SFBS: a Forward-Backward Splitting algorithm for constrained tensor decomposition

Elaheh Sobhani, Pierre Comon, Christian Jutten, Massoud Babaie-Zadeh

SFBS: A Forward-Backward Splitting algorithm for constrained tensor decomposition

Elaheh Sobhani[a,b,1,*], Pierre Comon[a], Christian Jutten[a], Massoud Babaie-Zadeh[b]

[a] *GIPSA-Lab, UMR CNRS 5216, Univ. Grenoble Alpes, CNRS, Grenoble INP, GIPSA-lab, 38000 Grenoble, France*
[b] *Department of Electrical Engineering, Sharif University of Technology, Tehran 11365-11155, Iran*

Abstract

Tensors (multi-way arrays) and *constrained* tensor decomposition are very practical in various applications. The existing constrained decomposition algorithms, which are based on Alternating Direction Method of Multipliers (ADMM) or proximal methods, suffer either from a lack of complete convergence guarantee or from the lack of expected accuracy. In this paper, we propose a constrained decomposition algorithm, called SFBS, which stands for "**S**imple **F**orward-**B**ackward **S**plitting" and is based on a particular proximal method to handle constraints. SFBS is theoretically and practically ahead compared to the state-of-the-art, since (i) not only SFBS achieves state-of-the-art performances, but also has a complete convergence guarantee, unlike ADMM-based algorithms; (ii) SFBS is much more robust against additive noise and computationally less expensive; (iii) unlike some existing algorithms, SFBS requires to adjust fewer hyperparameters, which are easy to set according to the convergence condition.

Keywords: Tensor, constraint, Proximal, ADMM, Simplex, Non-negative.

*Corresponding author
Email addresses: sobhani.es@gmail.com (Elaheh Sobhani), pierre.comon@gipsa-lab.grenoble-inp.fr (Pierre Comon), christian.jutten@gipsa-lab.grenoble-inp.fr (Christian Jutten), mbzadeh@sharif.edu (Massoud Babaie-Zadeh)

[1] Portions of this research were done while the author was a dual degree Ph.D. student at GIPSA-Lab and Sharif University of Technology

1. Introduction

Considering vectors and matrices as one-way and two-way arrays, respectively, *tensors* may be viewed as multi-way arrays, which can have more than two dimensions. In order to recover latent variables by means of tensors, data tensors are typically decomposed into some multi-way arrays (vectors, matrices or tensors), which can be interpreted as the desired latent variables of the problem. Canonical Polyadic (CP) decomposition [1], also referred to as CANDECOMP or PARAFAC [2] in some communities, is one of the prevalent tensor decompositions in the literature.

Tensors have appeared in many applications so far, such as Blind Source Separation (BSS) based on data cumulants [2], many tasks of machine learning including classification [3], data fusion[4], topic modeling and the estimation of parameters in mixture models [5, 6], to cite a few. According to the application, it is preferred to add some constraints to the tensor decomposition that results normally in much more accurate and reasonable solutions. Non-negativity, belonging to simplex set, orthogonality and sparsity are some examples of key constraints often introduced in applications such as medical image and signal processing [7], probability estimation in topic modeling [6] and dictionary learning [8].

Some of tensor decomposition algorithms are generalizations of matrix decompositions in the literature, for example Non-negative Tensor Factorization (NTF) [7] is inspired from Non-negative Matrix Factorization (NMF), or the tensor Power method [5] is dedicated to symmetric tensors and mimics the matrix Power method aiming at calculating matrix eigenpairs. We will show experimentally that the tensor Power method is not robust against additive noise, and is not reliable because it does not take properly into account the constraints.

On the other hand, some of the existing constrained tensor decomposition algorithms are based on Alternating Optimization (AO) [3] or its special case, Alternating Least Squares (ALS) [4], in which the data fidelity term is the least square error. Although, an unconstrained tensor decomposition can be

computed with ALS as the Nway software [9] does, this approach is very sensitive to additive noise [3, Section VII-G]. Therefore, the implementation of constrained algorithms is preferred to unconstrained ones; examples include Alternating Optimization-Alternating Direction Method of Multipliers (AO-ADMM) [10] (employing ADMM in each step of AO), Alternating Proximal Gradient (APG) [11], FastNTF-APG [12], or BC-VMFB [13] (employing a proximal approach in each step of AO).

The convergence of AO in AO-ADMM is not guaranteed because the proximal regularization is ignored [10], and also the convergence of ADMM for the non-convex constraint ℓ_0 has not yet been proved [14]. Although it is proved that the algorithms based on proximal method [11, 12, 13] such as APG, FastNTF-APG and BC-VMFB converge to a critical point of the problem even for non-convex constraints, the performances of these kinds of algorithms have not yet achieved that of AO-ADMM. Moreover, some methods such as BC-VMFB [13] force the user to adjust several parameters having a strong impact on the result. Some other methods [15] in the same vein are customized for large dimension tensor decompositions, a subject out of the scope of the present contribution.

In this paper, an algorithm for constrained tensor decomposition based on ALS is proposed, in each step of which, a constrained minimization problem is solved by means of a specific proximal approach called *Forward-Backward Splitting* [16, 17]. We call this algorithm SFBS, which stands for "**S**imple **F**orward-**B**ackward **S**plitting", as it is easier to understand and implement comparing to state-of-the-art algorithms. The practical and theoretical contributions of this paper are as follows: firstly, not only SFBS performs either better than or as well as AO-ADMM [10], but also a complete convergence analysis exists for SFBS that can be applied even on non-convex and non-smooth constraints such as cardinality [18], which is not the case for AO-ADMM. Secondly, SFBS performs better than APG [11] in noisy cases and is computationally less expensive, because the computation of a large number of coefficients is avoided, unlike APG. Thirdly, contrary to BC-VMFB [13], SFBS does not require many critical settings of the parameters but just one, which is easy to set according to the

SFBS convergence condition. Fourthly, in spite of BC-VMFB, SFBS works with variables in matrix form, hence, there is no need to vectorize loading matrices, which brings an advantage in working with large dimension tensors. Fifthly, compared with unconstrained algorithms such as the tensor Power method [5], not only the results of SFBS are much more reliable due to employing proper constraints such as the membership to the simplex set, but also it benefits from theoretical convergence guarantees. Further, SFBS is much more robust against additive noise and is not dedicated to symmetric tensors, unlike the tensor Power method.

The paper is organized as follows. In Section 2, some preliminaries about tensors, CP decomposition and proximal methods are reviewed. SFBS and its convergence analysis are described in Section 3 and Appendix B, respectively. Finally, Section 4 reports computer results.

Notation. Vectors, matrices and tensors are denoted with bold lowercases (*e.g.*, $\boldsymbol{a}$), bold uppercases (*e.g.*, $\boldsymbol{A}$) and bold calligraphic letters (*e.g.*, $\boldsymbol{\mathcal{T}}$), respectively. The tensor (outer) product is denoted by $\otimes$.

2. Preliminaries

2.1. Tensors and CP decomposition

Tensors can be considered as a multi-linear map from a vector space to another one [2], or simply as multi-way (multi-dimensional or multi-index) numerical arrays [7]. The order of an array refers to the number of its ways [19]. Vectors and matrices are one-way and two-way arrays, respectively, but usually a tensor is associated with an array with three or more ways [3].

A decomposable tensor of order N is a tensor product of N vectors [2], *i.e.*, $\boldsymbol{\mathcal{D}} = \boldsymbol{a}^{(1)} \otimes \boldsymbol{a}^{(2)} \ldots \otimes \boldsymbol{a}^{(N)}$. According to [1], any tensor can be written as a linear combination of a finite number of decomposable tensors,

$$\boldsymbol{\mathcal{T}} = \sum_{r=1}^{R} \lambda_r \boldsymbol{\mathcal{D}}(r), \tag{1}$$

where $\boldsymbol{\mathcal{T}}$ is a tensor of order N, and $\boldsymbol{\mathcal{D}}(r) = \boldsymbol{a}_r^{(1)} \otimes \boldsymbol{a}_r^{(2)} \ldots \otimes \boldsymbol{a}_r^{(N)}$. In compact form, (1) can be represented as $\boldsymbol{\mathcal{T}} = [\![\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}]\!]$, in which $\boldsymbol{\lambda}$ is a *coefficient vector* of size R containing the values of λ_r and $\boldsymbol{a}_r^{(1)}, \boldsymbol{a}_r^{(2)}, \ldots \boldsymbol{a}_r^{(N)}$ are the r^{th} columns of N *loading matrices* $\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}$, respectively. The N matrices $\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}$ are called mode-1, mode-2, ..., mode-N loading matrices, respectively, since their columns are responsible for the construction of the first, second, ..., N^{th} dimension of $\boldsymbol{\mathcal{T}}$ [4].

