NEW ROBUST ALGORITHMS FOR SPARSE NON-NEGATIVE THREE-WAY TENSOR DECOMPOSITIONS

Viet-Dung Nguyen[1] *Karim Abed-Meraim*[1] *Nguyen Linh-Trung*[2]

[1] PRISME Lab., University of Orléans, France, [2] Faculty of Elec. and Telecom., VNU Univ. of Eng. & Tech., Vietnam

ABSTRACT

Tensor decomposition is an important tool for many applications in diverse disciplines such as signal processing, chemometrics, numerical linear algebra and data mining. In this work, we focus on PARAFAC and Tucker decompositions of three-way tensors with non-negativity and/or sparseness constraints. By using an all-at-once optimization approach, we propose two decomposition algorithms which are robust to tensor order over-estimation errors, a desired practical property when the tensor rank is unknown. Different algorithm versions are proposed depending on the desired constraint (or property) of the tensor factors or the core tensor. Finally, the performance of the algorithms is assessed via insightful simulation experiments on both simulated and real-life data.

***Index Terms*—** CANDECOMP/PARAFAC, Tucker decomposition, all-at-once approach, sparsity, non-negativity

1. INTRODUCTION

Tensor decomposition, as an extension of matrix decomposition to multi-way array, proves its important role on various applications in signal processing [1], chemometrics [2], numerical linear algebra [3] and data mining [4].

Two widely-used tensor decompositions are the Parallel Factor Analysis (PARAFAC) model and Tucker model one because they can be considered as generalizations of the Singular Value Decomposition (SVD) for multiway arrays. Depending on particular applications, one may choose models other than the above [1, 3]. However, in this paper, when we refer to tensor decomposition, we focus only on these two models.

To date, various methods have been introduced and developed for computing tensor decomposition. We can classify them into three main approaches: alternating approach, general optimization approach and algebraic approach. The alternating approach optimizes one factor at each step while keeping the others fixed. The general optimization approach casts the tensor decomposition problem into a nonlinear equation problem and then solve it using standard optimization tools, such as gradient methods. In a different aspect, as its name reveals, the algebraic approach aims to solve the tensor decomposition problem by using only algebra operators. We refer the reader to [5, 6] for a comparison of different algorithms for PARAFAC decomposition.

For a long time, the alternating approach is considered as the "workhorse" one because of its simplicity and relative efficiency. However, an *all-at-once* optimization framework [6], which has been recently proposed, has been reported to be as accurate as alternating-based algorithms while being more robust in the over-factoring case[1]. While estimating the rank of a tensor is NP-complete [3] and still an open problem, robustness is a desirable characteristic.

Moreover, among various constraints, two popular ones are non-negativity and sparsity because many problems come naturally with them, such as in text, image, and EEG signal analyses. As a result, many tensor decomposition algorithms with non-negativity and sparseness constraints have been proposed. We refer the reader to [1, 3, 7] for comprehensive review. For PARAFAC, imposing a non-negativity constraint, when applicable, not only improves the physical interpretation [7] but also helps to avoid diverging components [8]. Enforcing non-negativity and sparsity on factors and/or the core tensor also helps to improve the uniqueness of the Tucker decomposition [9]. However, as will be indicated in the sequel, even enforcing non-negativity and sparsity on factors, the state-of-the-art algorithms are not robust to over-factoring. Our work aims to overcome this shortcoming by bring these two desired characteristics together. In particular, we would like to have algorithms which are robust with over-factoring and easy to use with either non-negativity constraint or both non-negativity and sparseness constraints.

Our contributions are two folds: First, we tailor the all-at-one optimization framework [6] to impose non-negativity constraint or both non-negativity and sparseness constraints for tensor decomposition. Second, we apply this framework to two case studies: (i) sparse non-negative PARAFAC decomposition, and (ii) sparse non-negative Tucker decomposition. As shown in the simulation section, our sparse non-negative algorithms are as accurate as state-of-the-art algorithms but more robust to over-factoring. We would like to highlight that the all-at-once optimization approach has been applied to the PARAFAC decomposition with missing entries [10], coupled matrix/tensor decomposition without/with

We acknowledge funding by the French Agence Nationale pour la Recherche (ASTRID programme, DGA) under contract ANR-14-ASTR-0027.

[1]Over-factoring means that for example in PARAFAC we choose a tensor rank is larger than its true value.

missing value [11]. However, to our best knowledge, sparse non-negative PARAFAC and Tucker ones have not been proposed in the context of all-at-one optimization.

