell(\boldsymbol{\mathcal{C}}^{k,n}, \mathbf{A}_{j<n}^k, \mathbf{A}_n, \mathbf{A}_{j<n}^{k-1})$ about $\mathbf{A}_n$.

        Choose $\omega_n^k \geq 0$ and set $\hat{\mathbf{A}}_n^k = \mathbf{A}_n^{k-1} + \omega_n^k(\mathbf{A}_n^{k-1} - \mathbf{A}_n^{k-2})$.

        Update $\mathbf{A}_n$ by

$$\mathbf{A}_n^k = \max\left(0, \hat{\mathbf{A}}_n^k - \frac{1}{L_n^k} \nabla_{\mathbf{A}_n} \ell(\boldsymbol{\mathcal{C}}^{k,n}, \mathbf{A}_{j<n}^k, \hat{\mathbf{A}}_n^k, \mathbf{A}_{j>n}^{k-1}) - \frac{\lambda_n}{L_n^k}\right). \tag{2.8}$$

        if $F(\boldsymbol{\mathcal{C}}^{k,n}, \mathbf{A}_{j\leq n}^k, \mathbf{A}_{j>n}^{k-1}) > F(\boldsymbol{\mathcal{C}}^{k,n-1}, \mathbf{A}_{j<n}^k, \mathbf{A}_{j\geq n}^{k-1})$ **then**

ReDo   Re-update $\boldsymbol{\mathcal{C}}^{k,n}$ and $\mathbf{A}_n^k$ by (2.7) and (2.8) with $\hat{\boldsymbol{\mathcal{C}}}^{k,n} = \boldsymbol{\mathcal{C}}^{k,n-1}$ and $\hat{\mathbf{A}}_n^k = \mathbf{A}_n^{k-1}$, respectively.

    Set $\boldsymbol{\mathcal{C}}^k = \boldsymbol{\mathcal{C}}^{k,N}$.

    if *Some stopping conditions are satisfied* **then**

        Output $(\boldsymbol{\mathcal{C}}^k, \mathbf{A}_1^k, \cdots, \mathbf{A}_N^k)$ and stop.

---

after (2.7) *and*

$$\mathbf{A}_n^k = \min\left(\max(1, \|\boldsymbol{\mathcal{M}}\|_\infty), \mathbf{A}_n^k\right) \tag{2.10}$$

after (2.8). *Omitting the superscript, it is easy to show that* (2.9) *and* (2.10) *respectively solve* (2.3) *and* (2.5) *with the extra constraints* (2.2) *and* (2.1).

2.3. Parameter settings. In our implementation of Algorithm 1, we set

$$L_c^{k,n} = \max\left(1, \left\|(\mathbf{A}_N^{k-1})^\top \mathbf{A}_N^{k-1} \otimes \cdots \otimes (\mathbf{A}_n^{k-1})^\top \mathbf{A}_n^{k-1} \otimes (\mathbf{A}_{n-1}^k)^\top \mathbf{A}_{n-1}^k \otimes \cdots \otimes (\mathbf{A}_1^k)^\top \mathbf{A}_1^k\right\|\right),$$

where $\|\cdot\|$ denotes matrix operator norm. Note that computing $L_c^{k,n}$ does not need to form the expensive Kronecker product because

$$\left\|\mathbf{A}_N^\top \mathbf{A}_N \otimes \cdots \otimes \mathbf{A}_1^\top \mathbf{A}_1\right\| = \prod_{i=1}^N \left\|\mathbf{A}_i^\top \mathbf{A}_i\right\|.$$

In the same way, we set

$$L_n^k = \max\left(1, \|\mathbf{B}_n^k (\mathbf{B}_n^k)^\top\|\right), \tag{2.11}$$

where

$$\mathbf{B}_n^k = \mathbf{C}_{(n)}^{k,n} \left(\mathbf{A}_N^{k-1} \otimes \cdots \otimes \mathbf{A}_{n+1}^{k-1} \otimes \mathbf{A}_{n-1}^k \otimes \cdots \otimes \mathbf{A}_1^k\right)^\top. \tag{2.12}$$

In addition, we take

$$\omega_c^{k,n} = \min\left(\hat{\omega}_c^{k,n}, 0.9999\sqrt{\frac{L_c^{k,n-1}}{L_c^{k,n}}}\right), \tag{2.13}$$

where $\hat{\omega}_c^{k,n}$ follows

$$\hat{\omega}_c^{k,n} = \frac{t^{k,n-1} - 1}{t^{k,n}}, \tag{2.14a}$$

$$t_c^{1,0} = 1, \ t_c^{k,0} = t_c^{k-1,N}, \ \text{for } k \geq 2, \tag{2.14b}$$

$$t_c^{k,n} = \frac{1}{2}\left(1 + \sqrt{1 + 4(t_c^{k,n-1})^2}\right), \ \text{for } k \geq 1, n = 1, \cdots, N. \tag{2.14c}$$

In the same way,

$$\omega_n^k = \min\left(\hat{\omega}^k, 0.9999\sqrt{\frac{L_n^{k-1}}{L_n^k}}\right), \tag{2.15}$$

where $\hat{\omega}^k$ follows

$$\hat{\omega}^k = \frac{t^{k-1} - 1}{t^k}, \tag{2.16a}$$

$$t^0 = 1, \ t^k = \frac{1}{2}\left(1 + \sqrt{1 + 4(t^{k-1})^2}\right), \ \text{for } k \geq 1. \tag{2.16b}$$

REMARK 2.3. *We perform "min" operation in (2.13) and (2.15) for convergence; see Theorem 2.2. The weights $\hat{\omega}_c^{k,n}$ in (2.14) and $\hat{\omega}^k$ in (2.16) are the same as that used in [3] for convex problems. Numerically, we observe that the extrapolation technique using the weights given in (2.13) and (2.15) can significantly speed up our algorithm. We also tested APG with the dynamically updated weight used in [23, 34] for non-convex matrix completion problem and observed that APG performs as well as that with the above extrapolation weights.*

2.4. Per-iteration complexity. Suppose $\mathcal{M} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and the core tensor $\mathcal{C} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$. Then the per-iteration cost of Algorithm 1 is roughly

$$N \cdot \mathcal{O}\left(\sum_{j=1}^N \left(\prod_{i=1}^j R_i\right)\left(\prod_{i=j}^N I_i\right) + \sum_{j=1}^N \left(\prod_{i=1}^j I_i\right)\left(\prod_{i=j}^N R_i\right)\right). \tag{2.17}$$

The detailed analysis is given in Appendix B.

REMARK 2.4. *If $N = \mathcal{O}(1)$ and $\max_n R_n \leq \mathcal{O}(\log \prod_{i=1}^N I_i)$, then the per-iteration cost of Algorithm 1 is scalable[2] about the data size $\prod_{i=1}^N I_i$.*

---

[2]Here, by scalability, we mean the cost is no greater than $s \cdot \log(s)$ if the data size is s.

2.5. Convergence results. It is shown in [35] that the APG method with cyclic block updating rule has global convergence to a stationary point. Since Algorithm 1 uses a different block updating order, its convergence cannot be directly obtained from [35]. However, we can still obtain the global convergence[3], which is summarized in Theorem 2.2. Although the proof idea for Theorem 2.2 is similar to that in [35], some places need careful modifications. Hence, for completeness, we include a modified proof in the Appendix.

THEOREM 2.2. *Let* $\left\{ \boldsymbol{\mathcal{W}}^k \triangleq (\boldsymbol{\mathcal{C}}^k, \mathbf{A}^k) \right\}$ *be the sequence generated by Algorithm 1. If* $\lambda_c, \lambda_1, \cdots, \lambda_N$ *are all positive, and*

1. *There exist positive constants* L_d, L_u *such that* $L_c^{k,n}, L_n^k \in [L_d, L_u]$;

2. *There is a positive constant* $\delta_\omega < 1$ *such that* $\omega_c^{k,n} \leq \delta_\omega \sqrt{\frac{L_c^{k,n-1}}{L_c^{k,n}}}$ *and* $\omega_n^k \leq \delta_\omega \sqrt{\frac{L_n^{k-1}}{L_n^k}}$ *for all* n *and* k, *where we use the notation* $L_c^{k,0} = L_c^{k-1,N}$;

then $\boldsymbol{\mathcal{W}}^k$ *converges to a stationary point* $\bar{\boldsymbol{\mathcal{W}}}$ *of* (1.5).

REMARK 2.5. *Positivity of sparse parameters implies the boundedness of* $\{\boldsymbol{\mathcal{W}}^k\}$, *and thus the existence of* L_d *and* L_u *can be guaranteed if* $L_c^{k,n}$ *and* L_n^k *are taken as in section 2.3.*

3. Sparse nonnegative Tucker decomposition with missing values. For some applications, $\boldsymbol{\mathcal{M}}$ may not be fully observed. This section modifies Algorithm 1 to handle this case. The problem is formulated as

$$\min_{\boldsymbol{\mathcal{C}}, \mathbf{A}} F_\Omega(\boldsymbol{\mathcal{C}}, \mathbf{A}) \equiv \tfrac{1}{2}\|\mathcal{P}_\Omega(\boldsymbol{\mathcal{C}} \times_1 \mathbf{A}_1 \cdots \times_N \mathbf{A}_N - \boldsymbol{\mathcal{M}})\|_F^2 + \lambda_c \|\boldsymbol{\mathcal{C}}\|_1 + \sum_{n=1}^N \lambda_n \|\mathbf{A}_n\|_1,$$
$$\text{s.t. } \boldsymbol{\mathcal{C}} \in \mathbb{R}_+^{R_1 \times \cdots \times R_N}, \mathbf{A}_n \in \mathbb{R}_+^{I_n \times R_n}, \ n = 1, \cdots, N, \tag{3.1}$$

where Ω indexes the observed entries of $\boldsymbol{\mathcal{M}}$, and $\mathcal{P}_\Omega(\boldsymbol{\mathcal{A}})$ keeps the entries of $\boldsymbol{\mathcal{A}}$ in Ω and zeros out all others. As did in [34, 36], we introduce variable $\boldsymbol{\mathcal{X}}$, restrict $\mathcal{P}_\Omega(\boldsymbol{\mathcal{X}}) = \mathcal{P}_\Omega(\boldsymbol{\mathcal{M}})$, and write (3.1) equivalently to

$$\min_{\boldsymbol{\mathcal{C}}, \mathbf{A}, \boldsymbol{\mathcal{X}}} \quad \tfrac{1}{2}\|\boldsymbol{\mathcal{C}} \times_1 \mathbf{A}_1 \cdots \times_N \mathbf{A}_N - \boldsymbol{\mathcal{X}}\|_F^2 + \lambda_c \|\boldsymbol{\mathcal{C}}\|_1 + \sum_{n=1}^N \lambda_n \|\mathbf{A}_n\|_1,$$
$$\text{s.t.} \quad \boldsymbol{\mathcal{C}} \in \mathbb{R}_+^{R_1 \times \cdots \times R_N}, \mathbf{A}_n \in \mathbb{R}_+^{I_n \times R_n}, \ n = 1, \cdots, N, \ \mathcal{P}_\Omega(\boldsymbol{\mathcal{X}}) = \mathcal{P}_\Omega(\boldsymbol{\mathcal{M}}). \tag{3.2}$$

To modify Algorithm 1 for (3.1) or equivalently (3.2), we set $\boldsymbol{\mathcal{X}}^0 = \mathcal{P}_\Omega(\boldsymbol{\mathcal{M}})$ in the beginning. At the k-th iteration of Algorithm 1, we use $\boldsymbol{\mathcal{M}} = \boldsymbol{\mathcal{X}}^{k-1}$, wherever $\boldsymbol{\mathcal{M}}$ is referred to. After Line **ReDo** of Algorithm 1, update $\boldsymbol{\mathcal{X}}$ by

$$\boldsymbol{\mathcal{X}}^k = \mathcal{P}_\Omega(\boldsymbol{\mathcal{M}}) + \mathcal{P}_{\Omega^c}(\boldsymbol{\mathcal{C}}^k \times_1 \mathbf{A}_1^k \cdots \times_N \mathbf{A}_N^k). \tag{3.3}$$

Compared to Algorithm 1, the modified method needs extra computation for the update (3.3), which costs about $2 \sum_{j=1}^N \left(\prod_{i=1}^j I_i \right) \left(\prod_{i=j}^N R_i \right)$. Therefore, the per-iteration complexity of the modified algorithm is still scalable about the data size if $N = \mathcal{O}(1)$ and $\max_n R_n \leq \mathcal{O}(\log \prod_{i=1}^N I_i)$. In addition, following the proof of Theorem 2.2, one can show that the same convergence result holds for the modified algorithm.

4. Extensions. For some applications, the core tensor $\boldsymbol{\mathcal{C}}$ may not be required nonnegative [10]. Algorithm 1 can be modified to handle this case by changing (2.7) to

$$\boldsymbol{\mathcal{C}}^{k,n} = \mathcal{S}_{\frac{\lambda_c}{L_c^{k,n}}} \left(\hat{\boldsymbol{\mathcal{C}}}^{k,n} - \frac{1}{L_c^{k,n}} \nabla_{\boldsymbol{\mathcal{C}}} \ell(\hat{\boldsymbol{\mathcal{C}}}^{k,n}, \mathbf{A}_{j<n}^k, \mathbf{A}_{j \geq n}^{k-1}) \right), \tag{4.1}$$

where $\mathcal{S}_\mu(\boldsymbol{\mathcal{X}})$ is a soft-thresholding operator defined component-wisely as

$$\mathcal{S}_\mu(x) = \text{sign}(x) \cdot \max(0, |x| - \mu).$$

---

[3]Since the problem is non-convex, we only get convergence to a stationary point, and different starting points can produce different limit points.