For the sake of convenience, tensors are sometimes transformed into matrices [2]. This transformation is called *unfolding* or *flattering* and can be done in each mode. The resulting matrix in mode n, $1 \leq n \leq N$, is called *mode-n unfolding* [2] and is denoted by $\boldsymbol{T}^{(n)}$.

For each tensor, the minimum value of R for which (1) holds is called tensor rank [2]. Therefore, the rank of a decomposable tensor is one. The decomposition described in (1) is called *Polyadic decomposition* [1] or also *CANDECOMP* or *PARAFAC* [2]. Based on the number of unknowns in (1), the expected rank of a tensor is defined as $R^{\circ} \triangleq \left\lceil \dfrac{D}{1 - N + \sum_i I_i} \right\rceil$, where N is the order of the tensor of dimensions $I_1 \times \ldots \times I_N$ and $D = \prod_i I_i$. It has been shown in [20] that $R \leq R^{\circ} - 1$ ensures almost surely the uniqueness of decomposition (1). If this polyadic decomposition is unique, it can be called *Canonical Polyadic (CP)* decomposition [2].

As in practice data stored in the form of a tensor are usually corrupted by noise, the best rank-R approximation must be estimated. Although low-rank approximation is useful, generally it is ill-posed, since the set of tensors of rank at most R is not closed [2]. Therefore, imposing some constraints (such as non-negativity or the membership to the simplex set[2]) on loading matrices and coefficient vectors is proposed in the literature to overcome this difficulty [4].

[2] A simplex set is defined by: $\mathcal{S} \triangleq \{\boldsymbol{x} : \boldsymbol{x} \geq 0, \|\boldsymbol{x}\|_1 = 1\}$, where the inequality is to be understood entry-wise. This constraint corresponds to a sum-to-one also called column-stochasticity. This is the case when the entries of the vector $\boldsymbol{x}$ are probabilities that sum to one.

2.2. Proximity operator

The projection of a vector $\boldsymbol{x} \in \mathbb{R}^N$ onto a closed convex set $\mathcal{S} \subset \mathbb{R}^N$ is a classical problem in signal processing and can be formulated as $\text{proj}_{\mathcal{S}}(\boldsymbol{x}) = \text{argmin}_{\boldsymbol{y}\in\mathbb{R}^N} \left\{ i_{\mathcal{S}}(\boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2 \right\}$ [21], where $i_{\mathcal{S}}$ is the indicator function defined by:

$$i_{\mathcal{S}}(\boldsymbol{y}) \triangleq \begin{cases} 0 & \text{if} \quad \boldsymbol{y} \in \mathcal{S} \\ \infty & \text{if} \quad \boldsymbol{y} \notin \mathcal{S} \end{cases} \tag{2}$$

Let $\Gamma_0(\mathbb{R}^N)$ be the class of lower semi-continuous functions $f : \mathbb{R}^N \mapsto (-\infty, +\infty]$, with $\text{dom}(f) \neq \varnothing$[3]. Then $i_{\mathcal{S}}$ belongs to $\Gamma_0(\mathbb{R}^N)$.

The definition of *Proximity operator* is obtained by replacing $i_{\mathcal{S}}(\boldsymbol{y})$ in projection minimization with any arbitrary function in $\Gamma_0(\mathbb{R}^N)$ [23]:

Definition 1 (Proximity operator [16])**.** *For every $\boldsymbol{x} \in \mathbb{R}^N$, the unique solution of* $\text{argmin}_{\boldsymbol{y}\in\mathbb{R}^N} f(\boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2$ *is defined as the proximity operator of the function $f \in \Gamma_0(\mathbb{R}^N)$, and it is denoted by $prox_f(\boldsymbol{x})$. The term $\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2$ is also called* proximal regularization *in the literature [11]. Thus, the proximity operator of f is $prox_f : \mathbb{R}^N \mapsto \mathbb{R}^N$, and it is characterized by:*

$$\boldsymbol{p} = prox_f(\boldsymbol{x}) \Leftrightarrow (\boldsymbol{x}-\boldsymbol{p}) \in \partial f(\boldsymbol{p}), \quad \forall (\boldsymbol{x}, \boldsymbol{p}) \in \mathbb{R}^N \times \mathbb{R}^N,$$

where $\partial f(\cdot)$ is the subgradient[4] of f. Note that $\partial f(\boldsymbol{p})$ is replaced by $\nabla f(\boldsymbol{p})$ for differentiable f.

The above definition indicates that $\text{prox}_f(\boldsymbol{x})$ is a point that minimizes f and simultaneously is as close as possible to $\boldsymbol{x}$. Therefore, $\text{prox}_f(\boldsymbol{x})$ is also called a *proximal point of $\boldsymbol{x}$ with respect to f* [17]. See Table 10.2 in [16] for a list of popular functions and their corresponding proximity operators.

Remark 1. ***Kurdyka-Lojasiewicz property (KL) [18]:*** *The function $f : \mathbb{R}^n \mapsto \mathbb{R} \cup \{+\infty\}$ has the Kurdyka-Lojasiewicz (KL) property at $\boldsymbol{x}^* \in$*

[3] $\text{dom}(f)$ denotes the domain of a function f, *i.e.*, the set of feasible solutions if $f(x)$ is to be minimized [22].

[4] If $f(\boldsymbol{x}) > f(\overline{\boldsymbol{x}}) + \langle \boldsymbol{v}, \boldsymbol{x}-\overline{\boldsymbol{x}} \rangle + o(|\boldsymbol{x}-\overline{\boldsymbol{x}}|)$, then $\boldsymbol{v}$ is a subgradient of f at $\overline{\boldsymbol{x}}$, *i.e.*, $\boldsymbol{v} \in \partial f$ [16].

dom(f) *if there exist* $\eta \in (0, +\infty]$, *a neighborhood* U *of* $\boldsymbol{x}^*$ *and a continuous concave function* $\varphi : [0, \eta) \mapsto \mathbb{R}_+$ *such that (i)* $\varphi(0) = 0$, *(ii)* φ *is differentiable on* $(0, \eta)$, *(iii)* $\varphi'(y) \geq 0$ *for all* $y \in (0, \eta)$, *(iv)* $\varphi'(f(\boldsymbol{x}) - f(\boldsymbol{x}^*))dist(0, \partial f(\boldsymbol{x})) \geq 1$ *(the Kurdyka-Lojasiewicz inequality) holds for all* $\boldsymbol{x} \in U \cap \{\boldsymbol{x} | f(\boldsymbol{x}^*) < f(\boldsymbol{x}) < f(\boldsymbol{x}^*) + \eta\}$, *where dist*(.) *denotes the distance function. In the rest of this paper, this property is referred to as "KL", in short. Many functions encountered in finite-dimensional applications and in particular many convex functions have KL property [18], while it is not trivial to check the conditions in the KL definition [11]. Some examples of functions with KL property are mentioned in Appendix B.1.*

2.3. Forward-Backward Splitting

By following a proximal approach, an optimization containing a non-differentiable and/or non-convex function can be solved by means of a particular algorithm called *Forward-Backward Splitting* as explained in Theorem 1, which is mentioned in [16].

Theorem 1 (Forward-Backward Splitting [16])**.** *Suppose* $f : \mathbb{R}^N \mapsto \mathbb{R} \cup \{+\infty\}$ *is a proper*[5] *lower semi-continuous function, which has the KL property and is bounded from below. If* f *can be split into two parts as* $f = h + g$, *where* g *is lower semi-continuous and* $h : \mathbb{R}^N \mapsto \mathbb{R}$ *is a finite-valued, differentiable function with a* β*-Lipschitz continuous gradient,* i.e., $\exists \beta$ *such that* $\|\nabla h(\boldsymbol{x}) - \nabla h(\boldsymbol{y})\|_2 \leq \beta \|\boldsymbol{x} - \boldsymbol{y}\|_2$, *then it can be shown [25] that the minimizer of* f *satisfies the following fixed point equation:*

$$\boldsymbol{x} = prox_{\gamma g}(\boldsymbol{x} - \gamma \nabla h(\boldsymbol{x})), \tag{3}$$

where $\gamma \in (0, +\infty)$.