Notations: We follow the notations used in [3]. Calligraphic letters are used for tensors ($\mathcal{A}, \mathcal{B}, \ldots$). Matrices, vectors (both row and column), and scalars are denoted by boldface uppercase, boldface lowercase, and lowercase respectively; for example $\mathbf{A}, \mathbf{a}$, and a. Element (i, j, k) of a tensor $\mathcal{A} \in \mathbb{R}^{I \times J \times K}$ is symbolized as a_{ijk}, element (i, j) of a matrix $\mathbf{A} \in \mathbb{R}^{I \times J}$ as a_{ij}, and i-th entry of a vector $\mathbf{a} \in \mathbb{R}^{I}$ as a_i. Moreover, $\mathbf{A} \otimes \mathbf{B}$ defines the Kronecker product of $\mathbf{A}$ and $\mathbf{B}$, $\mathbf{A} \odot \mathbf{B}$ the Khatri-Rao (column-wise Kronecker) product, $\mathbf{A} * \mathbf{B}$ the Hadamard product which is the element-wise matrix product, $\mathbf{a} \circ \mathbf{b}$ the outer product of $\mathbf{a}$ and $\mathbf{b}$, and $[\mathbf{A}]^{+} = \max\{0, a_{ij}\}$, for all i, j, is a positive orthant projection of a real-valued matrix $\mathbf{A}$. A non-negative matrix $\mathbf{A}$ is denoted by $\mathbf{A} \geqslant 0$, where its entries satisfy $a_{ij} \geqslant 0$ for all i, j.

2. TENSOR OPERATORS AND MODELS

In this section, we introduce some basic tensor operators and models that we will use later. Before starting with tensor operators, we introduce some useful matrix product equality

$$\begin{aligned}
(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) &= (\mathbf{AC}) \otimes (\mathbf{BD}) \\
(\mathbf{A} \odot \mathbf{B})^T(\mathbf{A} \odot \mathbf{B}) &= (\mathbf{A}^T\mathbf{A}) * (\mathbf{B}^T\mathbf{B}) \\
\mathrm{vec}(\mathbf{ABC}^T) &= (\mathbf{C} \otimes \mathbf{A})\,\mathrm{vec}(\mathbf{B}),
\end{aligned}$$

where $\mathrm{vec}()$ performs vectorization of a matrix that stacks the columns of the matrix into a vector (*e.g.*, given $\mathbf{B} \in \mathbb{R}^{I \times J}$, $\mathrm{vec}(\mathbf{B}) = [\mathbf{b}_1^T, \cdots, \mathbf{b}_J^T]^T$).

Basic tensor operators:

Three mode-n unfoldings of a tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ are

$$\begin{aligned}
\mathbf{X}_{(1)} &: [\mathbf{X}_{(1)}]_{i, j+(k-1)J} = x_{ijk} \\
\mathbf{X}_{(2)} &: [\mathbf{X}_{(2)}]_{j, i+(k-1)I} = x_{ijk} \\
\mathbf{X}_{(3)} &: [\mathbf{X}_{(3)}]_{k, i+(j-1)I} = x_{ijk}.
\end{aligned}$$

The inner product of two same-size tensors $\mathcal{X}$, $\mathcal{Y} \in \mathbb{R}^{I \times J \times K}$ is defined as

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} x_{ijk} y_{ijk}.$$

As a consequence, we have $\langle \mathcal{X}, \mathcal{X} \rangle = \|\mathcal{X}\|^2$.

The l_1 of a tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ is defined as

$$\|\mathcal{X}\|_1 = \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} |x_{ijk}|.$$

PARAFAC and Tucker models:

The PARAFAC decomposition of $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ can be defined as

$$\mathcal{X} \approx [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \equiv \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r, \tag{1}$$

which is the sum of R rank-one tensors, with R being the tensor rank. The set of vectors, $\{\mathbf{a}_r\}, \{\mathbf{b}_r\}, \{\mathbf{c}_r\}$ can be grouped into the so-called loading matrices $\mathbf{A} = [\mathbf{a}_1 \ldots \mathbf{a}_R] \in \mathbb{C}^{I \times R}$, $\mathbf{B} = [\mathbf{b}_1 \ldots \mathbf{b}_R] \in \mathbb{C}^{J \times R}$, and $\mathbf{C} = [\mathbf{c}_1 \ldots \mathbf{c}_R] \in \mathbb{C}^{K \times R}$. Equation (1) can also be formulated in matrix form using mode-n unfolding as