The APG method can also be adapted to solve sparse higher-order principal component analysis (HOPCA), which imposes orthogonality constraint on each factor matrix. The problem is formulated as

$$\min_{\boldsymbol{\mathcal{C}},\mathbf{A}} \frac{1}{2}\|\boldsymbol{\mathcal{C}}\times_1\mathbf{A}_1\cdots\times_N\mathbf{A}_N - \boldsymbol{\mathcal{M}}\|_F^2 + \lambda_c\|\boldsymbol{\mathcal{C}}\|_1 + \sum_{n=1}^{N}\lambda_n\|\mathbf{A}_n\|_1,$$
$$\text{s.t. } \mathbf{A}_n^\top\mathbf{A}_n = \mathbf{I}_n, \ n = 1,\cdots,N, \tag{4.2}$$

where $\mathbf{I}_n$ is an identity matrix of appropriate size. When $\lambda_c = 0$, the optimal $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{M}}\times_1\mathbf{A}_1^\top\cdots\times_N\mathbf{A}_N$, and one can eliminate $\boldsymbol{\mathcal{C}}$ as shown in [19]. The concurrency of sparsity and orthogonality constraints makes the problem much more difficult. The work [1] considers rank-1 factor matrix with only one column and relaxes the orthogonality constraint to $\mathbf{A}_n^\top\mathbf{A}_n \leq 1$. Then it applies block coordinate minimization method to solve the relaxed problem. When some $\mathbf{A}_n$ has more than one columns, we relax (4.2) to

$$\min_{\boldsymbol{\mathcal{C}},\mathbf{A}} \frac{1}{2}\|\boldsymbol{\mathcal{C}}\times_1\mathbf{A}_1\cdots\times_N\mathbf{A}_N - \boldsymbol{\mathcal{M}}\|_F^2 + \lambda_c\|\boldsymbol{\mathcal{C}}\|_1 + \sum_{n=1}^{N}\lambda_n\|\mathbf{A}_n\|_1 + \frac{\mu}{2}\sum_{n=1}^{N}\sum_{i\neq j}\left(\mathbf{a}_{n,i}^\top\mathbf{a}_{n,j}\right)^2$$
$$\text{s.t. } \|\mathbf{a}_{n,j}\|_2 \leq 1, \ n = 1,\cdots,N, \forall j, \tag{4.3}$$

where $\mathbf{a}_{n,j}$ denotes the j-th column of $\mathbf{A}_n$, $\sum_{i\neq j}\left(\mathbf{a}_{n,i}^\top\mathbf{a}_{n,j}\right)^2$ is used to promote the orthogonality of $\mathbf{A}_n$, and μ is a penalty parameter. We want to mention that our orthogonality regularization term is similar to that used in [30] for promoting the discrepancy of dictionaries and also that used on pp. 222 of [7].

Our method for (4.3) is similar to Algorithm 1 and cycles over the variables by $\boldsymbol{\mathcal{C}},\mathbf{A}_1,\boldsymbol{\mathcal{C}},\mathbf{A}_2,\cdots,\boldsymbol{\mathcal{C}},\mathbf{A}_N$. The update of $\boldsymbol{\mathcal{C}}$ is done by (4.1), and $\mathbf{A}_n$ is updated one column by one column. Specifically, assume the current iterate is $(\boldsymbol{\mathcal{C}}^{k,n},\mathbf{A}_{i<n}^k,\mathbf{A}_{i\geq n}^{k-1})$. Let $\mathbf{B}_n^k$ be the one obtained from (2.12). Using (A.3), we update the columns of $\mathbf{A}_n$ from $j = 1$ to R_n by

$$\mathbf{a}_{n,j}^k = \operatorname*{argmin}_{\|\mathbf{a}_{n,j}\|_2\leq 1} \frac{1}{2}\left\|\mathbf{a}_{n,j}\mathbf{b}_n^{k,j} + (\tilde{\mathbf{A}}_n^k)_{j^c}(\mathbf{B}_n^k)^{j^c} - \mathbf{M}_{(n)}\right\|_F^2 + \lambda_n\|\mathbf{a}_{n,j}\|_1$$
$$+ \mu\left(\left\langle(\tilde{\mathbf{A}}_n^k)_{j^c}(\tilde{\mathbf{A}}_n^k)_{j^c}^\top\hat{\mathbf{a}}_{n,j}^k,\mathbf{a}_{n,j} - \hat{\mathbf{a}}_{n,j}^k\right\rangle + \frac{L_{n,j}^k}{2}\|\mathbf{a}_{n,j} - \hat{\mathbf{a}}_{n,j}^k\|_2^2\right), \tag{4.4}$$

where $\mathbf{b}_n^{k,j}$ denotes the j-th row of $\mathbf{B}_n^k$, $(\mathbf{B}_n^k)^{j^c}$ is the submatrix by taking all rows of $\mathbf{B}_n^k$ except the j-th one,

$$\hat{\mathbf{a}}_{n,j}^k = \mathbf{a}_{n,j}^{k-1} + \omega_{n,j}^k(\mathbf{a}_{n,j}^{k-1} - \mathbf{a}_{n,j}^{k-2})$$

is an extrapolated point, $(\tilde{\mathbf{A}}_n^k)_{j^c}$ is short for $\left(\mathbf{a}_{n,1}^k,\cdots,\mathbf{a}_{n,j-1}^k,\mathbf{a}_{n,j+1}^{k-1},\cdots,\mathbf{a}_{n,R_n}^{k-1}\right)$, and $L_{n,j}^k$ is a Lipschitz constant of the gradient of

$$\frac{1}{2}\left(\sum_{i<j}\left(\mathbf{a}_{n,j}^\top\mathbf{a}_{n,i}^k\right)^2 + \sum_{i>j}\left(\mathbf{a}_{n,j}^\top\mathbf{a}_{n,i}^{k-1}\right)^2\right)$$

with respect to $\mathbf{a}_{n,j}$. One can easily write the update in (4.4) explicitly as

$$\mathbf{a}_{n,j}^k = \mathcal{P}_{B_1}\left[\mathcal{S}_{\frac{\lambda_n}{b+\mu L}}\left(\frac{\mu L}{b+\mu L}\hat{\mathbf{a}}_{n,j}^k - \frac{\left((\tilde{\mathbf{A}}_n^k)_{j^c}(\mathbf{B}_n^k)^{j^c} - \mathbf{M}_{(n)}\right)(\mathbf{b}_n^{k,j})^\top}{b+\mu L} - \frac{\mu}{b+\mu L}(\tilde{\mathbf{A}}_n^k)_{j^c}(\tilde{\mathbf{A}}_n^k)_{j^c}^\top\hat{\mathbf{a}}_{n,j}^k\right)\right], \tag{4.5}$$

where $b = \|\mathbf{b}_n^{k,j}\|_2^2$, $L = L_{n,j}^k$, and $\mathcal{P}_{B_1}$ denotes the projection to unit Euclidean ball.

Following the proof of Theorem 2.2, one can show that the method described above has global convergence if the parameters $L_{n,j}^k, \omega_{n,j}^k, L_c^k, \omega_c^{k,n}$ satisfy conditions as those in Theorem 2.2. We do not repeat it here.

5. Numerical experiments. In this section, we compare Algorithm 1 (APG), HONMF in [26], and HALS in [28] for solving (sparse) NTD on both synthetic and real world data. Also, we test the modified version of Algorithm 1 and HONMF for solving (sparse) NTD with missing values. The code of all compared solvers is accessible online. There are of course more other solvers for (sparse) NTD such as dGN in [29], ALS in [12], and CCD in [24]. However, we do not get the code of dGN, and the code of CCD and ALS only handles the case where the core tensor is fixed to identity tensor.

All the tests are performed on a laptop with an i7-620m CPU and 3GB RAM and running 32-bit Windows 7 and MATLAB 2010b with Statistics Toolbox and Tensor Toolbox of version 2.5 [2].

5.1. Implementation details. This subsection specifies the implementation of Algorithm 1 in details about initialization and stopping criteria. Unless specified, all parameters for HONMF and HALS are set to their default values.

Initialization. For all the compared algorithms, we use the same starting point. Throughout the tests, we first randomly generate $\mathbf{A}_1^0, \cdots, \mathbf{A}_N^0$ and then process them by the Higher-order Orthogonal Iteration algorithm in [9]. Specifically, for (1.5), let

$$\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{M}} \times_1 \left(\mathbf{A}_1^0\right)^\top \cdots \times_{n-1} \left(\mathbf{A}_{n-1}^0\right)^\top \times_{n+1} \left(\mathbf{A}_{n+1}^0\right)^\top \times_N \left(\mathbf{A}_N^0\right)^\top, \tag{5.1}$$

and update $\mathbf{A}_n^0 = \max(\epsilon_{machine}, \mathbf{U}_n)$ alternatively for $n = 1, \cdots, N$, where $\epsilon_{machine}$ stands for machine precision and $\mathbf{U}_n$ contains the left R_n singular vectors of $\mathbf{B}_{(n)}$. Then set

$$\boldsymbol{\mathcal{C}}^0 = \boldsymbol{\mathcal{M}} \times_1 \left(\mathbf{A}_1^0\right)^\top \cdots \times_N \left(\mathbf{A}_N^0\right)^\top. \tag{5.2}$$

For (3.1), we use the same initialization except replacing $\boldsymbol{\mathcal{M}}$ to $\mathcal{P}_\Omega(\boldsymbol{\mathcal{M}})$ in (5.1) and (5.2). It is observed that all the algorithms perform better with this kind of starting point than a random one, in both convergence speed and chance of avoiding local minima. The use of strictly positive initial points is mainly due to the consideration that HONMF does not allow its iterates to have zero components.

Stopping criteria. We stop Algorithm 1 and its modified version in section 3 if a maximum number of iterations or maximum time is reached or one of the following conditions is satisfied

$$\frac{\|\mathcal{P}_\Omega\left(\boldsymbol{\mathcal{C}}^k \times_1 \mathbf{A}_1^k \cdots \times_N \mathbf{A}_N^k - \boldsymbol{\mathcal{M}}\right)\|_F}{\|\mathcal{P}_\Omega(\boldsymbol{\mathcal{M}})\|_F} \leq tol, \quad \text{for some } k, \tag{5.3a}$$

$$\frac{|F_\Omega^k - F_\Omega^{k+1}|}{1 + F_\Omega^k} \leq tol, \quad \text{for three consecutive } k\text{'s}, \tag{5.3b}$$

where $F_\Omega^k \triangleq F_\Omega(\boldsymbol{\mathcal{C}}^k, \mathbf{A}_1^k, \cdots, \mathbf{A}_N^k)$ and tol is a small positive value specified below. Note that for Algorithm 1, Ω contains all indices.

5.2. Nonnegative Tucker decomposition. In this subsection, we compare APG, HONMF, and HALS on solving NTD, i.e., (1.5) with all of $\lambda_c, \lambda_1, \cdots, \lambda_N$ set to *zero*. We first test them on two sets of synthetic data and then on two image datasets.

Synthetic data. In the first synthetic dataset, each tensor has the form $\boldsymbol{\mathcal{M}} = \boldsymbol{\mathcal{C}} \times_1 \mathbf{A}_1 \times_2 \mathbf{A}_2 \times_3 \mathbf{A}_3$, where $\boldsymbol{\mathcal{C}}$ is generated by MATLAB's command `rand(5,5,5)` and each $\mathbf{A}_i$ by command `max(0,randn(80,5))`. Then $\boldsymbol{\mathcal{M}}$ is re-scaled to have unit maximum component. Each tensor $\boldsymbol{\mathcal{M}}$ in the second test is generated in the same way but has an *unbalanced* dimension $10 \times 10 \times 1000$, and the core tensor is $3 \times 3 \times 30$. We emphasize that uniformly random $\boldsymbol{\mathcal{C}}$ makes the problem more difficult[4] than Gaussian random one because

---

[4]For the case that $\boldsymbol{\mathcal{C}}$ is also Gaussian randomly generated, the performance of APG and HALS is similar.

TABLE 5.1

Average results over 20 independent runs by APG, HONMF and HALS on two synthetic datasets

	APG		HONMF		HALS	
noise level	rel. err.	# iter	rel. err.	# iter	rel. err.	# iter
$(I_1, I_2, I_3) = (80, 80, 80), (R_1, R_2, R_3) = (5, 5, 5)$						
$\eta = 0.00$	7.09e-004	467	6.06e-002	87	2.45e-003	758
$\eta = 0.05$	2.79e-003	468	6.86e-002	48	3.27e-003	732
$\eta = 0.10$	4.78e-003	466	7.15e-002	47	5.44e-003	759
$(I_1, I_2, I_3) = (10, 10, 1000), (R_1, R_2, R_3) = (3, 3, 30)$						
$\eta = 0.00$	5.12e-004	653	2.52e-002	287	2.97e-003	737
$\eta = 0.05$	1.49e-002	668	3.00e-002	232	1.50e-002	739
$\eta = 0.10$	3.02e-002	670	3.84e-002	222	3.01e-002	740

the former is not *zero*-mean. The true dimension is used in our tests, namely, $I_n = 50, R_n = 5, \forall n$ is set in (1.5) for the first dataset and $(I_1, I_2, I_3) = (10, 10, 1000), (R_1, R_2, R_3) = (3, 3, 30)$ for the second one.