Equation (3) suggests an iterative approach, called the *Forward-Backward*

[5] A function $f : \mathbb{R}^N \mapsto [-\infty, +\infty]$ has effective domain $\mathrm{dom}(f) = \{\nu \,|\, f(\nu) < \infty\}$, and is called proper if the set $\mathrm{dom}(f)$ is non empty [24, Chapter 1].

Algorithm 1 Forward-Backward Splitting [25, 16, Algorithm 10.5]

Input: The function $f = h + g$ as defined in Theorem 1, β, $\boldsymbol{x}_0 \in \mathbb{R}^N$

Output: The minimizer of f

1: Fix $\epsilon_0 \in (0, \min\{1, \frac{1}{\beta}\})$
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: $\quad \gamma_k \in [\epsilon_0, \frac{2}{\beta} - \epsilon_0]$
4: $\quad \boldsymbol{y}_k = \boldsymbol{x}_k - \gamma_k \nabla h(\boldsymbol{x}_k)$
5: $\quad \alpha_k \in [\epsilon_0, 1]$
6: $\quad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k(\mathrm{prox}_{\gamma_k g}(\boldsymbol{y}_k) - \boldsymbol{x}_k)$
7: **end for**

Splitting algorithm [16]:

$$\boldsymbol{x}_{k+1} = \mathrm{prox}_{\gamma g}(\boldsymbol{x}_k - \gamma_k \nabla h(\boldsymbol{x}_k)), \tag{4}$$

where the values of γ_k should be chosen in a suitable bounded interval.

Several variations of implementing Forward-Backward Splitting exist and are reported in [16]. Two of them are restated here (Algorithm 1 and Algorithm 2) to which we will refer in the rest of this paper. Although Algorithm 2 is more user-friendly than Algorithm 1 in terms of the number of required parameters to be set, the computational complexity of Algorithm 1 is smaller than that of Algorithm 2 due to the fact that some coefficients, like t_k and λ_k in lines 5 and 6 of Algorithm 2, are not calculated.

In many applications (including our proposed method, SFBS, in Section 3), the function g is an indicator function of a particular set ($\mathcal{S}$), $i_{\mathcal{S}}$, and its proximity operator is a projection onto that set [16, Table 10.2]. If the desired set is non-convex, the projection onto it may not result in a unique point. It has been proved [18] that in spite of the multi-valued projection, the convergence property of Theorem 1 is not affected. Note that this interesting conclusion is valid only if the assumptions of Theorem 1 are satisfied, the most important being the KL property satisfied by $h + i_{\mathcal{S}}$.

Algorithm 2 Beck-Teboulle proximal gradient algorithm [16, Algorithm 10.7] based on FISTA [26]

Input: The function $f = h + g$ as defined in Theorem 1, β, $\boldsymbol{x}_0 \in \mathbb{R}^N$

Output: The minimizer of f

1: Set $\boldsymbol{z}_0 = \boldsymbol{x}_0$ and $t_0 = 1$
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: $\quad \boldsymbol{y}_k = \boldsymbol{z}_k - \beta^{-1}\nabla h(\boldsymbol{z}_k)$
4: $\quad \boldsymbol{x}_{k+1} = \text{prox}_{\beta^{-1}g}(\boldsymbol{y}_k)$
5: $\quad t_{k+1} = \dfrac{1 + \sqrt{4t_k^2 + 1}}{2}$
6: $\quad \lambda_k = 1 + \dfrac{t_k - 1}{t_{k+1}}$
7: $\quad \boldsymbol{z}_{k+1} = \boldsymbol{x}_k + \lambda_k(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)$
8: **end for**

3. Proposed: Simple Forward-Backward Splitting (SFBS)

Consider the N-th order tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots I_N}$ of rank R. Assume that $\boldsymbol{\mathcal{T}} = [\![\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}]\!]$, where $\boldsymbol{\lambda} \in \mathbb{R}^R$ and $\boldsymbol{A}^{(n)} \in \mathbb{R}^{I_n \times R}$. When all entries of an array $\boldsymbol{B}$ are constrained to belong to a set $\mathcal{E}$, the associated constraint and the corresponding indicator function are denoted by $\mathcal{C}_{\mathcal{E}}(\boldsymbol{B})$ and $i_{\mathcal{E}}(\boldsymbol{B})$, respectively. This notation may apply for instance to $\boldsymbol{\lambda}$ and to $\boldsymbol{A}^{(n)}$, and either to the non-negative orthant $\mathcal{N}$ or to the simplex set $\mathcal{S}$. A general problem of the constrained CP decomposition of $\boldsymbol{\mathcal{T}}$ can be formulated as follows:

$$\min_{\boldsymbol{\lambda}, \boldsymbol{A}^{(n)}} \frac{1}{2}\|\boldsymbol{\mathcal{T}} - [\![\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}]\!]\|_F^2 \tag{5}$$
$$\text{s.t. } \mathcal{C}_{\mathcal{E}}(\boldsymbol{\lambda}), \mathcal{C}_{\mathcal{E}}(\boldsymbol{A}^{(n)}),\ 1 \leq n \leq N,$$

where $\mathcal{C}_{\mathcal{E}}(\boldsymbol{\lambda}), \mathcal{C}_{\mathcal{E}}(\boldsymbol{A}^{(n)})$ are, respectively, the constraints on the vector $\boldsymbol{\lambda}$ and on matrices $\boldsymbol{A}^{(n)}$[6].

[6]Note that each array $(\boldsymbol{\lambda}, \boldsymbol{A}^{(n)},\ 1 \leq n \leq N)$ may have its own constraint, hence its corresponding set $\mathcal{E}$ may be different from other arrays.

Remark 2. *The considered constraints in this paper are the non-negativity* 160 *($\mathcal{E} = \mathcal{N}$) and the membership in the simplex set ($\mathcal{E} = \mathcal{S}$). To be more precise, for all arrays, i.e., λ and all factor matrices $A^{(n)}$, in the simulations of Section 4.1.1 (Fig. 1) and Section 4.1.2 (Fig. 2), $\mathcal{E}$ is $\mathcal{N}$ and $\mathcal{S}$, respectively. In addition, $\mathcal{E} = \mathcal{S}$ in the experiment of Section 4.2.*

A common strategy is to solve (5) via ALS and adding the indicator function of the constraint set to the cost function [10]. To be more precise, at the n^{th} step of ALS for solving (5), we have[7]:

$$\min_{\boldsymbol{A}^{(n)}} \frac{1}{2} \left\| \boldsymbol{\mathcal{T}} - [\![\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(n)}, \ldots, \boldsymbol{A}^{(N)}]\!] \right\|_F^2 + i_{\mathcal{E}}(\boldsymbol{A}^{(n)}), \tag{6}$$

where $i_{\mathcal{E}}(\boldsymbol{A}^{(n)})$ is an indicator function as defined in (2). Define $\boldsymbol{W} \triangleq (\boldsymbol{A}^{(N)} \odot \ldots \odot \boldsymbol{A}^{(n+1)} \odot \boldsymbol{A}^{(n-1)} \odot \ldots \odot \boldsymbol{A}^{(1)})^T$, where $\odot$ is the Khatri-Rao product. Then by the mode-n unfolding of (6), we have:

$$\min_{\boldsymbol{A}^{(n)}} \frac{1}{2} \|\boldsymbol{T}^{(n)} - \boldsymbol{A}^{(n)} \boldsymbol{W}\|_F^2 + i_{\mathcal{E}}(\boldsymbol{A}^{(n)}). \tag{7}$$

Observe that $i_{\mathcal{E}}(\boldsymbol{A}^{(n)})$ is lower semi-continuous [22, Definition 1.5] for $\mathcal{E} = \mathcal{N}$ 165 or $\mathcal{E} = \mathcal{S}$. Furthermore, $\frac{1}{2}\|\boldsymbol{T}^{(n)} - \boldsymbol{A}^{(n)} \boldsymbol{W}\|_F^2$ is finite-valued, differentiable and β-Lipschitz continuous gradient, where $\beta = \|\boldsymbol{W}\boldsymbol{W}^T\|_\sigma$ denotes the spectral norm[8] of the matrix $\boldsymbol{W}\boldsymbol{W}^T$ (calculated in Appendix A). Since $\boldsymbol{W}\boldsymbol{W}^T$ is a symmetric matrix, its singular values are the squared of those of $\boldsymbol{W}$. Moreover, the cost function in (7) is proper, lower semi-continuous with the KL 170 property [18] (proved in Appendix B). Consequently, based on Theorem 1, the minimizer of (7) is $\boldsymbol{A}^{(n)} = \text{prox}_{\gamma i_{\mathcal{E}}(\boldsymbol{A}^{(n)})}\{\boldsymbol{A}^{(n)} - \gamma(\boldsymbol{A}^{(n)}\boldsymbol{W}\boldsymbol{W}^T - \boldsymbol{W}\boldsymbol{T}^{(n)^T})\}$. Since the proximity operator of $\gamma i_{\mathcal{E}}(\boldsymbol{A}^{(n)})$ is the projection onto $\mathcal{C}_{\mathcal{E}}(\boldsymbol{A}^{(n)})$, we have $\boldsymbol{A}^{(n)} = \text{proj}_{\mathcal{C}_{\mathcal{E}}(\boldsymbol{A}^{(n)})}\{\boldsymbol{A}^{(n)} - \gamma(\boldsymbol{A}^{(n)}\boldsymbol{W}\boldsymbol{W}^T - \boldsymbol{W}\boldsymbol{T}^{(n)^T})\}$.