$$\mathbf{X}_{(1)} \approx \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T \tag{2}$$

$$\mathbf{X}_{(2)} \approx \mathbf{B}(\mathbf{C} \odot \mathbf{A})^T \tag{3}$$

$$\mathbf{X}_{(3)} \approx \mathbf{C}(\mathbf{B} \odot \mathbf{A})^T. \tag{4}$$

The PARAFAC decomposition is generically unique (up to scales and permutation) if it satisfies the following condition [12]:

$$2R(R-1) \leqslant I(I-1)K(K-1), \qquad R \leqslant J.$$

On the other hand, the Tucker decomposition of $\mathcal{Y} \in \mathbb{R}^{I \times J \times K}$ can be written as follows:

$$\mathcal{Y} \approx [\![\mathcal{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \equiv \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} \mathbf{a}_p \circ \mathbf{b}_q \circ \mathbf{c}_r, \tag{5}$$

where $\mathbf{A} = [\mathbf{a}_1 \ldots \mathbf{a}_P] \in \mathbb{C}^{I \times P}$, $\mathbf{B} = [\mathbf{b}_1 \ldots \mathbf{b}_Q] \in \mathbb{C}^{J \times Q}$ and $\mathbf{C} = [\mathbf{c}_1 \ldots \mathbf{c}_R] \in \mathbb{C}^{K \times R}$ are the factor matrices and $\mathcal{G} \in \mathbb{C}^{P \times Q \times R}$ is called core tensor. Matrix and vector forms of (5) can be presented as

$$\mathbf{Y}_{(1)} \approx \mathbf{A}\mathbf{G}_{(1)}(\mathbf{C} \otimes \mathbf{B})^T \tag{6}$$

$$\mathbf{Y}_{(2)} \approx \mathbf{B}\mathbf{G}_{(2)}(\mathbf{C} \otimes \mathbf{A})^T \tag{7}$$

$$\mathbf{Y}_{(3)} \approx \mathbf{C}\mathbf{G}_{(3)}(\mathbf{B} \otimes \mathbf{A})^T \tag{8}$$

$$\mathbf{y} \approx (\mathbf{C} \otimes \mathbf{B} \otimes \mathbf{A})\mathbf{g}, \tag{9}$$

where $\mathbf{y} = \mathrm{vec}(\mathcal{Y})$ and $\mathbf{g} = \mathrm{vec}(\mathcal{G})$. The "standard" Tucker model sets column-wise orthogonal constraints on the factors. Those constraints simplify the calculation when implementing the algorithms but are, in general, optional. In contrast to the PARAFAC model, the Tucker model is not unique except when necessary constraints are added, for example both sparsity and non-negativity constraints.

3. ALL-AT-ONCE OPTIMIZATION WITH NONNEGATIVE AND SPARSE CONSTRAINTS

The all-at-once optimization approach [6] includes three steps:

Step 1: Define a cost function f and variable $\mathbf{x}$.

Step 2: Compute the gradient $\nabla f(\mathbf{x})$.

Step 3: Optimize using nonlinear conjugate gradient.

The special point comes from variable $\mathbf{x}$ which is a concatenation of the vectorized forms of the loading matrices for PARAFAC, or vectorized forms of the loading factors and the core tensor for Tucker (see next sections for more details).

To impose the non-negativity constraint or both the non-negativity and sparseness constraints, we propose to replace

Step 1: Define a cost function f and variable $\mathbf{x}$.
Step 2: Compute the gradient $\nabla f(\mathbf{x})$.
Step 3: Optimize using first order projected gradient.