We add normalized noise to each tensor, namely, we input to each algorithm with $\mathcal{M}^{nois} = \mathcal{M} + \eta \frac{\|\mathcal{M}\|_F}{\|\mathcal{N}\|_F} \mathcal{N}$, where the entries of $\mathcal{N}$ follow i.i.d standard Gaussian distribution. We run each algorithm to $t_{\max}$ (sec) and compare their relative error $\frac{\|\mathcal{C}^r \times_1 \mathbf{A}_1^r \times_2 \mathbf{A}_2^r \times_3 \mathbf{A}_3^r - \mathcal{M}\|_F}{\|\mathcal{M}\|_F}$, where $(\mathcal{C}^r, \mathbf{A}_1^r, \mathbf{A}_2^r, \mathbf{A}_3^r)$ is a solution obtained by running an algorithm. Table 5.1 shows the average relative error and number of iterations for the three algorithms over 20 independent runs with $t_{\max} = 10$ and different η's. Figure 5.1 plots how the relative error changes with respect to the running time for each algorithm with $t_{\max} = 20$ and also to iterations.

From the table, we see that APG performs significantly better than HONMF and HALS for noiseless case. When there is noise, i.e., $\eta > 0$, APG is still much better than HONMF and comparable to HALS. From the figure, we see that HONMF converges very slowly[5] in both cases and HALS works well for $\mathcal{M}$ with balanced dimension but converges slowly for the unbalanced one. APG converges faster than both HONMF and HALS, in particular for the unbalanced case.

To see how the algorithms perform on decomposing nonnegative tensors with larger ranks, we also test them on random tensors generated in the same way as above with size $80 \times 80 \times 80$ and each mode rank r, where r varies from 3 to 30 with increment 3. Each algorithm runs to 1,000 iterations. Figure 5.2 plots the average relative errors of 10 independent runs for each algorithm. From the figure, we see that APG performs consistently better than HONMF and HALS and much better when r is small.

Image data. The first test uses the Swimmer dataset constructed in [11], which has 256 swimmer images and each one has resolution of 32×32. We form a $32 \times 32 \times 256$ tensor $\mathcal{M}$ using the dataset and then re-scale it to have unit maximum component. The core dimension is set to $(24, 20, 20)$[6]. We run APG, HONMF, and HALS to $t_{\max} = 30$ (sec) and plot their relative errors on the left of Figure 5.3. The second test uses a brain MRI image of size $181 \times 217 \times 181$, which has been tested in [24] for sparse nonnegative tensor decomposition. We re-scale it to have unit maximum pixel and set the core size to $(30, 30, 30)$. All the three algorithms run to $t_{\max} = 600$ (sec), and the relative errors are plotted on the right of Figure 5.3. From the figure, we see that HONMF performs the worst and HALS decreases the objective faster than APG in the beginning but APG eventually converges faster. In particular for the test with Swimmer dataset, the overall convergence speed of APG is much faster than that of HALS, and APG reaches much lower relative

---

[5]The code of HONMF is implemented for NTD with missing value. Its running time would be reduced if it were implemented separately for the NTD. However, we observe that HONMF converges much slower than our algorithm.

[6]The mode-n ranks of $\mathcal{M}$ are 24, 14, and 13 for $n = 1, 2, 3$, respectively. Larger size is used to improve the data fitting.

FIG. 5.1. *Convergence behavior of APG, HONMF and HALS on synthetic data. Left:* $80 \times 80 \times 80$ *nonnegative tensor* $\boldsymbol{\mathcal{M}}$ *and* $5 \times 5 \times 5$ *core tensor* $\boldsymbol{\mathcal{C}}$*; Right:* $10 \times 10 \times 1000$ *nonnegative tensor* $\boldsymbol{\mathcal{M}}$ *and* $3 \times 3 \times 30$ *core tensor* $\boldsymbol{\mathcal{C}}$*.*

FIG. 5.2. *Average relative errors of 10 independent runs for APG, HONMF, and HALS on synthetic tensors of size* $80 \times 80 \times 80$ *and with each mode rank* r*.*

errors while HALS seems to be trapped at some local solution[7].

5.3. Sparse nonnegative Tucker decomposition. In this subsection, we compare APG and HONMF for solving sparse NTD, i.e., (1.5) with at least one of $\lambda_c, \lambda_1, \cdots, \lambda_N$ set to be positive. HALS is not coded[8] for sparse NTD. Hence, we do not include HALS for comparison.

---

[7]Sometimes, APG is also trapped at some local solution. We run the three algorithms on the Swimmer dataset to maximum 30 seconds. If the relative error is below 10^{-3}, we regard the algorithm reaches a global solution. Among 20 independent runs, APG, HONMF, and HALS reach a global solution 11, 0, and 5 times, respectively. We also test the three algorithms with smaller rank (24,18,17), in which case APG, HONMF, and HALS reach a global solution 16, 0, and 4 times respectively among 20 independent runs.

[8]In the implementation of HALS, all factor matrices are re-scaled such that each column has unit length after each iteration. The re-scaling is necessary for efficient update of the core tensor and does not change the objective value of (1.5) if all sparsity parameters are *zero*. However, it will change the objective if some of $\lambda_c, \lambda_1, \cdots, \lambda_N$ are positive.

FIG. 5.3. *Convergence behavior of APG, HONMF, and HALS on Swimmer dataset (left) and a brain MRI image (right).*
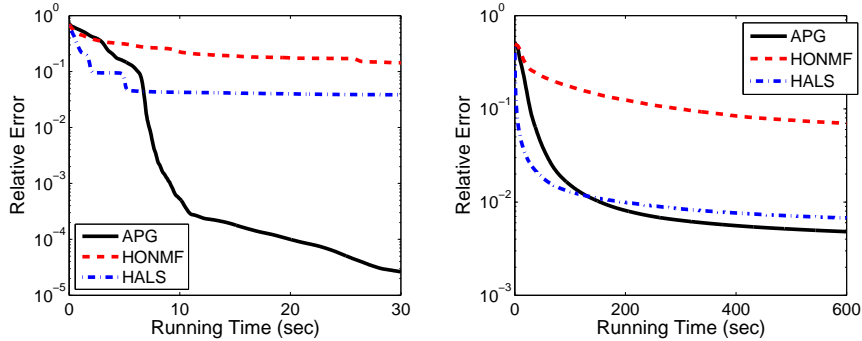

TABLE 5.2
Average results by APG and HONMF on a brain MRI image with the core size $R_1 = R_2 = R_3 = 30$

time	APG					HONMF				
	obj.	rel. err.	fac. den.	core den.	# iter	obj.	rel. err.	fac. den.	core den.	# iter
100	1.6622e+3	6.15e-2	32.45%	7.62%	185	6.2934e+3	1.77e-1	32.47%	31.84%	31
200	8.4659e+2	2.94e-2	23.22%	14.45%	370	4.7762e+3	1.48e-1	30.61%	31.01%	48
300	6.8898e+2	2.25e-2	20.49%	16.87%	555	4.0240e+3	1.30e-1	29.14%	29.91%	63

We compare APG and HONMF on the brain MRI image used above and the CBCL face image dataset[9] which has been tested in [32] for nonnegative tensor decomposition. For the brain MRI image, we set $R_1 = R_2 = R_3 = 30$ and $\lambda_c = \lambda_1 = \lambda_2 = \lambda_3 = 0.5$ in (1.5). We run APG and HONMF to $t_{\max} = 300$ (sec) and report the results at time $t = 100, 200, 300$ (sec). Table 5.2 summarizes the average results of 10 independent runs. The "core den." is calculated by $\frac{\# \text{ nonzeros of } \mathcal{C}^r}{30^3}$ and "fac. den." by $\frac{\sum_{n=1}^{3} \# \text{ nonzeros of } \mathbf{A}_n^r}{30 \cdot (181 + 217 + 181)}$. We see that APG reaches much lower objective values and relative errors than those by HONMF. In addition, the solutions obtained by APG are sparser than those by HONMF and are potentially easier to interpret.

The CBCL dataset has 6977 face images, and each one is 19×19. We use all these images to form a $19 \times 19 \times 6977$ nonnegative tensor $\mathcal{M}$, which is then re-scaled to have unit maximum component. The core size is set to $(R_1, R_2, R_3) = (5, 5, 50)$ and the sparsity parameters to $\lambda_c = 0.5, \lambda_1 = \lambda_2 = \lambda_3 = 0$, namely, we only want the core tensor to be sparse. Table 5.3 reports the average results obtained by APG and HONMF at running time $t = 25, 50, 75, 100$ (sec). We see that APG reaches much lower objective values and also lower relative errors than those by HONMF. The solutions given by APG are much sparser than those by HONMF. This may be because APG uses the constraints (2.1) while HONMF simply normalizes each factor matrix after every iteration. However, it somehow validates the use of the constraints (2.1).

5.4. Sparse nonnegative Tucker decomposition with missing values. In this subsection, we test APG for solving (3.1) on synthetic data and compare it to HONMF on the brain MRI image used above.

Performance of APG with different sample ratios. First, we show that APG using partial observations can achieve similar accuracies as that using full observations. Each tensor has the form $\mathcal{M} = \mathcal{C} \times_1 \mathbf{A}_1 \times_2 \mathbf{A}_2 \times_3 \mathbf{A}_3$ and is re-scaled to have unit maximum component, where $\mathcal{C}$ is generated by MATLAB's command `max(0,randn(R,R,R))` and each factor matrix $\mathbf{A}_n$ by `max(0,randn(50,R))` with R varying among $\{5, 8, 11, 14, 17, 20, 23, 26\}$. We choose SR $= 10\%, 30\%, 50\%, 100\%$ samples uniformly at random and compare the performance of APG using different SRs. The maximum number of iterations
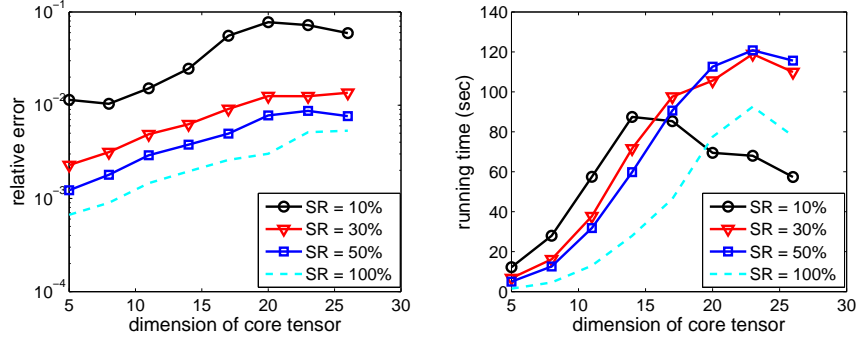
---

[9]http://www.ai.mit.edu/projects/cbcl

Table 5.3

Average results by APG and HONMF on CBCL dataset with the core size $(R_1, R_2, R_3) = (5, 5, 50)$

	APG				HONMF			
time	obj.	rel. err.	core den.	# iter	obj.	rel. err.	core den.	# iter
25	3.2469e+4	2.72e-1	11.45%	135	5.9824e+4	3.63e-1	90.19%	29
50	3.1453e+4	2.68e-1	7.56%	271	5.3017e+4	3.40e-1	69.26%	57
75	3.1370e+4	2.68e-1	6.78%	408	4.9786e+4	3.28e-1	58.03%	84
100	3.1344e+4	2.67e-1	6.46%	545	4.7289e+4	3.20e-1	51.26%	112

FIG. 5.4. *Average relative errors (left) and runnting time (right) by APG using different sample ratios*

is set to 5,000 and the stopping tolerance to $tol = 10^{-5}$. Figure 5.4 plots the average relative errors and running time (sec) of APG over 20 independent trials. We see that APG using 30% and 50% samples gives similar accuracies as that using full observations. APG with 10% samples can still make relative errors low to about 1% as $R \leq 14$, but 10% samples seem not enough when $R \geq 17$. Longer time by APG with partial observations is due to the extra update (3.3) and more iterations. When $R \geq 17$, the running time of APG with 10% samples decreases because it stops earlier.

Comparison with HONMF[10]. Secondly, we compare APG to HONMF on the brain MRI image used above. The core dimension is set to $R_1 = R_2 = R_3 = 30$ and sparsity parameters to $\lambda_c = \lambda_1 = \lambda_2 = \lambda_3 = 0.5$. We compare the two algorithms using SR $= 10\%, 30\%, 50\%$ uniformly randomly chosen samples and run them to $t_{\max} = 600$ (sec). Table 5.4 shows the average results at time $t = 150, 300, 450, 600$ for different SRs over 5 independent trials. From the table, we see that HONMF fails with 10% samples while APG can still work reasonably. In all cases, APG performs better than HONMF in both accuracy and speed. The solutions given by APG are sparser than those by HONMF for SR $= 30\%, 50\%$.