Putting all together, this results in our proposed algorithm, which we call 175 Simple Forward-Backward Splitting (SFBS). Since SFBS requires fewer param-

[7]The vector $\boldsymbol{\lambda}$ is omitted in (6), since it can be calculated by normalizing loading matrices $(\boldsymbol{A}^{(n)})$. Otherwise, the vector $\boldsymbol{\lambda}$ as one of the unknown variables can be optimized in one of the steps of ALS.

[8]The spectral norm of a matrix is equal to its maximum singular value.

Algorithm 3 Proposed: Simple Forward-Backward Splitting (SFBS)

Input: $\boldsymbol{\mathcal{T}}$, $\mathcal{C}_{\mathcal{E}}(\boldsymbol{A}^{(n)})$, initial $\boldsymbol{A}_0^{(n)}$, $n \in [1, \ldots, N]$, e

Output: Estimated $\boldsymbol{A}^{(n)}$, $n \in [1, \ldots, N]$

1: **repeat**
2: **for** $n = 1, 2, \ldots, N$ **do**
3: $\boldsymbol{W} = (\boldsymbol{A}^{(N)} \odot \ldots \odot \boldsymbol{A}^{(n+1)} \odot \boldsymbol{A}^{(n-1)} \odot \ldots \odot \boldsymbol{A}^{(1)})^T$
4: $\beta = \{\max(\text{singular value}(\boldsymbol{W}))\}^2$, set $\gamma = \frac{e}{\beta}$ and choose α_g.
5: **for** $g = 0, 1, 2, \ldots$ **do**
6: $\boldsymbol{Y} = \boldsymbol{A}_g^{(n)} - \gamma(\boldsymbol{A}_g^{(n)}(\boldsymbol{W}\boldsymbol{W}^T) - \boldsymbol{T}^{(n)}\boldsymbol{W}^T)$
7: $\boldsymbol{A}_{g+1}^{(n)} = \boldsymbol{A}_g^{(n)} + \alpha_g(\text{proj}_{\mathcal{C}_{\mathcal{E}}(\boldsymbol{A}^{(n)})}(\boldsymbol{Y}) - \boldsymbol{A}_g^{(n)})$
8: **end for**
9: **end for**
10: **until** some termination criterion

eters to be calculated[9] than the methods based on Algorithm 2 like APG, and is also easier to understand and implement, it is qualified as *Simple*. SFBS is described in Algorithm 3, which is a customized version of Algorithm 1 for constrained CP decomposition.

Remark 3. *The projection onto the non-negative orthant* $\mathcal{N}$ *is done with the* max *operator, thereby the line* 7 *of Algorithm 3 would be* $\boldsymbol{A}_{g+1}^{(n)} = \boldsymbol{A}_g^{(n)} + \alpha_g(\max(\boldsymbol{Y}, 0) - \boldsymbol{A}_g^{(n)})$.

Remark 4. *A detailed description of SFBS imposing joint constraints* $\mathcal{C}_{\mathbb{S}}(\boldsymbol{\lambda})$ *and* $\mathcal{C}_{\mathbb{S}}(\boldsymbol{A}^{(n)})$ *is provided in Algorithm 4, in line* 13 *and* 16 *of which the method proposed in [27] is employed to project a vector onto a simplex set. In Algorithm 4, the simplex constraint is applied to every column of every matrix* $\boldsymbol{A}^{(n)}, n \in [1, \ldots, N-1]$*, and only to the vectorization of matrix* $\boldsymbol{A}_{\boldsymbol{\lambda}}^{(N)} = \boldsymbol{A}^{(N)}\mathit{Diag}(\boldsymbol{\lambda})$*, where* $\mathit{Diag}(\boldsymbol{\lambda})$ *is a diagonal matrix containing* $\boldsymbol{\lambda}$ *on its diago-*

[9] Algorithm 1 has more parameters to be *set* by the user, however Algorithm 2 has more parameters to be *calculated* like t_k, λ_k, as mentioned here.

nal. It can then be proved easily that vector $\boldsymbol{\lambda}$ *and every column of loading matrices estimated by Algorithm 4 indeed lie in the simplex set.*

4. Simulation

In this section, SFBS is compared to AO-ADMM [10], APG [11], FastNTF-APG [12] and BC-VMFB [13][10] and *Nway*, a software for CP decomposition via least squares, from a well-known tensor toolbox [9]. Algorithms are either tested on artificially generated tensors or on an estimated tensor (third order moments) of a real text data set.

One should note some technical points about our practical implementations. First, in order to compute the Lipschitz constant (line 4 (resp. line 9) of Algorithm 3 (resp. Algorithm 4)), one can employ $[\texttt{norm(W)}]^2$ command in MATLAB, where $\boldsymbol{W}$ is the matrix composed of constant loading matrices and is defined in line 3 (resp. line 5 and 7) of Algorithm 3 (resp. Algorithm 4). Second, one could employ Corcondia [30] to obtain an estimation of the rank R, which is a required input for all algorithms considered in this section. However, in synthetic scenarios, the real rank is known. Working with real text data set, some points about rank selection are noted in Section 4.2. Lastly, we fix the value of $\gamma_k = \dfrac{e}{\beta}$ in Algorithm 3 and Algorithm 4 for all iterations over k with $e = [1.4, 1.9]$, which is experimentally observed to be proper. In addition, by setting α_g to 1, we remove the effect of linear combination of obtained points in two successive iterations (See line 7 of Algorithm 3).

In order to fairly compare the performances of algorithms, all the algorithms are required to iterate until either they reach a maximum predefined number of iterations (denoted by *iterations max-number*) or the variation of *relative*

[10] We would like to thank the corresponding authors of FastNTF-APG [12] and BC-VMFB [13], Guoxu Zhou and Caroline Chaux, respectively, who sent us the MATLAB codes of their methods. The MATLAB codes of AO-ADMM [10] and APG [11] are made available by the authors at [28] and [29], respectively. Therefore, the original codes of authors have been used to obtain the results reported in all figures of this section.