Table 1: All-at-once optimization with non-negativity constraints

the nonlinear conjugate gradient by a first-order projected gradient method. This framework is simple and easy to implement. Moreover, it allows us to solve a large class of problems (*e.g.*, different models) with non-negativity and sparseness constraints. When the sparseness constraint comes after the non-negativity constraint, it is straightforward to calculate the gradient of variable with l_1 norm following the rule $\frac{\partial \|\mathbf{x}\|_1}{\partial \mathbf{x}} = \mathbf{1}$ where $\mathbf{1}$ is a vector[2] whose entries are one. Thus, we can use the proposed framework without modifying Step 3 in the optimization algorithm. A summary of the framework is presented in Table 1. We adapt the projected gradient algorithm [13] to use in Step 3. We choose this algorithm because it is well-understood and the results for convergence are available [14]. The algorithm is a variant of the projected gradient algorithm using the Armijo rule along the projection arc [15]. In particular, the step size in the following update step:

$$\mathbf{x}^{(k+1)} = [\mathbf{x}^k - \alpha_k \nabla f(\mathbf{x}^k)]^+ \qquad (10)$$

should be such that the following condition is satisfied:

$$f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k) \leqslant \sigma \nabla f(\mathbf{x}^k)^T (\mathbf{x}^{k+1} - \mathbf{x}^k), \qquad (11)$$

where $\alpha_k = \beta^{t_k}$ and t_k is the first integer which makes (11) hold and σ is a pre-defined positive scalar. The main idea now is to reduce the search time of α_k by using α_{k-1} as an initial guess. This search strategy stems from the fact that α_k and α_{k-1} may be close. Thus we can increase or decrease α_k until the largest β^{t_k} that satisfies (11) is found. A summary of the projected gradient is presented in Table 1.

For stopping condition, we terminate the algorithm if $\|[\nabla f(\mathbf{x}^k)]^+\| \leqslant \epsilon \|[\nabla f(\mathbf{x}^1)]\|$ or if the algorithm reaches a maximum number of iterations. The former means that we stop if a solution at the k-th step $\mathbf{x}^k$ is close to a stationary point where $\epsilon > 0$ is a small user-defined number. The latter is to avoid a too-long run-time.

4. FIRST CASE STUDY: PARAFAC MODEL

In this section, we define the cost function for the sparse non-negative PARAFAC decomposition as

$$\begin{aligned} &\text{minimize} && f(\mathbf{A},\mathbf{B},\mathbf{C}) \\ &\text{subject to} && \mathbf{A} \geqslant 0, \mathbf{B} \geqslant 0, \mathbf{C} \geqslant 0 \end{aligned}$$

where

$$\begin{aligned} f(\mathbf{A},\mathbf{B},\mathbf{C}) =& \frac{1}{2}\|\mathcal{X} - [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]\|_F^2 \\ &+ \lambda_A\|\mathbf{A}\|_1 + \lambda_B\|\mathbf{B}\|_1 + \lambda_C\|\mathbf{C}\|_1 \qquad (12) \end{aligned}$$

[2]In this paper, we will ”over-use” notation $\mathbf{1}$ which can be vector or matrix depending on the variable size

Algorithm 1: Modified projected gradient [13].

```
   Input : $\mathbf{x}_0 \in \mathbb{R}_+$
   Output: $\mathbf{x} \in \mathbb{R}_+$ such that (11) is minimized
 1 initialization for $\beta \in (0,1)$, $\sigma \in (0,1)$, $\alpha_0 = 1$
 2 while a stopping condition is not met do
 3     if (11) holds, then
 4         repeat
 5             $\alpha_k \leftarrow \alpha_k/\beta$
 6             $\hat{\mathbf{x}} \leftarrow [\mathbf{x} - \alpha_k \nabla f(\mathbf{x}^k)]^+$
 7         until (11) does not hold, or $\hat{\mathbf{x}} = \mathbf{x}$;
 8     else
 9         repeat
10             $\alpha_k \leftarrow \alpha_k \beta$
11             $\hat{\mathbf{x}} \leftarrow [\mathbf{x} - \alpha_k \nabla f(\mathbf{x}^k)]^+$
12         until (11) holds, or $\hat{\mathbf{x}} = \mathbf{x}$;
13     end
14 end
```

and λ_A, λ_B and λ_C are penalty terms which control the regularization strength. We can write out three equivalent expressions of (12) in terms of unfolded tensors as follows:

$$\begin{aligned} f(\mathbf{A},\mathbf{B},\mathbf{C}) &= \frac{1}{2}\|\mathbf{X}_{(1)} - \mathbf{A}(\mathbf{C}\odot\mathbf{B})^T\|_F^2 \\ &\quad + \lambda_A\|\mathbf{A}\|_1 + \lambda_B\|\mathbf{B}\|_1 + \lambda_C\|\mathbf{C}\|_1 \qquad (13) \\ f(\mathbf{A},\mathbf{B},\mathbf{C}) &= \frac{1}{2}\|\mathbf{X}_{(2)} - \mathbf{B}(\mathbf{C}\odot\mathbf{A})^T\|_F^2 \\ &\quad + \lambda_A\|\mathbf{A}\|_1 + \lambda_B\|\mathbf{B}\|_1 + \lambda_C\|\mathbf{C}\|_1 \qquad (14) \\ f(\mathbf{A},\mathbf{B},\mathbf{C}) &= \frac{1}{2}\|\mathbf{X}_{(3)} - \mathbf{C}(\mathbf{B}\odot\mathbf{A})^T\|_F^2 \\ &\quad + \lambda_A\|\mathbf{A}\|_1 + \lambda_B\|\mathbf{B}\|_1 + \lambda_C\|\mathbf{C}\|_1. \qquad (15) \end{aligned}$$

These expressions are convenient when computing the partial derivatives corresponding to variables $\mathbf{A}$ or $\mathbf{B}$ or $\mathbf{C}$. To be specific, we have

$$\begin{aligned} \frac{\partial f}{\partial \mathbf{A}} &= -\mathbf{X}_{(1)}(\mathbf{C}\odot\mathbf{B}) + \mathbf{A}(\mathbf{C}^T\mathbf{C} * \mathbf{B}^T\mathbf{B}) + \lambda_A\mathbf{1} \qquad (16) \\ \frac{\partial f}{\partial \mathbf{B}} &= -\mathbf{X}_{(2)}(\mathbf{C}\odot\mathbf{A}) + \mathbf{B}(\mathbf{C}^T\mathbf{C} * \mathbf{A}^T\mathbf{A}) + \lambda_B\mathbf{1} \qquad (17) \\ \frac{\partial f}{\partial \mathbf{C}} &= -\mathbf{X}_{(3)}(\mathbf{B}\odot\mathbf{A}) + \mathbf{C}(\mathbf{B}^T\mathbf{B} * \mathbf{A}^T\mathbf{A}) + \lambda_C\mathbf{1}. \qquad (18) \end{aligned}$$

We then obtain the gradient by concatenating their vectorized forms as

$$\nabla f(\mathbf{x}) = \left[\text{vec}(\frac{\partial f}{\partial \mathbf{A}})^T, \text{vec}(\frac{\partial f}{\partial \mathbf{B}})^T, \text{vec}(\frac{\partial f}{\partial \mathbf{C}})^T\right]^T, \qquad (19)$$

where $\mathbf{x} = [\text{vec}(\mathbf{A})^T, \text{vec}(\mathbf{B})^T, \text{vec}(\mathbf{C})^T]^T$. Finally, we use this gradient in Step 2 of the framework given by Table 1. We refer to this implementation as the All-at-once optimization based Sparse Nonegative PARAFAC (ASNP) algorithm.

5. SECOND CASE STUDY: TUCKER MODEL

Similar to the PARAFAC case, we present a sparse non-negative Tucker decomposition. Depending on applications, we can choose to impose a sparseness constraint on several factors and the core tensor, or only the core tensor. If dimensions of the core tensor are smaller than those of the underlying tensor, we have the standard Tucker model. Otherwise, we have a tensor dictionary learning problem. Consider the following cost function:

$$\begin{array}{ll}\text{minimize} & h(\mathbf{A},\mathbf{B},\mathbf{C},\mathcal{G}) \\ \text{subject to} & \mathbf{A}\geqslant 0,\mathbf{B}\geqslant 0,\mathbf{C}\geqslant 0,\mathcal{G}\geqslant 0\end{array} \tag{20}$$

where

$$\begin{aligned}h(\mathbf{A},\mathbf{B},\mathbf{C},\mathcal{G}) &= \frac{1}{2}\|\mathcal{Y}-[\![\mathcal{G};\mathbf{A},\mathbf{B},\mathbf{C}]\!]\|_F^2 \\ &+\lambda_A\|\mathbf{A}\|_1+\lambda_B\|\mathbf{B}\|_1+\lambda_C\|\mathbf{C}\|_1+\lambda_G\|\mathcal{G}\|_1\end{aligned} \tag{21}$$

and λ_A, λ_B, λ_C and λ_G are penalty terms of $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathcal{G}$ respectively. We can write out four equivalent expressions of (21) in terms of unfolded tensors and vectorized form as follows:

$$\begin{aligned}h(\mathbf{A},\mathbf{B},\mathbf{C},\mathcal{G}) &= \frac{1}{2}\|\mathbf{Y}_{(1)}-\mathbf{A}\mathbf{G}_{(1)}(\mathbf{C}\otimes\mathbf{B})^T\|_F^2 \\ &+\lambda_A\|\mathbf{A}\|_1+\lambda_B\|\mathbf{B}\|_1+\lambda_C\|\mathbf{C}\|_1+\lambda_G\|\mathbf{G}_{(1)}\|_1 \quad (22)\\ h(\mathbf{A},\mathbf{B},\mathbf{C},\mathcal{G}) &= \frac{1}{2}\|\mathbf{Y}_{(2)}-\mathbf{B}\mathbf{G}_{(2)}(\mathbf{C}\otimes\mathbf{A})^T\|_F^2 \\ &+\lambda_A\|\mathbf{A}\|_1+\lambda_B\|\mathbf{B}\|_1+\lambda_C\|\mathbf{C}\|_1+\lambda_G\|\mathbf{G}_{(2)}\|_1 \quad (23)\\ h(\mathbf{A},\mathbf{B},\mathbf{C},\mathcal{G}) &= \frac{1}{2}\|\mathbf{Y}_{(3)}-\mathbf{C}\mathbf{G}_{(3)}(\mathbf{B}\otimes\mathbf{A})^T\|_F^2 \\ &+\lambda_A\|\mathbf{A}\|_1+\lambda_B\|\mathbf{B}\|_1+\lambda_C\|\mathbf{C}\|_1+\lambda_G\|\mathbf{G}_{(3)}\|_1 \quad (24)\\ h(\mathbf{A},\mathbf{B},\mathbf{C},\mathcal{G}) &= \frac{1}{2}\|\mathbf{y}-(\mathbf{C}\otimes\mathbf{B}\otimes\mathbf{A})\mathbf{g}\|_F^2 \\ &+\lambda_A\|\mathbf{A}\|_1+\lambda_B\|\mathbf{B}\|_1+\lambda_C\|\mathbf{C}\|_1+\lambda_G\|\mathbf{g}\|_1. \quad (25)\end{aligned}$$

The partial derivatives of $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathcal{G}$ can then be obtained as

$$\begin{aligned}\frac{\partial h}{\partial\mathbf{A}} &= -\mathbf{X}_{(1)}(\mathbf{C}\otimes\mathbf{B})\mathbf{G}_{(1)}^T \\ &+\mathbf{A}\mathbf{G}_{(1)}(\mathbf{C}^T\mathbf{C}\otimes\mathbf{B}^T\mathbf{B})\mathbf{G}_{(1)}^T+\lambda_A\mathbf{1} \quad (26)\\ \frac{\partial h}{\partial\mathbf{B}} &= -\mathbf{X}_{(2)}(\mathbf{C}\otimes\mathbf{A})\mathbf{G}_{(2)}^T \\ &+\mathbf{B}\mathbf{G}_{(2)}(\mathbf{C}^T\mathbf{C}\otimes\mathbf{A}^T\mathbf{A})\mathbf{G}_{(2)}^T+\lambda_B\mathbf{1} \quad (27)\\ \frac{\partial h}{\partial\mathbf{C}} &= -\mathbf{X}_{(3)}(\mathbf{B}\otimes\mathbf{A})\mathbf{G}_{(3)}^T \\ &+\mathbf{C}\mathbf{G}_{(3)}(\mathbf{B}^T\mathbf{B}\otimes\mathbf{A}^T\mathbf{A})\mathbf{G}_{(3)}^T+\lambda_C\mathbf{1} \quad (28)\\ \frac{\partial h}{\partial\mathbf{g}} &= -(\mathbf{C}^T\otimes\mathbf{B}^T\otimes\mathbf{A}^T)\mathbf{g} \\ &+(\mathbf{C}^T\mathbf{C}\otimes\mathbf{B}^T\mathbf{B}\otimes\mathbf{A}^T\mathbf{A})\mathbf{g}+\lambda_G\mathbf{1}. \quad (29)\end{aligned}$$

α	β	σ	maxIter	ϵ	λ_A	λ_B	λ_C	λ_G
1	0.1	0.01	1000	10^{-4}	0.01	0	0	N/A
1	0.1	0.01	1000	10^{-4}	0	0	0	0

Table 2: Particular parameters set in our experiment.