5.5. Sparse higher-order principal component analysis. We use a simple test with synthetic data to show that (4.3) can be better than unregularized HOPCA that sets all of $\lambda_c, \lambda_1, \cdots, \lambda_N$ to *zero* in (4.2). We use the APG method described in Section 4 for (4.3) and HOOI [9] for the unregularized HOPCA. We set $L_{n,j}^k = \|(\tilde{\mathbf{A}}_n^k)_{j^c}(\tilde{\mathbf{A}}_n^k)_{j^c}^\top\|$ in (4.5) and $\omega_{n,j}^k$ in the same way as in (2.15).
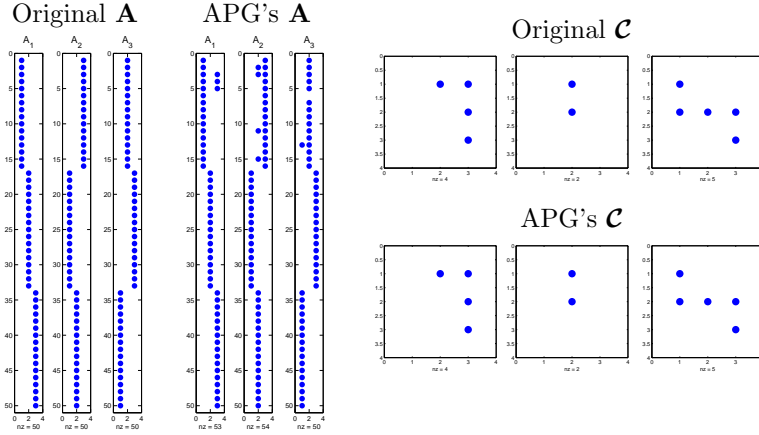
We generate a $50 \times 50 \times 50$ tensor in the form of $\boldsymbol{\mathcal{M}} = \boldsymbol{\mathcal{C}} \times_1 \mathbf{A}_1 \times_2 \mathbf{A}_2 \times \mathbf{A}_3 + \boldsymbol{\mathcal{N}}$. Here, $\boldsymbol{\mathcal{C}}$ is $3 \times 3 \times 3$, and each element is drawn from standard Gaussian distribution. Then 60% components of $\boldsymbol{\mathcal{C}}$ are selected uniformly at random and set to *zero*. Factor matrices have sparsity patterns shown in Figure 5.5, and each non-zero element is drawn from standard Gaussian distribution. Then each column is normalized. $\boldsymbol{\mathcal{N}}$ is

---

[10]Although HONMF converges very slowly, it is the only one we can find that is also coded for sparse nonnegative Tucker decomposition with missing values.

Table 5.4

Average results by APG and HONMF on a brain MRI image from different samples

	APG					HONMF				
time	obj.	rel. err.	fac. den.	core den.	# iter	obj.	rel. err.	fac. den.	core den.	# iter
					SR = 10%					
150	1.3418e+3	2.32e-1	30.49%	0.72%	208	1.5608e+4	1.00e+0	0.00%	0.00%	67
300	9.3130e+2	1.80e-1	17.03%	0.88%	416	1.5608e+4	1.00e+0	0.00%	0.00%	150
450	7.9761e+2	1.60e-1	14.28%	1.23%	623	1.5608e+4	1.00e+0	0.00%	0.00%	232
600	7.4748e+2	1.53e-1	13.60%	1.40%	831	1.5608e+4	1.00e+0	0.00%	0.00%	315
					SR = 30%					
150	1.5808e+3	1.27e-1	35.15%	2.31%	191	2.4525e+3	1.88e-1	31.80%	41.87%	40
300	9.0284e+2	8.17e-2	19.65%	4.38%	384	1.9277e+3	1.58e-1	28.90%	38.48%	67
450	7.0151e+2	6.25e-2	17.29%	6.34%	576	1.6362e+3	1.38e-1	26.44%	33.07%	96
600	6.2076e+2	5.34e-2	15.69%	7.60%	769	1.4587e+3	1.25e-1	24.54%	30.12%	129
					SR = 50%					
150	1.8767e+3	1.08e-1	35.03%	3.64%	184	3.7494e+3	1.91e-1	31.75%	36.66%	40
300	9.5363e+2	5.88e-2	22.42%	7.55%	367	2.8367e+3	1.59e-1	28.70%	40.15%	64
450	7.0877e+2	4.03e-2	19.29%	10.74%	550	2.3369e+3	1.37e-1	26.48%	42.33%	92
600	6.2737e+2	3.32e-2	17.75%	12.41%	733	2.0265e+3	1.23e-1	24.96%	42.58%	124

FIG. 5.5. *Sparsity pattern of the orginal $\mathcal{C}$ and $\mathbf{A}$ and those given by APG method*

Gaussian random noise and makes the signal-to-noise-ratio SNR = 60. The sparsity parameters are set to $\lambda_c = \lambda_1 = \lambda_2 = \lambda_3 = 0.02$, and orthogonality parameter is tuned to $\mu = 0.1$ in (4.3). The sparsity patterns of the original $\mathcal{C}$ and $\mathbf{A}$ and those[11] given by APG are plotted in Figure 5.5. We see that the solution given by APG have almost the same sparsity pattern as the original ones. To see how close to orthogonality each factor matrix is given by APG, we first normalize each column of the factor matrices and then calculate $\|\mathbf{A}_n^\top \mathbf{A}_n - \mathbf{I}\|_F / \|\mathbf{I}\|_F$, which are $2.95 \times 10^{-3}, 1.36 \times 10^{-3}, 7.24 \times 10^{-5}$, respectively for $n = 1, 2, 3$. Hence, they are almost orthogonal. Although the solution by HOOI makes a relatively higher data fitting, it is highly dense with no zero element. Therefore, the relaxed model (4.3) can potentially give better solution than (4.2) for some applications such as classification.

---

[11] We permute the columns of the factor matrices and do permutations to the core tensor accordingly.

6. Conclusions. Sparse NTD aims at decomposing a tensor into the product of a core tensor and some factor matrices with nonnegativity and sparsity constraints. Existing algorithms for this problem either converge rapidly with very expensive per-iteration cost or have low per-iteration cost with very slow convergence speed. We have proposed the APG method, which owns both low per-iteration complexity and fast convergence speed. Moreover, the algorithm has been modified for sparse NTD from partial observations of a target tensor. The modified algorithm also has low per-iteration cost and can give similar decompositions from half of or even fewer observations as those from full observations.

Appendix A. Efficient computation. The most expensive step in Algorithm 1 is the computation of $\nabla_{\mathcal{C}}\ell(\mathcal{C}, \mathbf{A})$ and $\nabla_{\mathbf{A}_n}\ell(\mathcal{C}, \mathbf{A})$ in (2.7) and (2.8), respectively. Note that we have omitted the superscript. Next, we discuss how to efficiently compute them.

Computation of $\nabla_{\mathcal{C}}\ell$. According to (1.2), we have

$$\ell(\mathcal{C}, \mathbf{A}) = \frac{1}{2}\left\|\left(\otimes_{n=N}^1 \mathbf{A}_n\right)\mathrm{vec}(\mathcal{C}) - \mathrm{vec}(\mathcal{M})\right\|_2^2.$$

Using the properties of Kronecker product (see [14], for example), we have

$$\mathrm{vec}\left(\nabla_{\mathcal{C}}\ell(\mathcal{C}, \mathbf{A})\right) = \left(\otimes_{n=N}^1 \mathbf{A}_n^\top \mathbf{A}_n\right)\mathrm{vec}(\mathcal{C}) - \left(\otimes_{n=N}^1 \mathbf{A}_n^\top\right)\mathrm{vec}(\mathcal{M}). \tag{A.1}$$

It is extremely expensive to explicitly reformulate the Kronecker products in (A.1). Fortunately, we can use (1.2) again to have

$$\left(\otimes_{n=N}^1 \mathbf{A}_n^\top \mathbf{A}_n\right)\mathrm{vec}(\mathcal{C}) = \mathrm{vec}\left(\mathcal{C} \times_1 \mathbf{A}_1^\top \mathbf{A}_1 \cdots \times_N \mathbf{A}_N^\top \mathbf{A}_N\right)$$

and

$$\left(\otimes_{n=N}^1 \mathbf{A}_n^\top\right)\mathrm{vec}(\mathcal{M}) = \mathrm{vec}\left(\mathcal{M} \times_1 \mathbf{A}_1^\top \cdots \times_N \mathbf{A}_N^\top\right).$$

Hence, we have from (A.1) and the above two equalities that

$$\nabla_{\mathcal{C}}\ell(\mathcal{C}, \mathbf{A}) = \mathcal{C} \times_1 \mathbf{A}_1^\top \mathbf{A}_1 \cdots \times_N \mathbf{A}_N^\top \mathbf{A}_N - \mathcal{M} \times_1 \mathbf{A}_1^\top \cdots \times_N \mathbf{A}_N^\top. \tag{A.2}$$

Computation of $\nabla_{\mathbf{A}_n}\ell$. According to (1.4), we have

$$\ell(\mathcal{C}, \mathbf{A}) = \frac{1}{2}\left\|\mathbf{A}_n \mathbf{C}_{(n)}\left(\otimes_{\substack{i=N \\ i \neq n}}^1 \mathbf{A}_i\right)^\top - \mathbf{M}_{(n)}\right\|_F^2. \tag{A.3}$$

Hence,

$$\nabla_{\mathbf{A}_n}\ell(\mathcal{C}, \mathbf{A}) = \mathbf{A}_n(\mathbf{B}_n \mathbf{B}_n^\top) - \mathbf{M}_{(n)}\mathbf{B}_n^\top \tag{A.4}$$

where

$$\mathbf{B}_n = \mathbf{C}_{(n)}\left(\otimes_{\substack{i=N \\ i \neq n}}^1 \mathbf{A}_i\right)^\top. \tag{A.5}$$

Similar to what has been done to (A.1), we do not explicitly reformulate the Kronecker product in (A.5) but let

$$\mathcal{X} = \mathcal{C} \times_1 \mathbf{A}_1 \cdots \times_{n-1} \mathbf{A}_{n-1} \times_{n+1} \mathbf{A}_{n+1} \cdots \times_N \mathbf{A}_N. \tag{A.6}$$

Then we have $\mathbf{B}_n = \mathbf{X}_{(n)}$ according to (1.4).

Appendix B. Complexity analysis of Algorithm 1. The main cost of Algorithm 1 lies in computing $\nabla_{\boldsymbol{\mathcal{C}}}\ell(\boldsymbol{\mathcal{C}}, \mathbf{A})$ and $\nabla_{\mathbf{A}_n}\ell(\boldsymbol{\mathcal{C}}, \mathbf{A})$, which are required in (2.7) and (2.8), respectively. Note that we have omitted all superscripts for simplicity. Through (A.2), the computation of $\nabla_{\boldsymbol{\mathcal{C}}}\ell(\boldsymbol{\mathcal{C}}, \mathbf{A})$ requires

$$C\left(\sum_{j=1}^{N} R_j^2 I_j + \sum_{j=1}^{N} R_j \prod_{i=1}^{N} R_i + \sum_{j=1}^{N}\big(\prod_{i=1}^{j} R_i\big)\big(\prod_{i=j}^{N} I_i\big)\right) \tag{B.1}$$

flops, where $C \approx 2$, the first part comes from the computation of all $\mathbf{A}_i^\top \mathbf{A}_i$'s, and the second and third parts are respectively from the computations of the first and second terms in (A.2). Disregarding[12] the time for unfolding a tensor and using (A.4), we have the cost for $\nabla_{\mathbf{A}_n}\ell(\boldsymbol{\mathcal{C}}, \mathbf{A})$ to be

$$C\left(\underbrace{\sum_{j=1}^{n-1}\big(\prod_{i=1}^{j} I_i\big)\big(\prod_{i=j}^{N} R_i\big) + R_n\big(\prod_{i=1}^{n-1} I_i\big) \sum_{j=n+1}^{N}\big(\prod_{i=n+1}^{j} I_i\big)\big(\prod_{i=j}^{N} R_i\big)}_{\text{part 1}}\right.$$

$$\left. + \underbrace{R_n^2 \prod_{i\neq n} I_i + R_n^2 I_n}_{\text{part 2}} + \underbrace{R_n \prod_{i=1}^{N} I_i}_{\text{part 3}}\right), \tag{B.2}$$

where C is the same as that in (B.1), "part 1" is for the computation of $\mathbf{B}_n$ via (A.6), "part 2" and "part 3" are respectively from the computations of the first and second terms in (A.4).