Algorithm 4 SFBS for simplex constraint

Input: $\mathcal{T}$, $\mathcal{C}_{\mathbb{S}}(\boldsymbol{A}^{(n)})$, initial $\boldsymbol{A}_0^{(n)},\ n \in [1,\ldots,N]$, $\boldsymbol{\lambda}_0^R$, e

Output: Estimated $\boldsymbol{A}^{(n)},\ n \in [1,\ldots,N]$, $\boldsymbol{\lambda}$

$\boldsymbol{A}_{\boldsymbol{\lambda},0}^{(N)} = \boldsymbol{A}_0^{(N)}\text{Diag}(\boldsymbol{\lambda}_0)$, where $\text{Diag}(\boldsymbol{\lambda}_0)$ is a diagonal matrix containing $\boldsymbol{\lambda}_0$ on its diagonal.

repeat

  for $n = 1, 2, \ldots, N$ **do**

    if $n < N$ **then**

      $\boldsymbol{W} = (\boldsymbol{A}_{\boldsymbol{\lambda}}^{(N)} \odot \ldots \odot \boldsymbol{A}^{(n+1)} \odot \boldsymbol{A}^{(n-1)} \odot \ldots \odot \boldsymbol{A}^{(1)})^T$

    else

      $\boldsymbol{W} = (\boldsymbol{A}^{(N-1)} \odot \ldots \odot \boldsymbol{A}^{(n+1)} \odot \boldsymbol{A}^{(n-1)} \odot \ldots \odot \boldsymbol{A}^{(1)})^T$

    end if

    $\beta = \{\max(\text{singular value}(\boldsymbol{W}))\}^2$ and set $\gamma = \frac{e}{\beta}$.

    for $g = 0, 1, 2, \ldots$ **do**

      if $n < N$ **then**

        $\boldsymbol{Y} = \boldsymbol{A}_g^{(n)} - \gamma(\boldsymbol{A}_g^{(n)}(\boldsymbol{W}\boldsymbol{W}^T) - \boldsymbol{T}^{(n)}\boldsymbol{W}^T)$

        $\boldsymbol{A}_{g+1}^{(n)}(:, r) = \text{proj}_{\mathcal{C}_{\mathbb{S}}}(\boldsymbol{Y}(:, r)),\ 1 \le r \le R$

      else

        $\boldsymbol{Y} = \boldsymbol{A}_{\boldsymbol{\lambda},g}^{(N)} - \gamma(\boldsymbol{A}_{\boldsymbol{\lambda},g}^{(N)}(\boldsymbol{W}\boldsymbol{W}^T) - \boldsymbol{T}^{(n)}\boldsymbol{W}^T)$

        $\boldsymbol{A}_{\boldsymbol{\lambda},g+1}^{(N)} = \text{vec}^{-1}(\text{proj}_{\mathcal{C}_{\mathbb{S}}}(\text{vec}(\boldsymbol{Y})))$, where vec is the vectorization operator

      end if

    end for

  end for

  $\lambda_r = \|\boldsymbol{A}_{\boldsymbol{\lambda}}^{(N)}(:, r)\|_1,\ \boldsymbol{A}^{(N)}(:, r) = \dfrac{\boldsymbol{A}_{\boldsymbol{\lambda}}^{(N)}(:, r)}{\lambda_r},\ 1 \le r \le R$

until some termination criterion

objective value between two successive iterations is less than a desired small value, namely ζ. If the objective value in iteration k is denoted by $\Psi(k)$, which is the difference between the input tensor and the estimated tensor, the criterion to stop iterations is $\Delta\Psi(k) = \left|\frac{\Psi(k) - \Psi(k-1)}{\Psi(k)}\right| \leq \zeta$. In addition, the reported result of each experiment is averaged over several realizations of the tensors, and for each realization, all the methods are initialized by an identical set of initializations, which are generated randomly, to choose the best initialization point. Therefore, it is possible that the best result for each method is not achieved for the same initialization point, but nonetheless, the comparison is fair, since the set of initializations is the same for all the considered methods. In the sequel, the number of realizations of tensors and the number of initialization points are mentioned by *average number* and *initialization number*, respectively.

All computer experiments reported in this section have been executed on a laptop with a 3.1 GHz Intel Core i5 processor, 16 GB of RAM running macOS Mojave and MATLAB 2019a.

4.1. Synthetic data

According to the considered constraints, loading matrices and a coefficient vector are generated randomly (uniform distribution in the interval $[0, 1]$). Then, the noiseless tensor $\boldsymbol{\mathcal{T}}_o$ is computed via (1). In order to work in a noisy context, a noise tensor, $\boldsymbol{\mathcal{T}}_n$, with i.i.d. entries of Gaussian distribution with zero mean and unit variance, of the same size as $\boldsymbol{\mathcal{T}}_o$, is weighted by the parameter σ and added to $\boldsymbol{\mathcal{T}}_o$. As $\boldsymbol{\mathcal{T}}_n$ has unit variance[11], then the variance of $\sigma\boldsymbol{\mathcal{T}}_n$ is σ^2. One can adjust σ such that a desired Signal to Noise Ratio (SNR) is reached according to the relation:

$$\mathrm{SNR} = 10\log_{10}\frac{\frac{1}{M}\sum_{i,j,k}\boldsymbol{\mathcal{T}}_o(i,j,k)^2}{\frac{1}{M}\sum_{i,j,k}\sigma^2\boldsymbol{\mathcal{T}}_n(i,j,k)^2},$$

where M is the total number of elements in tensors $\boldsymbol{\mathcal{T}}_o$ or $\boldsymbol{\mathcal{T}}_n$.

Denote by $\boldsymbol{\mathcal{T}}$ the tensor to be decomposed, $\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{T}}_o + \sigma\boldsymbol{\mathcal{T}}_n$. After decomposing $\boldsymbol{\mathcal{T}}$, the estimation of $\boldsymbol{\mathcal{T}}_o$ can be calculated using (1), as a rank-R ap-

[11] The variance of a multi-way array is defined as the sum of the variances of its entries.

proximation, which we call $\hat{\boldsymbol{\mathcal{T}}}$. The relative reconstruction error is computed as $\epsilon(\hat{\boldsymbol{\mathcal{T}}}) = \frac{\|\hat{\boldsymbol{\mathcal{T}}} - \boldsymbol{\mathcal{T}}_o\|_F^2}{\|\boldsymbol{\mathcal{T}}_o\|_F^2}$, which will be reported as it is, not in the form of percentage[12]. In addition, the error of estimating loading factors is reported according to Remark 5. Different simulations of this section may differ from each other in terms of SNR or constraints imposed on the arrays. However, in all simulations, $\zeta = 10^{-8}$ and the loop of line 5 (resp. line 10) is repeated five times for each mode in Algorithm 3 (resp. Algorithm 4).

Remark 5. *It is hard to assess the relative error made on loading matrices, because of the scaling and the permutation ambiguities of tensor decomposition [2]. So as to overcome these ambiguities, exact errors using,* e.g.*, Hungarian [31] is reported. However, if the error on* each *loading factor is reported separately, it would be an optimistic measure, since implicitly a specific (not common) permutation is permitted for each loading matrix. In order to have a more reliable performance index, the error based on the matrix $\boldsymbol{X}$ is reported, which consists of* all *loading factors together. In other words, the difference between $\boldsymbol{X}$ and $\widehat{\boldsymbol{X}}$ is reported, where $\boldsymbol{X}^T = \left[\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}\right]^T$, $\widehat{\boldsymbol{X}}^T = \left[\widehat{\boldsymbol{A}^{(1)}}, \widehat{\boldsymbol{A}^{(2)}}, \ldots, \widehat{\boldsymbol{A}^{(N)}}\right]^T$ and $\widehat{\boldsymbol{A}^{(n)}}$ is the estimation of $\boldsymbol{A}^{(n)}$ with the scaling and the permutation ambiguities. Nevertheless, there are other ways to report the performance such as CorrIndex measure described in [32], which computationally costs less than Hungarian in case of large size tensors.*

4.1.1. Non-negativity constraint

Figure 1.a shows ϵ in decomposing a noiseless tensor of size $10 \times 10 \times 10$, of rank $R = 6$ with initialization number, average number, iterations max-number and the parameter in line 4 of Algorithm 3, e, equal to 10, 10, 5000 and 1.9, respectively (cf. see Section 4.1 for the definition of these parameters). BC-VMFB results in a relative error around 1, which means 100%. Although we consulted the corresponding author of BC-VMFB, we found it hard to adjust

[12] For example, if we report $\epsilon(\hat{\boldsymbol{\mathcal{T}}}) = 1.5$, it should not be interpreted as 1.5%, but as 150%.