By concatenating their vectorized form, we attain

$$\nabla h(\mathbf{x}) = \Big[\mathrm{vec}(\tfrac{\partial h}{\partial\mathbf{A}})^T,\mathrm{vec}(\tfrac{\partial h}{\partial\mathbf{B}})^T,\mathrm{vec}(\tfrac{\partial h}{\partial\mathbf{C}})^T,(\tfrac{\partial h}{\partial\mathbf{g}})^T\Big]^T, \tag{30}$$

where $\mathbf{x}=[\mathrm{vec}(\mathbf{A})^T,\mathrm{vec}(\mathbf{B})^T,\mathrm{vec}(\mathbf{C})^T,\mathbf{g}^T]^T$. We use the above gradient in place of the gradient in Step 2 of the framework of Table 1. We refer to this implementation as the All-at-once optimization based Sparse Non-negative Tucker (ASNT) algorithm.

6. SIMULATION

In this section, the performance of the proposed algorithms are assessed. We implement the considered algorithms using the Matlab Tensor Toolbox in Matlab [16]. Parameters of the algorithms are summarized in Table 2. Because of limited space, we select two experiments to compare our algorithms with several available algorithms from literature. In all experiments, we keep all default parameters of the comparing algorithms.

To assess the accuracy of ASNP[3] and its robustness, we compare our algorithm in over-factoring case with two state-of-the-art algorithms: CPOPT [6] from the Tensor Toolbox and the Non-negative Alternating Least-Squares (NALS-PARAFAC) from the N-way Toolbox [17]. In particular, we use amino acids fluorescence data from [2]. This data is a third-order tensor corresponding to 5 chemical samples measured by fluorescence at 61 excitation and 201 emission wavelengths. The true rank of the tensor is three. Since PARAFAC is unique up to scales and permutation, we follow the normalization method proposed in [2]. Particularly, we normalize the loading vectors of the second and third modes and keep the variance of the corresponding vectors in the first mode. Then we arrange them in descending order of amplitude. This normalization method was also used for CPOPT when comparing with the CP-ALS algorithm in [6]. As shown in Figure 1, our method is accurate and more robust than CPOPT and NALS-PARAFAC.

To assess the performance of ANST, we randomly generate a $20\times20\times20$ non-negative tensor following the Tucker model (5). The core tensor size is set to $5\times5\times5$. Then, the noisy observation is $\tilde{\mathcal{Y}}=\mathcal{Y}+\eta\frac{\|\mathcal{Y}\|}{\|\mathcal{N}\|}\mathcal{N}$, where $\mathcal{N}(t)$ is the noise whose size is identical to that of $\mathcal{Y}$ and parameter η controls the noise level. We compare performance of ANST with the Block Coordinate Descent based Tucker (BCDT) in [18]. The relative error was used as the performance criterion $\varrho=\frac{\|\mathcal{Y}-\mathcal{Y}_{est}\|_F}{\|\mathcal{Y}\|_F}$ where $\mathcal{Y}_{est}$ is the estimated tensor. The average result obtained from 100 Monte Carlo simulation

[3]With chosen parameters, the result of the ASNP algorithm for the non-negativity constraint is almost identical to that for both the sparseness and non-negativity constraints. Thus, we decided to present the latter only.

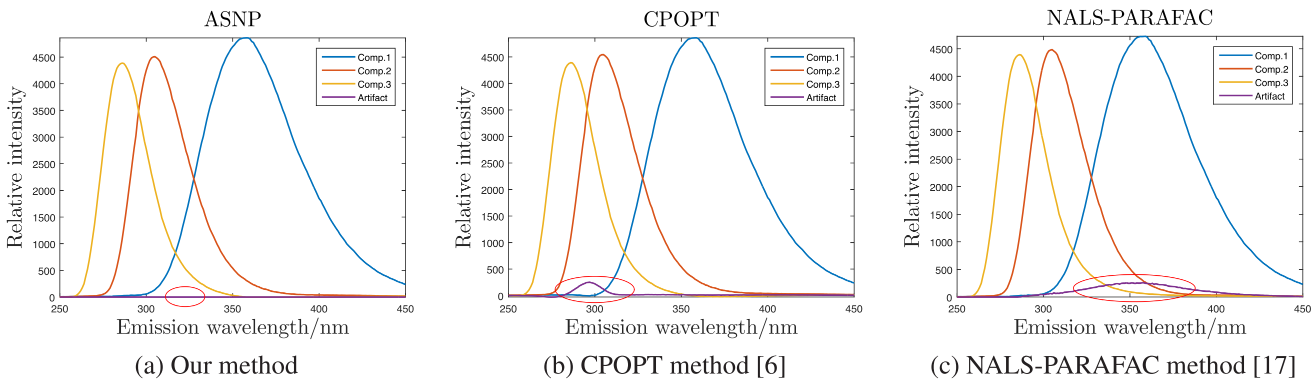

Fig. 1: Illustration of loading components estimated from three methods for overfactoring case, $R = R_{true} + 1$.