Suppose $R_i < I_i$ for all $i = 1, \cdots, N$. Then the quantity of (B.1) is dominated by the third part because in this case,

$$R_j^2 I_j < \big(\prod_{i=1}^{j} R_i\big)\big(\prod_{i=j}^{N} I_i\big), \qquad R_j \prod_{i=1}^{N} R_i < \big(\prod_{i=1}^{j} R_i\big)\big(\prod_{i=j}^{N} I_i\big).$$

The quantity of (B.2) is dominated by the first and third parts. Only taking account of the dominating terms, we claim that the quantities of (B.1) and (B.2) are similar. To see this, assume $R_i = R, I_i = I$, for all i's. Then the third part of (B.1) is $\sum_{j=1}^{N} R^j I^{N-j+1}$, and the sum of the first and third parts of (B.2) is

$$\sum_{j=1}^{n-1}\big(\prod_{i=1}^{j} I_i\big)\big(\prod_{i=j}^{N} R_i\big) + R_n\big(\prod_{i=1}^{n-1} I_i\big) \sum_{j=n+1}^{N}\big(\prod_{i=n+1}^{j} I_i\big)\big(\prod_{i=j}^{N} R_i\big) + R_n \prod_{i=1}^{N} I_i$$

$$= \sum_{j=1}^{n-1} I^j R^{N-j+1} + \sum_{j=n+1}^{N} I^{j-1} R^{N-j+2} + RI^N$$

$$= \sum_{j=N-n+2}^{N} R^j I^{N-j+1} + \sum_{j=2}^{N-n+1} R^j I^{N-j+1} + RI^N$$

$$= \sum_{j=1}^{N} R^j I^{N-j+1}.$$

---

[12]In tensor-matrix multiplications, unfolding and folding a tensor both happens, and they can take about a half of time in the whole process of tensor-matrix multiplication. The readers can refer to [31] for issues about the cost of tensor unfolding and permutation.

Hence, the costs for computing $\nabla_{\mathcal{C}}\ell(\mathcal{C}, \mathbf{A})$ and $\nabla_{\mathbf{A}_n}\ell(\mathcal{C}, \mathbf{A})$ are similar.

After obtaining the partial gradients $\nabla_{\mathcal{C}}\ell(\mathcal{C}, \mathbf{A})$ and $\nabla_{\mathbf{A}_n}\ell(\mathcal{C}, \mathbf{A})$, it remains to do some projections to nonnegative orthant to finish the updates in (2.7) and (2.8), and the cost is proportional to the size of $\mathcal{C}$ and $\mathbf{A}_n$, i.e., $C_p \prod_{i=1}^N R_i$ and $C_p I_n R_n$ with $C_p \approx 4$. The data fitting term can be evaluated by

$$\ell(\mathcal{C}, \mathbf{A}) = \frac{1}{2}\left(\langle \mathbf{A}_n^\top \mathbf{A}_n, \mathbf{B}_n \mathbf{B}_n^\top \rangle - 2\langle \mathbf{A}_n, \mathbf{M}_{(n)}\mathbf{B}_n^\top \rangle + \|\mathcal{M}\|_F^2\right),$$

where $\mathbf{B}_n$ is defined in (A.5). Note that $\mathbf{A}_n^\top \mathbf{A}_n$, $\mathbf{B}_n \mathbf{B}_n^\top$ and $\mathbf{M}_{(n)}\mathbf{B}_n^\top$ have been obtained during the computation of $\nabla_{\mathcal{C}}\ell(\mathcal{C}, \mathbf{A})$ and $\nabla_{\mathbf{A}_n}\ell(\mathcal{C}, \mathbf{A})$, and $\|\mathcal{M}\|_F^2$ can be pre-computed before running the algorithm. Hence, we need $C(R_n^2 + I_n R_n)$ additional flops to evaluate $\ell(\mathcal{C}, \mathbf{A})$, where $C \approx 2$. To get the objective value, we need $C(\prod_{i=1}^N R_i + \sum_{i=1}^N I_i R_i)$ more flops for the regularization terms.

Some more computations occur in choosing Lipschitz constants L_c and L_n's. When $R_n \ll I_n$ for all n, the cost for computing Lipschitz constants, projection to nonnegative orthant and objective evaluation is negligible compared to that for computing partial gradients $\nabla_{\mathcal{C}}\ell(\mathcal{C}, \mathbf{A})$ and $\nabla_{\mathbf{A}_n}\ell(\mathcal{C}, \mathbf{A})$. Omitting the negligible cost and only accounting the main cost in (B.1) and (B.2), the per-iteration complexity of Algorithm 1 is

$$N \cdot \mathcal{O}\left(\sum_{j=1}^N (\prod_{i=1}^j R_i)(\prod_{i=j}^N I_i) + \sum_{j=1}^N (\prod_{i=1}^j I_i)(\prod_{i=j}^N R_i)\right). \tag{B.3}$$

Appendix C. Proof of Theorem 2.2.

C.1. Subsequence convergence. First, we give a subsequence convergence result, namely, any limit point of $\{\mathcal{W}^k\}$ is a stationary point. Using Lemma 2.1 of [35], we have

$$F(\mathcal{C}^{k,n-1}, \mathbf{A}_{j<n}^k, \mathbf{A}_{j\geq n}^{k-1}) - F(\mathcal{C}^{k,n}, \mathbf{A}_{j<n}^k, \mathbf{A}_{j\geq n}^{k-1})$$

$$\geq \frac{L_c^{k,n}}{2}\|\hat{\mathcal{C}}^{k,n} - \mathcal{C}^{k,n}\|_F^2 + L_c^{k,n}\left\langle \hat{\mathcal{C}}^{k,n} - \mathcal{C}^{k,n-1}, \mathcal{C}^{k,n} - \hat{\mathcal{C}}^{k,n}\right\rangle$$

$$= \frac{L_c^{k,n}}{2}\|\mathcal{C}^{k,n-1} - \mathcal{C}^{k,n}\|_F^2 - \frac{L_c^{k,n}}{2}(\omega_c^{k,n})^2\|\mathcal{C}^{k,n-2} - \mathcal{C}^{k,n-1}\|_F^2 \tag{C.1}$$

$$\geq \frac{L_c^{k,n}}{2}\|\mathcal{C}^{k,n-1} - \mathcal{C}^{k,n}\|_F^2 - \frac{L_c^{k,n-1}}{2}\delta_\omega^2\|\mathcal{C}^{k,n-2} - \mathcal{C}^{k,n-1}\|_F^2, \tag{C.2}$$

where we have used $\omega_c^{k,n} \leq \delta_\omega\sqrt{\frac{L_c^{k,n-1}}{L_c^{k,n}}}$ to get the last inequality. Note that if the re-update in Line **ReDo** is performed, then $\omega_c^{k,n} = 0$ in (C.1), and (C.2) still holds. Similarly, we have

$$F(\mathcal{C}^{k,n}, \mathbf{A}_{j<n}^k, \mathbf{A}_{j\geq n}^{k-1}) - F(\mathcal{C}^{k,n}, \mathbf{A}_{j\leq n}^k, \mathbf{A}_{j>n}^{k-1})$$
$$\geq \frac{L_n^k}{2}\|\mathbf{A}_n^{k-1} - \mathbf{A}_n^k\|_F^2 - \frac{L_n^{k-1}}{2}\delta_\omega^2\|\mathbf{A}_n^{k-2} - \mathbf{A}_n^{k-1}\|_F^2. \tag{C.3}$$

Summing (C.2) and (C.3) together over n and noting $\mathcal{C}^{k,-1} = \mathcal{C}^{k-1,N-1}, \mathcal{C}^{k,0} = \mathcal{C}^{k-1,N}$ yield

$$
\begin{aligned}
&F(\mathcal{W}^{k-1}) - F(\mathcal{W}^k) \\
&\geq \sum_{n=1}^{N} \left(\frac{L_c^{k,n}}{2} \|\mathcal{C}^{k,n-1} - \mathcal{C}^{k,n}\|_F^2 - \frac{L_c^{k,n-1}}{2} \delta_\omega^2 \|\mathcal{C}^{k,n-2} - \mathcal{C}^{k,n-1}\|_F^2 \right. \\
&\qquad \left. + \frac{L_n^k}{2} \|\mathbf{A}_n^{k-1} - \mathbf{A}_n^k\|_F^2 - \frac{L_n^{k-1}}{2} \delta_\omega^2 \|\mathbf{A}_n^{k-2} - \mathbf{A}_n^{k-1}\|_F^2 \right) \\
&= \frac{L_c^{k,N}}{2} \|\mathcal{C}^{k,N-1} - \mathcal{C}^{k,N}\|_F^2 - \frac{L_c^{k-1,N}}{2} \delta_\omega^2 \|\mathcal{C}^{k-1,N-1} - \mathcal{C}^{k-1,N}\|_F^2 \\
&\quad + \sum_{n=1}^{N-1} \frac{(1-\delta_\omega^2) L_c^{k,n}}{2} \|\mathcal{C}^{k,n-1} - \mathcal{C}^{k,n}\|_F^2 \\
&\quad + \sum_{n=1}^{N} \left(\frac{L_n^k}{2} \|\mathbf{A}_n^{k-1} - \mathbf{A}_n^k\|_F^2 - \frac{L_n^{k-1}}{2} \delta_\omega^2 \|\mathbf{A}_n^{k-2} - \mathbf{A}_n^{k-1}\|_F^2 \right).
\end{aligned}
\tag{C.4}
$$

Summing (C.4) over k, we have

$$
\begin{aligned}
&F(\mathcal{W}^0) - F(\mathcal{W}^K) \\
&\geq \sum_{k=1}^{K} \sum_{n=1}^{N} \left(\frac{(1-\delta_\omega^2) L_c^{k,n}}{2} \|\mathcal{C}^{k,n-1} - \mathcal{C}^{k,n}\|_F^2 + \frac{(1-\delta_\omega^2) L_n^k}{2} \|\mathbf{A}_n^{k-1} - \mathbf{A}_n^k\|_F^2 \right) \\
&\geq \frac{(1-\delta_\omega^2) L_d}{2} \sum_{k=1}^{K} \sum_{n=1}^{N} \left(\|\mathcal{C}^{k,n-1} - \mathcal{C}^{k,n}\|_F^2 + \|\mathbf{A}_n^{k-1} - \mathbf{A}_n^k\|_F^2 \right).
\end{aligned}
\tag{C.5}
$$

Letting $K \to \infty$ and observing F is lower bounded, we have

$$
\sum_{k=1}^{\infty} \sum_{n=1}^{N} \left(\|\mathcal{C}^{k,n-1} - \mathcal{C}^{k,n}\|_F^2 + \|\mathbf{A}_n^{k-1} - \mathbf{A}_n^k\|_F^2 \right) < \infty.
\tag{C.6}
$$

Suppose $\bar{\mathcal{W}} = (\bar{\mathcal{C}}, \bar{\mathbf{A}}_1, \cdots, \bar{\mathbf{A}}_N)$ is a limit point of $\{\mathcal{W}^k\}$. Then there is a subsequence $\{\mathcal{W}^{k'}\}$ converging to $\bar{\mathcal{W}}$. Since $\{L_c^{k,n}, L_n^k\}$ is bounded, passing another subsequence if necessary, we assume $L_c^{k',n} \to \bar{L}_c^n$ and $L_n^{k'} \to \bar{L}_n$. Note that (C.6) implies $\mathbf{A}^{k'-1} \to \bar{\mathbf{A}}$ and $\mathcal{C}^{m,n} \to \bar{\mathcal{C}}$ for all n and $m = k', k'-1, k'-2$, as $k \to \infty$. Hence, $\hat{\mathcal{C}}^{k',n} \to \bar{\mathcal{C}}$ for all n, as $k \to \infty$. Recall that

$$
\mathcal{C}^{k',n} = \underset{\mathcal{C} \geq 0}{\operatorname{argmin}} \left\langle \nabla_{\mathcal{C}} \ell(\hat{\mathcal{C}}^{k',n}, \mathbf{A}_{j<n}^{k'}, \mathbf{A}_{j\geq n}^{k'-1}), \mathcal{C} - \hat{\mathcal{C}}^{k',n} \right\rangle + \frac{L_c^{k',n}}{2} \|\mathcal{C} - \hat{\mathcal{C}}^{k',n}\|_F^2 + \lambda_c \|\mathcal{C}\|_1.
\tag{C.7}
$$

Letting $k \to \infty$ and using the continuity of the objective in (C.7) give

$$
\bar{\mathcal{C}} = \underset{\mathcal{C} \geq 0}{\operatorname{argmin}} \left\langle \nabla_{\mathcal{C}} \ell(\bar{\mathcal{C}}, \bar{\mathbf{A}}), \mathcal{C} - \bar{\mathcal{C}} \right\rangle + \frac{\bar{L}_c^n}{2} \|\mathcal{C} - \bar{\mathcal{C}}\|_F^2 + \lambda_c \|\mathcal{C}\|_1.
$$

Hence, $\bar{\mathcal{C}}$ satisfies the first-order optimality condition

$$
\left\langle \nabla_{\mathcal{C}} \ell(\bar{\mathcal{C}}, \bar{\mathbf{A}}) + \lambda_c \mathcal{P}_c, \mathcal{C} - \bar{\mathcal{C}} \right\rangle \geq 0, \text{ for all } \mathcal{C} \geq 0, \text{ some } \mathcal{P}_c \in \partial\|\bar{\mathcal{C}}\|_1.
\tag{C.8}
$$

Similarly, we have for all n that

$$
\left\langle \nabla_{\mathbf{A}_n} \ell(\bar{\mathcal{C}}, \bar{\mathbf{A}}) + \lambda_n \mathbf{P}_n, \mathbf{A}_n - \bar{\mathbf{A}}_n \right\rangle \geq 0, \text{ for all } \mathbf{A}_n \geq 0, \text{ some } \mathbf{P}_n \in \partial\|\bar{\mathbf{A}}_n\|_1.
\tag{C.9}
$$

Note (C.8) together with (C.9) gives the first-order optimality conditions of (1.5). Hence, $\bar{\mathcal{W}}$ is a stationary point.