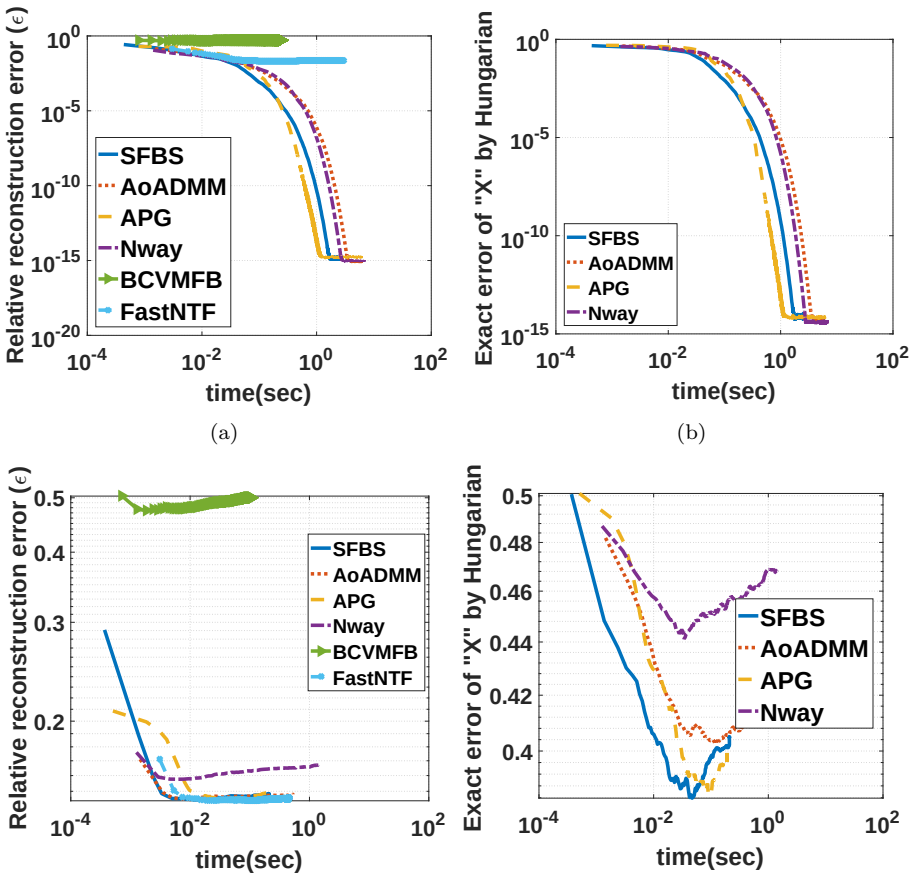

Figure 1: The performance of decomposing a tensor of dimension $10 \times 10 \times 10$, of rank $R = 6$ under the non-negativity constraint over all loading factors. For noiseless data, ζ, average number, initialization number, iterations max-number and the parameter in line 4 of Algorithm 3, e, are set to 10^{-8}, 10, 10, 5000 and 1.9, respectively. (a): The relative reconstruction error (ϵ) with noiseless data. (b): Comparison of the estimation of matrix $\boldsymbol{X}$ in the same experiment as Fig. 1.a.

For noisy data, SNR, ζ, average number, initialization number, iterations max-number and the parameter in line 4 of Algorithm 3, e, are set to 10 dB, 10^{-8}, 200, 20, 1000 and 1.9, respectively. (c): The relative reconstruction error (ϵ) with noisy data. (d): Comparison of the estimation of matrix $\boldsymbol{X}$ in the same experiment as Fig. 1.c.

several parameters of BC-VMFB. In addition, FastNTF-APG performs worse than APG, since it tries to decompose a low-rank approximation of the desired tensor. Although most of the methods achieve reasonable performances, APG and SFBS converge rather faster. In Fig. 1.b, the gap between $\boldsymbol{X}$ and $\widehat{\boldsymbol{X}}$ via Hungarian are reported[13] for the same experiment of Fig. 1.a, which can be interpreted in the same way as Fig. 1.a.

Figure 1.c indicates ϵ of a similar tensor as in Fig. 1.a in a noisy case with SNR, average number, initialization number, iterations max-number and the parameter in line 4 of Algorithm 3, e, equal to 10 dB, 200, 20, 1000 and 1.9, respectively. Although Nway carries out well in a noiseless case, its performance is not acceptable in the noisy situation. The result of FastNTF-APG is better than APG in a noisy scenario, as it replaces the noisy tensor with its low rank approximation, which helps to filter the noise out. Moreover, SFBS and AO-ADMM outperform other methods, therefore, the only algorithm that performs properly and better than others in both noiseless and noisy situations is SFBS. However, the average number of performed iterations for SFBS in this simulation is 550, while this value for AO-ADMM is equal to iterations max-number = 1000, which shows that SFBS converges faster.

In Fig. 1.d, the gap between $\boldsymbol{X}$ and $\widehat{\boldsymbol{X}}$ via Hungarian is reported. Although according to ϵ in Fig. 1.c, both AO-ADMM and SFBS outperform the others, in estimating loading factors, SFBS performs better than AO-ADMM. In spite of objective value, the error of estimating loading matrices can be ascending as it happens in Fig. 1.d, because the algorithms stop iterations based on the reconstruction error values, not on the loading matrix error values.

4.1.2. Simplex constraint

In this section, SFBS is only compared with AO-ADMM under the simplex constraint based on the Remark 4, since firstly, the implementation is directly investigated by the authors of AO-ADMM in [10]. Secondly, according to the

[13]The relative error of all the other loading factors shows almost the same results.

results of Section 4.1.1, AO-ADMM has the closest performance to that of SFBS.

Figure 2.a shows ϵ of the noiseless tensor of size $10 \times 10 \times 10$, of rank $R = 3$ with initialization number, average number, iterations max-number and the parameter in line 9 of Algorithm 4, e, equal to 10, 10, 20000 and 1.5, respectively, under the membership to the simplex set constraint. In addition, Fig. 2.b reportes the gap between $\boldsymbol{X}$ and $\widehat{\boldsymbol{X}}$ via Hungarian. By comparing Fig. 2.a and Fig. 1.a, it can be inferred that the membership to the simplex set is a more difficult constraint than non-negativity to be achieved, since the maximum required iterations for the simplex set is more than what is set for the non-negativity. As it can be seen in Fig. 2.a and Fig. 2.b, both SFBS and AO-ADMM achieve the same level of relative error, however SFBS converges slightly faster. To be more precise, the average number of performed iterations for SFBS in this simulation is 323, while this value for AO-ADMM is 798, which again shows that SFBS converges faster.

Figure 2.c indicates ϵ of a similar tensor as in Fig. 2.c in a noisy case with SNR, initialization number, average number, the parameter in line 9 of Algorithm 4, e, and iterations max-number equal to 10 dB, 20, 200, 1.9 and 1000, respectively. Figure 2.d represents the corresponding gap between $\boldsymbol{X}$ and $\widehat{\boldsymbol{X}}$ via Hungarian, and shows that SFBS performs better than AO-ADMM and converges slightly faster.

4.2. Real data

In this section, our experiments on a part of a well-known text data set, namely 20 Newsgroups, which consists of 11314 posts on 20 topics available online [33], are described. Let L be the number of words in a given document, $\boldsymbol{x}_\ell$ the observed words, $\ell \in \mathcal{L} = \{1, 2, \ldots, L\}$, and h, the topic of this document, encoded into a discrete variable taking K possible integer values, say from $\mathcal{H} = \{1, 2, \ldots, K\}$ with probability $\varphi(k) = \text{Prob}(h = k)$. All words belong to a known encoded dictionary $\Omega = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_D\}$ of cardinality D. In other words, one can consider a mapping γ_e (generally not injective) from $\mathcal{L}$ into $\{1, 2, \ldots, D\}$ such that $\boldsymbol{x}_\ell = \boldsymbol{u}_{\gamma_e(\ell)}$. In the context of text mining, D is the