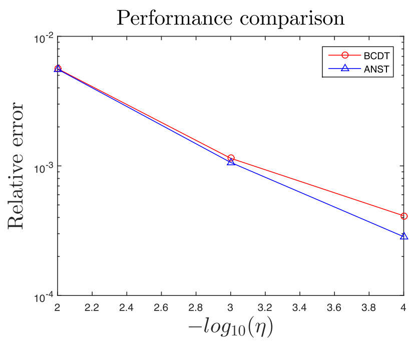

Fig. 2: Performance comparison of ANST and BCDT.

runs is shown in Figure 2. We can see that the performance of ANST is slightly better than that of BCDT. However, we confirm that our algorithm is slower than BCDT and our future work will focus on improving this drawback.

7. CONCLUSION

We have introduced in this paper two new tensor decomposition algorithms that take into account the sparsity and nonnegativity of the factors. Compared to other existing methods, the proposed solution has the advantages of simplicity and robustness to tensor-rank estimation errors. Simulation results have been provided to illustrate the effectiveness and robustness of the algorithms for both simulated and real-life data.

REFERENCES

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

[2] R. Bro, "PARAFAC. tutorial and applications," *Chemometrics and intelligent laboratory systems*, vol. 38, pp. 149–171, 1997.

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

[4] E. Acar and B. Yener, "Unsupervised multiway data analysis: A literature survey," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 21, no. 1, pp. 6–20, 2009.

[5] N. K. M. Faber, R. Bro, and P. K. Hopke, "Recent developments in candecomp/parafac algorithms: a critical review," *Chemometrics and Intelligent Laboratory Systems*, vol. 65, no. 1, pp. 119–137, 2003.

[6] E. Acar, D.M Dunlavy, and T. G. Kolda, "A scalable optimization approach for fitting canonical tensor decompositions," *J. Chemometrics*, vol. 25, no. 2, pp. 67–86, 2011.

[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] A. Stegeman, "Finding the limit of diverging components in three-way Candecomp/Parafaca demonstration of its practical merits," *Computational Statistics & Data Analysis*, vol. 75, pp. 203–216, 2014.

[9] M. Mørup, L. K. Hansen, and S. M. Arnfred, "Algorithms for sparse nonnegative Tucker decompositions," *Neural computation*, vol. 20, pp. 2112–2131, 2008.

[10] E. Acar, D. M. Dunlavy, T. G Kolda, and M. Mørup, "Scalable tensor factorizations for incomplete data," *Chemometrics and Intelligent Laboratory Systems*, vol. 106, no. 1, pp. 41–56, 2011.

[11] E. Acar, T. G. Kolda, and D. M. Dunlavy, "All-at-once optimization for coupled matrix and tensor factorizations," *arXiv preprint arXiv:1105.3422*, 2011.

[12] L. De Lathauwer, "A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization," *SIAM Journal on Matrix Analysis and Applications*, vol. 28, no. 3, pp. 642–666, 2006.

[13] C. J. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural computation*, vol. 19, no. 10, pp. 2756–2779, 2007.

[14] P. H. Calamai and J. J. Moré, "Projected gradient methods for linearly constrained problems," *Mathematical programming*, vol. 39, no. 1, pp. 93–116, 1987.

[15] D. P. Bertsekas, "Nonlinear programming," *Athena scientific*, 1999.

[16] B. W. Bader, T. G. Kolda, et al., "Matlab tensor toolbox version 2.6," Available online, February 2015.

[17] C. A. Andersson and R. Bro, "The n-way toolbox for MATLAB," *Chemometrics and Intelligent Laboratory Systems*, vol. 52, no. 1, pp. 1–4, 2000.

[18] Y. Xu and W. Yin, "A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion," *SIAM Journal on imaging sciences*, vol. 6, no. 3, pp. 1758–1789, 2013.