C.2. Global convergence. Next we show the entire sequence $\{\mathcal{W}^k\}$ converges to a limit point $\bar{\mathcal{W}}$. Since all $\lambda_c, \lambda_1, \cdots, \lambda_N$ are positive, the sequence $\{\mathcal{W}^k\}$ is bounded and admits a finite limit point $\bar{\mathcal{W}}$. Let $E = \{\mathcal{W} : \|\mathcal{W}\|_F \leq 4\nu\}$, where $\|\mathcal{W}\|_F \triangleq \sqrt{\|\mathcal{C}\|_F^2 + \|\mathbf{A}\|_F^2}$ and ν is a constant such that $\|(\mathcal{C}^{k,n}, \mathbf{A}^k)\|_F \leq \nu$ for all k, n. Let L_G be a uniform Lipschitz constant of $\nabla_{\mathcal{C}}\ell(\mathcal{W})$ and $\nabla_{\mathbf{A}_n}\ell(\mathcal{W}), n = 1, \cdots, N$, over E, namely,

$$\|\nabla_{\mathcal{C}}\ell(\mathcal{Y}) - \nabla_{\mathcal{C}}\ell(\mathcal{Z})\|_F \leq L_G\|\mathcal{Y} - \mathcal{Z}\|_F, \ \forall \mathcal{Y}, \mathcal{Z} \in E, \tag{C.10a}$$

$$\|\nabla_{\mathbf{A}_n}\ell(\mathcal{Y}) - \nabla_{\mathbf{A}_n}\ell(\mathcal{Z})\|_F \leq L_G\|\mathcal{Y} - \mathcal{Z}\|_F, \ \forall \mathcal{Y}, \mathcal{Z} \in E, \ \forall n, \tag{C.10b}$$

Let

$$H(\mathcal{C}, \mathbf{A}) = \ell(\mathcal{C}, \mathbf{A}) + \lambda_c\|\mathcal{C}\|_1 + \delta_+(\mathcal{C}) + \sum_{n=1}^{N}\left(\lambda_n\|\mathbf{A}_n\|_1 + \delta_+(\mathbf{A}_n)\right)$$

and

$$r_c(\mathcal{C}) = \lambda_c\|\mathcal{C}\|_1 + \delta_+(\mathcal{C}), \quad r_n(\mathbf{A}_n) = \lambda_n\|\mathbf{A}_n\|_1 + \delta_+(\mathbf{A}_n), \ n = 1, \cdots, N,$$

where $\delta_+(\cdot)$ is the indicator function on nonnegative orthant, namely, it equals *zero* if the argument is component-wise nonnegative and $+\infty$ otherwise.

Note that (1.5) is equivalent to

$$\min_{\mathcal{C}, \mathbf{A}} H(\mathcal{C}, \mathbf{A}). \tag{C.11}$$

Recall that H satisfies the KL property (see [4,25] for example) at $\bar{\mathcal{W}}$, namely, there exist $\gamma, \rho > 0$, $\theta \in [0, 1)$, and a neighborhood $B(\bar{\mathcal{W}}, \rho) \triangleq \{\mathcal{W} : \|\mathcal{W} - \bar{\mathcal{W}}\|_F \leq \rho\}$ such that

$$|H(\mathcal{W}) - H(\bar{\mathcal{W}})|^\theta \leq \gamma \cdot \mathrm{dist}(\mathbf{0}, \partial H(\mathcal{W})), \ \text{for all } \mathcal{W} \in B(\bar{\mathcal{W}}, \rho). \tag{C.12}$$

Denote $H_k = H(\mathcal{W}^k) - H(\bar{\mathcal{W}})$. Then $H_k \downarrow 0$. Since $\bar{\mathcal{W}}$ is a limit point of $\{\mathcal{W}^k\}$ and $\|\mathbf{A}^k - \mathbf{A}^{k+1}\|_F \to 0, \|\mathcal{C}^{k,n-1} - \mathcal{C}^{k,n}\|_F \to 0$ for all k, n from (C.6), for any $T > 0$, there must exist k_0 such that $\mathcal{W}^j \in B(\bar{\mathcal{W}}, \rho), j = k_0, k_0 + 1, k_0 + 2$ and

$$T\left(H_{k_0}^{1-\theta} + \|\mathbf{A}^{k_0} - \mathbf{A}^{k_0+1}\|_F + \|\mathbf{A}^{k_0+1} - \mathbf{A}^{k_0+2}\|_F + \|\mathcal{C}^{k_0+2,N-1} - \mathcal{C}^{k_0+2,N}\|_F\right)$$
$$+ \|\mathcal{W}^{k_0+2} - \bar{\mathcal{W}}\|_F < \rho.$$

Take T as specified in (C.24) and consider the sequence $\{\mathcal{W}^k\}_{k \geq k_0}$, which is equivalent to starting the algorithm from $\mathcal{W}^{k_0}$ and, thus without loss of generality, let $k_0 = 0$, namely, $\mathcal{W}^j \in B(\bar{\mathcal{W}}, \rho), j = 0, 1, 2$, and

$$T\left(H_0^{1-\theta} + \|\mathbf{A}^0 - \mathbf{A}^1\|_F + \|\mathbf{A}^1 - \mathbf{A}^2\|_F + \|\mathcal{C}^{2,N-1} - \mathcal{C}^{2,N}\|_F\right) + \|\mathcal{W}^2 - \bar{\mathcal{W}}\|_F < \rho. \tag{C.13}$$

The idea of our proof is to show

$$\mathcal{W}^k \in B(\bar{\mathcal{W}}, \rho), \ \text{for all } k, \tag{C.14}$$

and employ the KL inequality (C.12) to show $\{\mathcal{W}^k\}$ is a Cauchy sequence, thus the entire sequence converges. Assume $\mathcal{W}^k \in B(\bar{\mathcal{W}}, \rho)$ for $0 \leq k \leq K$. We go to show $\mathcal{W}^{K+1} \in B(\bar{\mathcal{W}}, \rho)$ and conclude (C.14) by induction.

Note that

$$\partial H(\mathcal{W}^k) = \left\{\partial r_1(\mathbf{A}_1^k) + \nabla_{\mathbf{A}_1}\ell(\mathcal{W}^k)\right\} \times \cdots \times \left\{\partial r_N(\mathbf{A}_N^k) + \nabla_{\mathbf{A}_N}\ell(\mathcal{W}^k)\right\} \times \left\{\partial r_c(\mathcal{C}^{k,N}) + \nabla_{\mathcal{C}}\ell(\mathcal{W}^k)\right\},$$

19

and for all n and k

$$
\begin{aligned}
-L_n^k(\mathbf{A}_n^k - \hat{\mathbf{A}}_n^k) - \nabla_{\mathbf{A}_n}\ell(\boldsymbol{\mathcal{C}}^{k,n}, \mathbf{A}_{j<n}^k, \hat{\mathbf{A}}_n^k, \mathbf{A}_{j\geq n}^{k-1}) + \nabla_{\mathbf{A}_n}\ell(\boldsymbol{\mathcal{W}}^k) &\in \partial r_n(\mathbf{A}_n^k) + \nabla_{\mathbf{A}_n}\ell(\boldsymbol{\mathcal{W}}^k), \\
-L_c^{k,N}(\boldsymbol{\mathcal{C}}^{k,N} - \hat{\boldsymbol{\mathcal{C}}}^{k,N}) - \nabla_{\boldsymbol{\mathcal{C}}}\ell(\hat{\boldsymbol{\mathcal{C}}}^{k,N}, \mathbf{A}_{j<N}^k, \mathbf{A}_N^{k-1}) + \nabla_{\boldsymbol{\mathcal{C}}}\ell(\boldsymbol{\mathcal{W}}^k) &\in \partial r_c(\boldsymbol{\mathcal{C}}^{k,N}) + \nabla_{\boldsymbol{\mathcal{C}}}\ell(\boldsymbol{\mathcal{W}}^k).
\end{aligned}
$$

Hence, for all $k \leq K$,

$$
\begin{aligned}
&\mathrm{dist}\big(\mathbf{0}, \partial H(\boldsymbol{\mathcal{W}}^k)\big) \\
\leq& \big\|(L_1^k(\mathbf{A}_1^k - \hat{\mathbf{A}}_1^k), \cdots, L_1^k(\mathbf{A}_1^k - \hat{\mathbf{A}}_1^k), L_c^{k,n}(\boldsymbol{\mathcal{C}}^{k,N} - \hat{\boldsymbol{\mathcal{C}}}^{k,N}))\big\|_F \\
&+ \sum_{n=1}^N \big\|\nabla_{\mathbf{A}_n}\ell(\boldsymbol{\mathcal{C}}^{k,n}, \mathbf{A}_{j<n}^k, \hat{\mathbf{A}}_n^k, \mathbf{A}_{j\geq n}^{k-1}) - \nabla_{\mathbf{A}_n}\ell(\boldsymbol{\mathcal{W}}^k)\big\|_F \\
&+ \big\|\nabla_{\boldsymbol{\mathcal{C}}}\ell(\hat{\boldsymbol{\mathcal{C}}}^{k,N}, \mathbf{A}_{j<N}^k, \mathbf{A}_N^{k-1}) - \nabla_{\boldsymbol{\mathcal{C}}}\ell(\boldsymbol{\mathcal{W}}^k)\big\|_F \\
\leq& L_u\big(\|\mathbf{A}^k - \mathbf{A}^{k-1}\|_F + \|\mathbf{A}^{k-1} - \mathbf{A}^{k-2}\|_F\big) + L_u\big(\|\boldsymbol{\mathcal{C}}^{k,N} - \boldsymbol{\mathcal{C}}^{k,N-1}\|_F + \|\boldsymbol{\mathcal{C}}^{k,N-1} - \boldsymbol{\mathcal{C}}^{k,N-2}\|_F\big) \\
&+ \sum_{n=1}^N L_G\big(\|\boldsymbol{\mathcal{C}}^{k,n} - \boldsymbol{\mathcal{C}}^{k,N}\|_F + \|\mathbf{A}^k - \mathbf{A}^{k-1}\|_F + \|\mathbf{A}^{k-1} - \mathbf{A}^{k-2}\|_F\big) \\
&+ L_G\big(\|\boldsymbol{\mathcal{C}}^{k,N} - \boldsymbol{\mathcal{C}}^{k,N-1}\|_F + \|\boldsymbol{\mathcal{C}}^{k,N-1} - \boldsymbol{\mathcal{C}}^{k,N-2}\|_F + \|\mathbf{A}^k - \mathbf{A}^{k-1}\|_F\big) \\
\leq& \big(L_u + (N+1)L_G\big)\Big(\|\mathbf{A}^k - \mathbf{A}^{k-1}\|_F + \|\mathbf{A}^{k-1} - \mathbf{A}^{k-2}\|_F \\
&\quad + \|\boldsymbol{\mathcal{C}}^{k,N} - \boldsymbol{\mathcal{C}}^{k,N-1}\|_F + \sum_{n=1}^{N-1}\|\boldsymbol{\mathcal{C}}^{k,n-1} - \boldsymbol{\mathcal{C}}^{k,n}\|_F\Big),
\end{aligned}
\tag{C.15}
$$

where we have used $L_n^k, L_c^{k,n} \leq L_u$, $\forall k, n$ and (C.10) to have the second inequality, and the third inequality is obtained from $\|\boldsymbol{\mathcal{C}}^{k,n} - \boldsymbol{\mathcal{C}}^{k,N}\|_F \leq \sum_{i=n}^{N-1}\|\boldsymbol{\mathcal{C}}^{k,i} - \boldsymbol{\mathcal{C}}^{k,i+1}\|_F$ and doing some simplification. Using the KL inequality (C.12) at $\boldsymbol{\mathcal{W}} = \boldsymbol{\mathcal{W}}^k$ and the inequality

$$
\frac{s^\theta}{1-\theta}(s^{1-\theta} - t^{1-\theta}) \geq s - t, \ \forall s, t \geq 0,
$$

we get

$$
\frac{\gamma}{1-\theta}\mathrm{dist}(\mathbf{0}, \partial H(\boldsymbol{\mathcal{W}}^k))(H_k^{1-\theta} - H_{k+1}^{1-\theta}) \geq H_k - H_{k+1}.
\tag{C.16}
$$

By (C.4), we have

$$
\begin{aligned}
H_k - H_{k+1} \geq& \frac{L_c^{k+1,N}}{2}\|\boldsymbol{\mathcal{C}}^{k+1,N-1} - \boldsymbol{\mathcal{C}}^{k+1,N}\|_F^2 - \frac{L_c^{k,N}}{2}\delta_\omega^2\|\boldsymbol{\mathcal{C}}^{k,N-1} - \boldsymbol{\mathcal{C}}^{k,N}\|_F^2 \\
&+ \sum_{n=1}^{N-1}\frac{(1-\delta_\omega^2)L_c^{k+1,n}}{2}\|\boldsymbol{\mathcal{C}}^{k+1,n-1} - \boldsymbol{\mathcal{C}}^{k+1,n}\|_F^2 \\
&+ \sum_{n=1}^N\left(\frac{L_n^{k+1}}{2}\|\mathbf{A}_n^k - \mathbf{A}_n^{k+1}\|_F^2 - \frac{L_n^k}{2}\delta_\omega^2\|\mathbf{A}_n^{k-1} - \mathbf{A}_n^k\|_F^2\right).
\end{aligned}
\tag{C.17}
$$