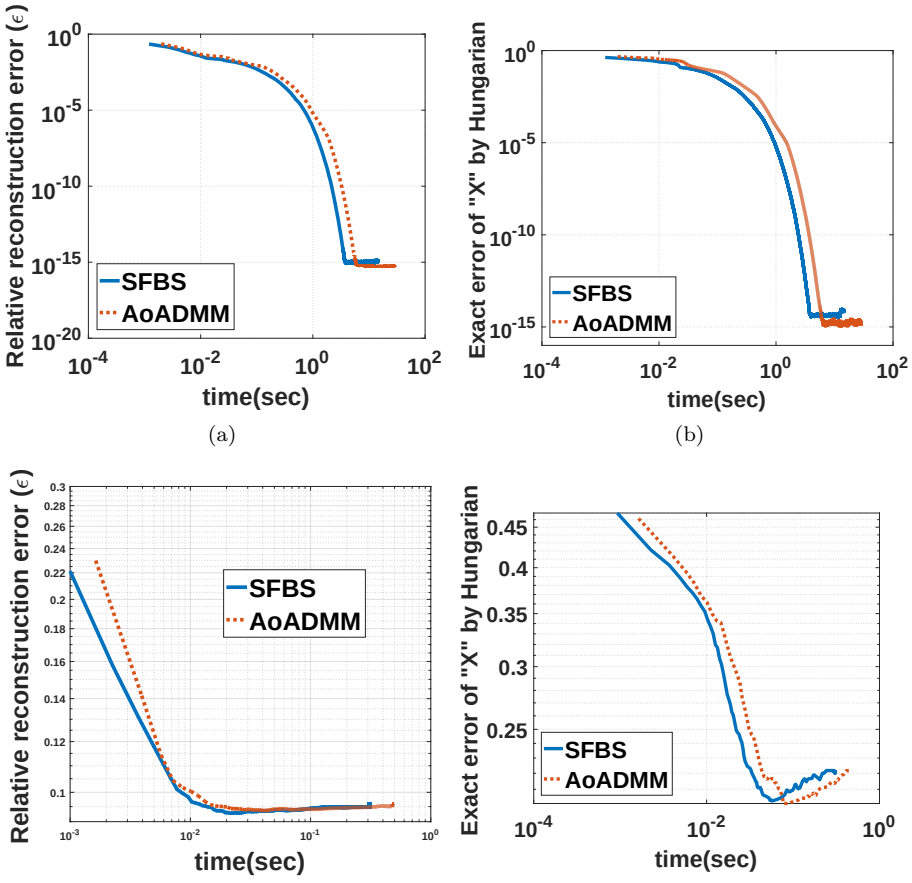

Figure 2: The performance of decomposing a tensor of dimension $10 \times 10 \times 10$, of rank $R = 3$ under the membership to the simplex set constraint over all loading factors and coefficient vector. For noiseless data, ζ, average number, initialization number, iterations max-number and the parameter in line 9 of Algorithm 4, e, are set to 10^{-8}, 10, 10, 20000 and 1.5, respectively. (a): The relative reconstruction error (ϵ) with noiseless data. (b): Comparison of the estimation of matrix $\boldsymbol{X}$ in the same experiment as Fig. 2.a.

For noisy data, SNR, ζ, average number, initialization number, iterations max-number and the parameter in line 9 of Algorithm 4, e, are set to 10 dB, 10^{-8}, 200, 20, 1000 and 1.5, respectively. (c): The relative reconstruction error (ϵ) with noisy data. (d): Comparison of the estimation of matrix $\boldsymbol{X}$ in the same experiment as Fig. 2.c.

number of words and K the number of topics. The conditional probability of each word $\boldsymbol{u}_d$ of the dictionary Ω, given a particular topic, $h = k$, is denoted by $f_k(d) = \text{Prob}(\boldsymbol{x} = \boldsymbol{u}_d | h = k)$.

In the sequel, the second order moment, $\boldsymbol{P}_m \overset{\text{def}}{=} \mathbb{E}_{\boldsymbol{x}}\{\boldsymbol{x}_p \otimes \boldsymbol{x}_q\}$, and third order moments, $\boldsymbol{\mathcal{T}}_m \overset{\text{def}}{=} \mathbb{E}_{\boldsymbol{x}}\{\boldsymbol{x}_p \otimes \boldsymbol{x}_q \otimes \boldsymbol{x}_r\}$, will be needed, where $\boldsymbol{P}_m$ is a $D \times D$ symmetric matrix and $\boldsymbol{\mathcal{T}}_m$ is a $D \times D \times D$ symmetric tensor. $\boldsymbol{x}_\ell$ is encoded to $\boldsymbol{u}_d$, and as in [5], $\boldsymbol{u}_d$ is chosen as the columns of the $D \times D$ identity matrix. Because of this choice for $\boldsymbol{u}_d$ and some other simplifying assumptions mentioned in [5], these moments exhibit the relations $\boldsymbol{P}_m = \sum_{k=1}^{K} \varphi_k \ \boldsymbol{a}_k \otimes \boldsymbol{a}_k$ and $\boldsymbol{\mathcal{T}}_m = \sum_{k=1}^{K} \varphi_k \ \boldsymbol{a}_k \otimes \boldsymbol{a}_k \otimes \boldsymbol{a}_k$, where $\boldsymbol{a}_k$ constructs the k^{th} column of a matrix, say matrix $\boldsymbol{A}$. Observe that third order moments have an exact CP decomposition of the form $\boldsymbol{\mathcal{T}}_m = [\![\boldsymbol{\varphi}; \boldsymbol{A}, \boldsymbol{A}, \boldsymbol{A}]\!]$.

By doing[14] some pre-processing steps[15] on a corpus (*i.e.*, a bunch of documents) and by keeping the words with more than 20% of the term-document frequency (*i.e.*, the frequency of each word in the corpus), a dictionary of size 14×14 ($D = 14$ words) is obtained. Then, by decomposing the estimated $\boldsymbol{P}_m$ and $\boldsymbol{\mathcal{T}}_m$, a pair of estimated probabilities ($\boldsymbol{\varphi}^{14}$, $\boldsymbol{A}^{14\times 14}$) is acquired. Since the tensor Power method [5] cannot handle $K > D$, we set the input rank $R = K = 14$. In the sequel, it is shown experimentally, even with this unfavorable value[16], constrained CP decompositions, such as SFBS, performs better.

Although in this corpus, the number of documents for all the topics is almost the same (which means that $\boldsymbol{\varphi}$ is expected to have a distribution close to the uniform one), the estimated $\boldsymbol{\varphi}$ by the tensor Power method [5], *i.e.*, $\widehat{\boldsymbol{\varphi}}_{\text{Power}}$, is not only non-uniform, but also is not a probability distribution at all, since it

[14] We used "`nltk`", "`Gensim`" and an implemented example available online [34].

[15] Some steps include removing stop words, punctuations and unnecessary characters, etc.

[16] Constrained CP decomposition results are much better for $14 < K \leq 20$ due to the fact that 20 topics exist in the considered data set.

does not lie in the probability simplex (*i.e.*, sum to one):

$$\widehat{\boldsymbol{\varphi}}_{\text{Power}} = [0.00, 0.02, 0.14, 0.03, 0.08, 0.26, 0.39,$$
$$0.35, 0.65, \mathbf{302.89}, 1.01, 0.87, 8.30, 0.95]^T.$$

As it can be observed in $\widehat{\boldsymbol{\varphi}}_{\text{Power}}$, its tenth element, which is in bold, dominates the others, which could be the consequence of rounding errors. Even if one projects $\widehat{\boldsymbol{\varphi}}_{\text{Power}}$ to the simplex set to obtain $\widehat{\boldsymbol{\varphi}}_{\text{Power-proj}}$, it would be meaningless according to the fact that "$\boldsymbol{\varphi}$ is expected to have a distribution close to the uniform", but we have: $\widehat{\boldsymbol{\varphi}}_{\text{Power-proj}} = \Big[0, 0, 0, 0, 0, 0, 0, 0, 0, \mathbf{1}, 0, 0, 0, 0\Big]^T$. On the other hand, the estimated $\boldsymbol{\varphi}$ by SFBS, *i.e.*, $\widehat{\boldsymbol{\varphi}}_{\text{SFBS}}$, is as follows:

$$\widehat{\boldsymbol{\varphi}}_{\text{SFBS}} = [0.10, 0.08, 0.09, 0.09, 0.10, 0.04, 0.02,$$
$$0.05, 0.06, 0.01, 0.06, 0.09, 0.08, 0.12]^T,$$

which is closer to uniform distribution than $\widehat{\boldsymbol{\varphi}}_{\text{Power-proj}}$. This again proves that applying constrained tensor decomposition is more reliable than unconstrained one.

In order to compare the performances on this real text data set, the ground truth values of $\boldsymbol{\varphi}$ and $\boldsymbol{A}$ are needed. To do this, we select a portion of 20 Newsgroup data set (four topics, namely "computer graphics", "baseball", "cryptography" and "Christianity"), and by keeping the words with term-frequency between 20% and 50%, a dictionary with $D = 17$ words and a corpus containing 1690 documents are obtained. To calculate the ground truth values of $\boldsymbol{\varphi}$, we divide the number of documents belonging to each topic by 1690. To calculate the ground truth of $\boldsymbol{A}$, we divide the occurrences of each word in the documents belonging to a topic by the total number of words in those documents.

By setting the input rank of all the algorithms to $K = 4$, $\widehat{\boldsymbol{\varphi}}^4$ and $\widehat{\boldsymbol{A}}^{17\times 4}$ are estimated and compared to the ground truth values. The performances are reported in Table 1, which shows that tensor Power method and Projected Power method in estimating $\boldsymbol{\varphi}$ perform much worse than constrained algorithms; how-

Table 1: Performances of estimating $\boldsymbol{\varphi}$ and $\boldsymbol{A}$. This experiment is carried out using 20 Newsgroups data set.