Combining (C.15), (C.16), (C.17) and noting $L_c^{k+1,n} \geq L_d$ yield

$$
\begin{aligned}
\frac{\gamma(L_u + (N+1)L_G)}{1-\theta}&(H_k^{1-\theta} - H_{k+1}^{1-\theta})\big[\|\mathbf{A}^k - \mathbf{A}^{k-1}\|_F + \|\mathbf{A}^{k-1} - \mathbf{A}^{k-2}\|_F \\
&+ \|\boldsymbol{\mathcal{C}}^{k,N} - \boldsymbol{\mathcal{C}}^{k,N-1}\|_F + \sum_{n=1}^{N-1} \|\boldsymbol{\mathcal{C}}^{k,n-1} - \boldsymbol{\mathcal{C}}^{k,n}\|_F\big] \\
+ \delta_\omega^2 \Big\| & (\sqrt{L_1^k}\mathbf{A}_1^{k-1}, \cdots, \sqrt{L_N^k}\mathbf{A}_N^{k-1}, \sqrt{L_c^{k,N}}\boldsymbol{\mathcal{C}}^{k,N-1}) \\
& - (\sqrt{L_1^k}\mathbf{A}_1^k, \cdots, \sqrt{L_N^k}\mathbf{A}_N^k, \sqrt{L_c^{k,N}}\boldsymbol{\mathcal{C}}^{k,N}) \Big\|_F^2 \\
\geq \Big\| & (\sqrt{L_1^{k+1}}\mathbf{A}_1^k, \cdots, \sqrt{L_N^{k+1}}\mathbf{A}_N^k, \sqrt{L_c^{k+1,N}}\boldsymbol{\mathcal{C}}^{k+1,N-1}) \\
& - (\sqrt{L_1^{k+1}}\mathbf{A}_1^{k+1}, \cdots, \sqrt{L_N^{k+1}}\mathbf{A}_N^{k+1}, \sqrt{L_c^{k+1,N}}\boldsymbol{\mathcal{C}}^{k+1,N}) \Big\|_F^2 \\
+ \frac{(1-\delta_\omega^2)L_d}{2} & \sum_{n=1}^{N-1} \|\boldsymbol{\mathcal{C}}^{k+1,n-1} - \boldsymbol{\mathcal{C}}^{k+1,n}\|_F^2.
\end{aligned}
\tag{C.18}
$$

By Cauchy-Schwart inequality, we estimate

$$
\begin{aligned}
\sqrt{\text{right side of inequality (C.18)}} \\
\geq \frac{1+\delta_\omega}{2} \Big\| & (\sqrt{L_1^{k+1}}\mathbf{A}_1^k, \cdots, \sqrt{L_N^{k+1}}\mathbf{A}_N^k, \sqrt{L_c^{k+1,N}}\boldsymbol{\mathcal{C}}^{k+1,N-1}) \\
& - (\sqrt{L_1^{k+1}}\mathbf{A}_1^{k+1}, \cdots, \sqrt{L_N^{k+1}}\mathbf{A}_N^{k+1}, \sqrt{L_c^{k+1,N}}\boldsymbol{\mathcal{C}}^{k+1,N}) \Big\|_F \\
+ \eta \sum_{n=1}^{N-1} & \|\boldsymbol{\mathcal{C}}^{k+1,n-1} - \boldsymbol{\mathcal{C}}^{k+1,n}\|_F,
\end{aligned}
\tag{C.19}
$$

where $\eta > 0$ is sufficiently small and depends on δ_ω, L_d, N, and

$$
\begin{aligned}
\sqrt{\text{left side of inequality (C.18)}} \\
\leq \frac{\mu\gamma(L_u + (N+1)L_G)}{4(1-\theta)}&(H_k^{1-\theta} - H_{k+1}^{1-\theta}) \\
+ \frac{1}{\mu}\big[\|\mathbf{A}^k - \mathbf{A}^{k-1}\|_F &+ \|\mathbf{A}^{k-1} - \mathbf{A}^{k-2}\|_F + \|\boldsymbol{\mathcal{C}}^{k,N} - \boldsymbol{\mathcal{C}}^{k,N-1}\|_F + \sum_{n=1}^{N-1} \|\boldsymbol{\mathcal{C}}^{k,n-1} - \boldsymbol{\mathcal{C}}^{k,n}\|_F\big] \\
+ \delta_\omega \Big\| & (\sqrt{L_1^k}\mathbf{A}_1^{k-1}, \cdots, \sqrt{L_N^k}\mathbf{A}_N^{k-1}, \sqrt{L_c^{k,N}}\boldsymbol{\mathcal{C}}^{k,N-1}) - (\sqrt{L_1^k}\mathbf{A}_1^k, \cdots, \sqrt{L_N^k}\mathbf{A}_N^k, \sqrt{L_c^{k,N}}\boldsymbol{\mathcal{C}}^{k,N}) \Big\|_F,
\end{aligned}
\tag{C.20}
$$

where $\mu > 0$ is a sufficiently large constant such that $\frac{1}{\mu} < \min(\eta, \frac{1-\delta_\omega}{4}\sqrt{\frac{L_d}{2}})$. Combining (C.18),(C.20),

(C.19) and summing them over k from 2 to K give

$$\frac{\mu\gamma(L_u + (N+1)L_G)}{4(1-\theta)}(H_2^{1-\theta} - H_{K+1}^{1-\theta})$$

$$+ \frac{1}{\mu}\sum_{k=2}^{K}[\|\mathbf{A}^k - \mathbf{A}^{k-1}\|_F + \|\mathbf{A}^{k-1} - \mathbf{A}^{k-2}\|_F + \|\mathcal{C}^{k,N} - \mathcal{C}^{k,N-1}\|_F + \sum_{n=1}^{N-1}\|\mathcal{C}^{k,n-1} - \mathcal{C}^{k,n}\|_F]$$

$$+ \delta_\omega\sum_{k=2}^{K}\left\|(\sqrt{L_1^k}\mathbf{A}_1^{k-1}, \cdots, \sqrt{L_N^k}\mathbf{A}_N^{k-1}, \sqrt{L_c^{k,N}}\mathcal{C}^{k,N-1}) - (\sqrt{L_1^k}\mathbf{A}_1^k, \cdots, \sqrt{L_N^k}\mathbf{A}_N^k, \sqrt{L_c^{k,N}}\mathcal{C}^{k,N})\right\|_F$$

$$\geq \frac{1+\delta_\omega}{2}\sum_{k=2}^{K}\left\|(\sqrt{L_1^{k+1}}\mathbf{A}_1^k, \cdots, \sqrt{L_N^{k+1}}\mathbf{A}_N^k, \sqrt{L_c^{k+1,N}}\mathcal{C}^{k+1,N-1})\right.$$

$$\left. - (\sqrt{L_1^{k+1}}\mathbf{A}_1^{k+1}, \cdots, \sqrt{L_N^{k+1}}\mathbf{A}_N^{k+1}, \sqrt{L_c^{k+1,N}}\mathcal{C}^{k+1,N})\right\|_F + \eta\sum_{k=2}^{K}\sum_{n=1}^{N-1}\|\mathcal{C}^{k+1,n-1} - \mathcal{C}^{k+1,n}\|_F.$$

Simplifying the above inequality, we have

$$\frac{\mu\gamma(L_u + (N+1)L_G)}{4(1-\theta)}(H_2^{1-\theta} - H_{K+1}^{1-\theta})$$

$$+ \frac{1}{\mu}\sum_{k=2}^{K}\left(\|\mathbf{A}^k - \mathbf{A}^{k-1}\|_F + \|\mathbf{A}^{k-1} - \mathbf{A}^{k-2}\|_F + \|\mathcal{C}^{k,N} - \mathcal{C}^{k,N-1}\|_F\right)$$

$$+ \delta_\omega\left\|(\sqrt{L_1^2}\mathbf{A}_1^1, \cdots, \sqrt{L_N^2}\mathbf{A}_N^1, \sqrt{L_c^{2,N}}\mathcal{C}^{2,N-1}) - (\sqrt{L_1^2}\mathbf{A}_1^2, \cdots, \sqrt{L_N^2}\mathbf{A}_N^2, \sqrt{L_c^{2,N}}\mathcal{C}^{2,N})\right\|_F$$

$$\geq \frac{1+\delta_\omega}{2}\left\|(\sqrt{L_1^{K+1}}\mathbf{A}_1^K, \cdots, \sqrt{L_N^{K+1}}\mathbf{A}_N^K, \sqrt{L_c^{K+1,N}}\mathcal{C}^{K+1,N-1})\right.$$

$$\left. - (\sqrt{L_1^{K+1}}\mathbf{A}_1^{K+1}, \cdots, \sqrt{L_N^{K+1}}\mathbf{A}_N^{K+1}, \sqrt{L_c^{K+1,N}}\mathcal{C}^{K+1,N})\right\|_F \tag{C.21}$$

$$+ \frac{1-\delta_\omega}{2}\sum_{k=2}^{K-1}\left\|(\sqrt{L_1^{k+1}}\mathbf{A}_1^k, \cdots, \sqrt{L_N^{k+1}}\mathbf{A}_N^k, \sqrt{L_c^{k+1,N}}\mathcal{C}^{k+1,N-1})\right.$$

$$\left. - (\sqrt{L_1^{k+1}}\mathbf{A}_1^{k+1}, \cdots, \sqrt{L_N^{k+1}}\mathbf{A}_N^{k+1}, \sqrt{L_c^{k+1,N}}\mathcal{C}^{k+1,N})\right\|_F$$

$$+ (\eta - \frac{1}{\mu})\sum_{k=2}^{K}\sum_{n=1}^{N-1}\|\mathcal{C}^{k+1,n-1} - \mathcal{C}^{k+1,n}\|_F.$$

Note that

$$\left\|(\sqrt{L_1^{k+1}}\mathbf{A}_1^k, \cdots, \sqrt{L_N^{k+1}}\mathbf{A}_N^k, \sqrt{L_c^{k+1,N}}\mathcal{C}^{k+1,N-1})\right.$$

$$\left. - (\sqrt{L_1^{k+1}}\mathbf{A}_1^{k+1}, \cdots, \sqrt{L_N^{k+1}}\mathbf{A}_N^{k+1}, \sqrt{L_c^{k+1,N}}\mathcal{C}^{k+1,N})\right\|_F^2$$

$$= \sum_{n=1}^{N}L_n^{k+1}\|\mathbf{A}_n^k - \mathbf{A}_n^{k+1}\|_F^2 + L_c^{k+1,N}\|\mathcal{C}^{k+1,N-1} - \mathcal{C}^{k+1,N}\|_F^2$$

$$\geq L_d(\|\mathbf{A}^k - \mathbf{A}^{k+1}\|_F^2 + \|\mathcal{C}^{k+1,N-1} - \mathcal{C}^{k+1,N}\|_F^2)$$

$$\geq \frac{L_d}{2}(\|\mathbf{A}^k - \mathbf{A}^{k+1}\|_F + \|\mathcal{C}^{k+1,N-1} - \mathcal{C}^{k+1,N}\|_F)^2 \tag{C.22}$$

22

Plugging (C.22) to inequality (C.21) gives

$$\frac{\mu\gamma(L_u + (N+1)L_G)}{4(1-\theta)}(H_2^{1-\theta} - H_{K+1}^{1-\theta})$$

$$+ \frac{1}{\mu}\sum_{k=2}^{K}\Big(\|\mathbf{A}^k - \mathbf{A}^{k-1}\|_F + \|\mathbf{A}^{k-1} - \mathbf{A}^{k-2}\|_F + \|\boldsymbol{\mathcal{C}}^{k,N} - \boldsymbol{\mathcal{C}}^{k,N-1}\|_F\Big)$$

$$+ \delta_\omega\|(\sqrt{L_1^2}\mathbf{A}_1^1, \cdots, \sqrt{L_N^2}\mathbf{A}_N^1, \sqrt{L_c^{2,N}}\boldsymbol{\mathcal{C}}^{2,N-1}) - (\sqrt{L_1^2}\mathbf{A}_1^2, \cdots, \sqrt{L_N^2}\mathbf{A}_N^2, \sqrt{L_c^{2,N}}\boldsymbol{\mathcal{C}}^{2,N})\|_F$$

$$\geq \frac{1+\delta_\omega}{2}\sqrt{\frac{L_d}{2}}(\|\mathbf{A}^K - \mathbf{A}^{K+1}\|_F + \|\boldsymbol{\mathcal{C}}^{K+1,N-1} - \boldsymbol{\mathcal{C}}^{K+1,N}\|_F)$$

$$+ \frac{1-\delta_\omega}{2}\sqrt{\frac{L_d}{2}}\sum_{k=2}^{K-1}(\|\mathbf{A}^k - \mathbf{A}^{k+1}\|_F + \|\boldsymbol{\mathcal{C}}^{k+1,N-1} - \boldsymbol{\mathcal{C}}^{k+1,N}\|_F)$$

$$+ (\eta - \frac{1}{\mu})\sum_{k=2}^{K}\sum_{n=1}^{N-1}\|\boldsymbol{\mathcal{C}}^{k+1,n-1} - \boldsymbol{\mathcal{C}}^{k+1,n}\|_F,$$

which implies by noting $H_0 \geq H_k \geq 0$, $\boldsymbol{\mathcal{C}}^{k+1,0} = \boldsymbol{\mathcal{C}}^{k,N}$ and $L_n^k, L_c^{k,n} \leq L_u$, $\forall k, n$ that

$$\frac{\mu\gamma(L_u + (N+1)L_G)}{4(1-\theta)}H_0^{1-\theta} + \frac{1}{\mu}\Big(2\|\mathbf{A}^1 - \mathbf{A}^2\|_F + \|\mathbf{A}^0 - \mathbf{A}^1\|_F + \|\boldsymbol{\mathcal{C}}^{2,N} - \boldsymbol{\mathcal{C}}^{2,N-1}\|_F\Big)$$

$$+ \delta_\omega\sqrt{L_u}(\|\mathbf{A}^1 - \mathbf{A}^2\|_F + \|\boldsymbol{\mathcal{C}}^{2,N-1} - \boldsymbol{\mathcal{C}}^{2,N}\|_F)$$

$$\geq \frac{1+\delta_\omega}{2}\sqrt{\frac{L_d}{2}}(\|\mathbf{A}^K - \mathbf{A}^{K+1}\|_F + \|\boldsymbol{\mathcal{C}}^{K+1,N-1} - \boldsymbol{\mathcal{C}}^{K+1,N}\|_F)$$

$$+ (\frac{1-\delta_\omega}{2}\sqrt{\frac{L_d}{2}} - \frac{2}{\mu})\sum_{k=2}^{K-1}(\|\mathbf{A}^k - \mathbf{A}^{k+1}\|_F + \|\boldsymbol{\mathcal{C}}^{k+1,N-1} - \boldsymbol{\mathcal{C}}^{k+1,N}\|_F)$$

$$+ (\eta - \frac{1}{\mu})\sum_{k=2}^{K}\|\boldsymbol{\mathcal{C}}^{k,N} - \boldsymbol{\mathcal{C}}^{k+1,N-1}\|_F,$$

$$\geq \tau\big(\|\mathbf{A}^K - \mathbf{A}^{K+1}\|_F + \|\boldsymbol{\mathcal{C}}^{K,N} - \boldsymbol{\mathcal{C}}^{K+1,N}\|_F\big) \tag{C.23}$$

$$+ \tau\sum_{k=2}^{K-1}(\|\mathbf{A}^k - \mathbf{A}^{k+1}\|_F + \|\boldsymbol{\mathcal{C}}^{k,N} - \boldsymbol{\mathcal{C}}^{k+1,N}\|_F),$$

where $\tau = \min\left(\frac{1-\delta_\omega}{2}\sqrt{\frac{L_d}{2}} - \frac{2}{\mu}, \ \eta - \frac{1}{\mu}\right)$. Let

$$T = \max\left(\frac{\mu\gamma(L_u + (N+1)L_G)}{4\tau(1-\theta)}, \ \frac{1}{2\mu\tau} + \frac{\delta_\omega}{\tau}\sqrt{L_u}\right). \tag{C.24}$$

Then (C.23) implies

$$T\big(H_0^{1-\theta} + \|\mathbf{A}^0 - \mathbf{A}^1\|_F + \|\mathbf{A}^1 - \mathbf{A}^2\|_F + \|\boldsymbol{\mathcal{C}}^{2,N-1} - \boldsymbol{\mathcal{C}}^{2,N}\|_F\big)$$

$$\geq \|\boldsymbol{\mathcal{W}}^K - \boldsymbol{\mathcal{W}}^{K+1}\|_F + \sum_{k=2}^{K-1}\|\boldsymbol{\mathcal{W}}^k - \boldsymbol{\mathcal{W}}^{k+1}\|_F, \tag{C.25}$$

from which we have

$$\|\boldsymbol{\mathcal{W}}^{K+1} - \bar{\boldsymbol{\mathcal{W}}}\|_F$$

$$\leq \|\boldsymbol{\mathcal{W}}^K - \boldsymbol{\mathcal{W}}^{K+1}\|_F + \sum_{k=2}^{K-1}\|\boldsymbol{\mathcal{W}}^k - \boldsymbol{\mathcal{W}}^{k+1}\|_F + \|\boldsymbol{\mathcal{W}}^2 - \bar{\boldsymbol{\mathcal{W}}}\|_F$$

$$\leq T\big(H_0^{1-\theta} + \|\mathbf{A}^0 - \mathbf{A}^1\|_F + \|\mathbf{A}^1 - \mathbf{A}^2\|_F + \|\boldsymbol{\mathcal{C}}^{2,N-1} - \boldsymbol{\mathcal{C}}^{2,N}\|_F\big) + \|\boldsymbol{\mathcal{W}}^2 - \bar{\boldsymbol{\mathcal{W}}}\|_F < \rho.$$

Hence, $\boldsymbol{\mathcal{W}}^{K+1} \in B(\bar{\boldsymbol{\mathcal{W}}}, \rho)$. By induction, we have $\boldsymbol{\mathcal{W}}^k \in B(\bar{\boldsymbol{\mathcal{W}}}, \rho)$ for all k, so (C.25) holds for all K. Letting $K \to \infty$ gives $\sum_{k=2}^{\infty} \|\boldsymbol{\mathcal{W}}^k - \boldsymbol{\mathcal{W}}^{k+1}\|_F < \infty$, namely, $\{\boldsymbol{\mathcal{W}}^k\}$ is a Cauchy sequence and, thus $\boldsymbol{\mathcal{W}}^k$ converges. Since $\bar{\boldsymbol{\mathcal{W}}}$ is a limit point of $\{\boldsymbol{\mathcal{W}}^k\}$, then $\boldsymbol{\mathcal{W}}^k \to \bar{\boldsymbol{\mathcal{W}}}$. This completes the proof.

REFERENCES

[1] G.I. ALLEN, *Sparse higher-order principal components analysis*, in Artificial Intelligence and Statistics, 2012.

[2] B. W. BADER, T. G. KOLDA, ET AL., *Matlab tensor toolbox version 2.5*, January 2012.

[3] A. BECK AND M. TEBOULLE, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM Journal on Imaging Sciences, 2 (2009), pp. 183–202.

[4] J. BOLTE, A. DANIILIDIS, AND A. LEWIS, *The Lojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems*, SIAM Journal on Optimization, 17 (2007), pp. 1205–1223.

[5] J.D. CARROLL AND J.J. CHANG, *Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition*, Psychometrika, 35 (1970), pp. 283–319.

[6] A. CICHOCKI, D. MANDIC, A.H. PHAN, C. CAIAFA, G. ZHOU, Q. ZHAO, AND L. DE LATHAUWER, *Tensor Decompositions for Signal Processing Applications: from Two-way to Multiway Component Analysis*, IEEE SPM, (2014).

[7] A. CICHOCKI, R. ZDUNEK, A.H. PHAN, AND S. AMARI, *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*, John Wiley & Sons, (2009).

[8] F. CONG, A.H. PHAN, Q. ZHAO, Q. WU, T. RISTANIEMI, AND A. CICHOCKI, *Feature extraction by nonnegative tucker decomposition from EEG data including testing and training observations*, Neural Information Processing, (2012), pp. 166–173.

[9] L. DE LATHAUWER, B. DE MOOR, AND J. VANDEWALLE, *On the best rank-1 and rank-$(r_1, r_2, \ldots, r_n)$ approximation of higher-order tensors*, SIAM Journal on Matrix Analysis and Applications, 21 (2000), pp. 1324–1342.

[10] CHRIS HQ DING, TAO LI, AND MICHAEL I JORDAN, *Convex and semi-nonnegative matrix factorizations*, Pattern Analysis and Machine Intelligence, IEEE Transactions on, 32 (2010), pp. 45–55.

[11] D. DONOHO AND V. STODDEN, *When does non-negative matrix factorization give a correct decomposition into parts*, Advances in neural information processing systems, 16 (2003).

[12] M.P. FRIEDLANDER AND K. HATZ, *Computing non-negative tensor factorizations*, Optimisation Methods and Software, 23 (2008), pp. 631–647.

[13] R.A. HARSHMAN, *Foundations of the parafac procedure: models and conditions for an" explanatory" multimodal factor analysis*, UCLA working papers in Phonetics, 16 (1970), pp. 1–84.

[14] R.A. HORN AND C.R. JOHNSON, *Topics in matrix analysis*, Cambridge Univ. Press Cambridge etc, 1991.

[15] H.A.L. KIERS, *Joint orthomax rotation of the core and component matrices resulting from three-mode principal components analysis*, Journal of Classification, 15 (1998), pp. 245–263.

[16] H. KIM AND H. PARK, *Non-negative matrix factorization based on alternating non-negativity constrained least squares and active set method*, SIAM J. Matrix Anal. Appl, 30 (2008), pp. 713–730.

[17] J. KIM AND H. PARK, *Toward faster nonnegative matrix factorization: A new algorithm and comparisons*, in Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on, IEEE, 2008, pp. 353–362.

[18] Y.D. KIM AND S. CHOI, *Nonnegative Tucker decomposition*, in Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, IEEE, 2007, pp. 1–8.

[19] T.G. KOLDA AND B.W. BADER, *Tensor decompositions and applications*, SIAM review, 51 (2009), p. 455.

[20] D.D. LEE AND H.S. SEUNG, *Learning the parts of objects by non-negative matrix factorization*, Nature, 401 (1999), pp. 788–791.

[21] D.D. LEE AND H.S. SEUNG, *Algorithms for Non-Negative Matrix Factorization*, Advances in Neural Information Processing Systems, 13 (2001), pp. 556–562.

[22] H. LEE, Y. KIM, A. CICHOCKI, AND S. CHOI, *Nonnegative tensor factorization for continuous EEG classification*, International journal of neural systems, 17 (2007), pp. 305–317.

[23] Q. LING, Y. XU, W. YIN, AND Z. WEN, *Decentralized low-rank matrix completion*, International Conference on Acoustics, Speech, and Signal Processing (ICASSP), SPCOM-P1.4 (2012).

[24] J. LIU, J. LIU, P. WONKA, AND J. YE, *Sparse non-negative tensor factorization using columnwise coordinate descent*, Pattern Recognition, (2011).

[25] S. ŁOJASIEWICZ, *Sur la géométrie semi-et sous-analytique*, Ann. Inst. Fourier (Grenoble), 43 (1993), pp. 1575–1595.

[26] M. MØRUP, L.K. HANSEN, AND S.M. ARNFRED, *Algorithms for sparse nonnegative Tucker decompositions*, Neural computation, 20 (2008), pp. 2112–2131.

[27] P. PAATERO AND U. TAPPER, *Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values*, Environmetrics, 5 (1994), pp. 111–126.

[28] A.H. PHAN AND A. CICHOCKI, *Extended hals algorithm for nonnegative tucker decomposition and its applications for*

multiway analysis and classification, Neurocomputing, 74 (2011), pp. 1956–1969.

[29] A.H. PHAN, P. TICHAVSKY, AND A. CICHOCKI, *Damped gauss-newton algorithm for nonnegative tucker decomposition*, in Statistical Signal Processing Workshop (SSP), IEEE, 2011, pp. 665–668.

[30] IGNACIO RAMIREZ, PABLO SPRECHMANN, AND GUILLERMO SAPIRO, *Classification and clustering via dictionary learning with structured incoherence and shared features*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 3501–3508.

[31] M. SCHATZ, T. LOW, V. GEIJN, A. ROBERT, AND T. KOLDA, *Exploiting Symmetry in Tensors for High Performance: Multiplication with Symmetric Tensors*, arXiv preprint arXiv:1301.7744, (2013).

[32] A. SHASHUA AND T. HAZAN, *Non-negative tensor factorization with applications to statistics and computer vision*, in Proceedings of the 22nd international conference on Machine learning, ACM, 2005, pp. 792–799.

[33] L.R. TUCKER, *Some mathematical notes on three-mode factor analysis*, Psychometrika, 31 (1966), pp. 279–311.

[34] Z. WEN, W. YIN, AND Y. ZHANG, *Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm*, Mathematical Programming Computation, (2012), pp. 1–29.

[35] Y. XU AND W. YIN, *A block coordinate descent method for regularized multi-convex optimization with applications to nonnegative tensor factorization and completion*, SIAM Journal on Imaging Sciences, 6 (2013), pp. 1758–1789.

[36] Y. XU, W. YIN, Z. WEN, AND Y. ZHANG, *An alternating direction algorithm for matrix completion with nonnegative factors*, Journal of Frontiers of Mathematics in China, Special Issue on Computational Mathematics, 7 (2011), pp. 365–384.

[37] S. ZAFEIRIOU, *Discriminant nonnegative tensor factorization algorithms*, Neural Networks, IEEE Transactions on, 20 (2009), pp. 217–235.

[38] Q. ZHANG, H. WANG, R.J. PLEMMONS, AND V. PAUCA, *Tensor methods for hyperspectral data analysis: a space object material identification study*, JOSA A, 25 (2008), pp. 3001–3012.

[39] Y. ZHANG, *An alternating direction algorithm for nonnegative matrix factorization*, Rice Technical Report, (2010).