Algorithm	Error of $\widehat{\boldsymbol{\varphi}}$	Error of $\widehat{\boldsymbol{A}}$
Power method	0.855	0.449
Projected Power method	0.942	0.449
Simplex SFBS	0.629	0.467
Simplex AO-ADMM	0.579	0.479

ever, the estimation of $\boldsymbol{A}$ is a bit better[17]. In addition, constrained algorithms, such as SFBS, perform properly in both estimating $\boldsymbol{\varphi}$ and $\boldsymbol{A}$.

4.3. Discussion

Concerning constrained algorithms, the convergence of AO employed in AO-ADMM [10] is not guaranteed because of ignoring the proximal regularization, and, for the non-convex constraint ℓ_0, the convergence of ADMM has not yet been proved, whereas the convergence of SFBS is proved in Appendix B.

SFBS performs better than APG in noisy cases (cf. Fig. 1.d), while it is also computationally less expensive (because the calculation of coefficients is avoided), unlike APG. Moreover, BC-VMFB [13] force the user to adjust several parameters that have an important impact on the result, while the only parameter of SFBS is e, which appears in line 4 of Algorithm 3 or line 9 in Algorithm 4, for which we provided a proper range ($[1.5, 1.9]$) based on our experiments.

Concerning unconstrained algorithms, SFBS is much more robust against additive noise than Nway [9] or the tensor Power method [5]. Moreover, the tensor Power method and its variants does not have any convergence guarantee for an odd-order tensor (including the tensor of third order moments), whereas

[17]This is probably due to the L_2 normalization of the columns of $\widehat{\boldsymbol{A}}$ and storing them as the estimation of $\boldsymbol{\varphi}$ (cf. [5]). However, the non-normalized $\widehat{\boldsymbol{\varphi}}$ may suffer from rounding errors.

the convergence of SFBS is proved in Appendix B. More importantly, another critical drawback of the tensor Power method in face of real text data sets is the dominant estimated element (*i.e.*, a very large element in $\boldsymbol{\varphi}$, which drops out $\boldsymbol{\varphi}$ from the simplex set), and this problem cannot be solved with a projection onto the simplex set, as the Projected Power method does.

5. Conclusion and perspectives

In this paper, constrained tensor decompositions based on proximal approaches are investigated. It is discussed that the state-of-the-art algorithms either (i) suffer from a lack of complete convergence guarantee (*e.g.*, AO-ADMM[18] [10]) or (ii) suffer from a poor performance against additive noise (*e.g.*, APG [11]), or (iii) force the user to adjust several parameters (*e.g.*, BC-VMFB [13]), or (iv) are unable to decompose all kinds of tensors (*e.g.*, the tensor Power method [5] is limited to symmetric tensors whose ranks are smaller than their dimensions).

Our constrained tensor decomposition, SFBS, is based on Forward-Backward Splitting, and performs either better than or as well as AO-ADMM [10]. Furthermore, there exists a complete convergence proof for SFBS, which is not the case for AO-ADMM. SFBS performs also better than APG [11] in noisy cases and computationally costs less than APG, because the calculation of coefficients t_k and λ_k is avoided, unlike APG. In spite of BC-VMFB [13], SFBS requires to adjust only one parameter (the parameter in line 4 of Algorithm 3, e), which is easy to set according to the convergence condition.

Compared to unconstrained algorithms such as the tensor Power method [5], not only SFBS performs better and is much more robust against additive noise, but also enjoys theoretical convergence guarantees. Moreover, it is ensured that the probabilities estimated by SFBS always lie in the probability simplex, unlike

[18] The convergence of AO in AO-ADMM is not guaranteed due to ignoring the proximal regularization. Moreover, the convergence analysis of ADMM for the non-convex constraint ℓ_0 has not yet been proved.

the tensor Power method. However, unconstrained algorithms converge much faster than iterative algorithms including SFBS.

Working with large dimension tensors is a big challenge, as most iterative constrained algorithms are very time consuming, hence, intractable in practice. CPRAND [35] is an unconstrained CP decomposition, which is suitable for large dimensions, since it employs the *sample Khatri-Rao* function, sketching data for the calculation of Khatri-Rao products. Therefore, a future extension of SFBS would be to use the *sample Khatri-Rao* function of CPRAND in each iteration of SFBS.

Appendix A. Lipschitz constant

Let $h(\boldsymbol{A}^{(n)}) \triangleq \frac{1}{2}\|\boldsymbol{T}^{(n)} - \boldsymbol{A}^{(n)}\boldsymbol{W}\|_F^2$, where $\boldsymbol{W} = (\boldsymbol{A}^{(N)} \odot \ldots \odot \boldsymbol{A}^{(n+1)} \odot \boldsymbol{A}^{(n-1)} \odot \ldots \odot \boldsymbol{A}^{(1)})^T$. We show that $h(\boldsymbol{A}^{(n)})$ is β-Lipschitz gradient by calculating the corresponding Lipschitz constant of $\nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}^{(n)})$, as follows:

$$\|\nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{X}) - \nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{Y})\|_F \leq \beta\|\boldsymbol{X} - \boldsymbol{Y}\|_F \Rightarrow$$
$$\|(\boldsymbol{X} - \boldsymbol{Y})\boldsymbol{W}\boldsymbol{W}^T\|_F \leq \beta\|\boldsymbol{X} - \boldsymbol{Y}\|_F \Rightarrow \beta = \max \frac{\|(\boldsymbol{X} - \boldsymbol{Y})\boldsymbol{W}\boldsymbol{W}^T\|_F}{\|\boldsymbol{X} - \boldsymbol{Y}\|_F}.$$

Note that the definition of "spectral norm" of a matrix is [22]: $\|\boldsymbol{A}\|_\sigma \triangleq \max_{\|\boldsymbol{X}\|_F \neq 0} \frac{\|\boldsymbol{A}\boldsymbol{X}\|_F}{\|\boldsymbol{X}\|_F}$, which is equal to the maximum singular value of $\boldsymbol{A}$ with $\boldsymbol{X}$ as a matrix. Therefore, we have: $\beta = \|\boldsymbol{W}\boldsymbol{W}^T\|_\sigma$.

Appendix B. Convergence analysis

Since SFBS utilizes AO (to be exact, ALS), the convergence analysis of SFBS is firstly studied for AO in Appendix B.1, and then, the convergence guarantee of each step of AO is investigated in Appendix B.2.

Appendix B.1. The convergence of AO with Proximal Minimization

We refer the overall convergence of SFBS to the analysis performed in [36], namely, Proximal Alternating Linearized Minimization (PALM), which is suitable for the algorithm based on Proximal Forward-Backward. Let us rewrite (5)

in an unconstrained manner as follows:

$$\Psi(\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}) = f_d(\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}) + \sum_{n=1}^{N} r_{\mathcal{E}}(\boldsymbol{A}^{(n)}) + r_{\mathcal{E}}(\boldsymbol{\lambda}), \tag{B.1}$$

where $r_{\mathcal{E}}(.)$ or $r_{\mathcal{E}}(\boldsymbol{\lambda})$ could be indicator functions or ℓ_p norm, depending on the constraints on $\boldsymbol{A}^{(n)}$ and $\boldsymbol{\lambda}$, and $f_d(.)$ is the differentiable quadratic fidelity term, *i.e.*, $f_d(\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}) = \frac{1}{2}\|\boldsymbol{\mathcal{T}} - [\![\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}]\!]\|_F^2$.

In the sequel, we shall explain how SFBS satisfies the required assumptions of PALM by considering (B.1). Some classes of functions that satisfy the KL property are summarized in [11, Section 2.2], among which *semi-algebraic* functions are practical in the following explanation. Here, we suffice to mention some critical properties of semi-algebraic functions:

1- The sum of semi-algebraic functions is semi-algebraic.

2- The sum of a polynomial and a semi-algebraic function is semi-algebraic.

3- The indicator functions of a polyhedral set (such as a non-negative set and the probability simplex) is semi-algebraic.

4- Not only $\ell_1, \ell_2, \ell_\infty$ norms are semi-algebraic, but also the sum of ℓ_0 pseudo-norm and a polynomial is semi-algebraic [18, Example 5.4].

Listed below are the PALM assumptions [36] along with our own justifications to show how SFBS meets the required PALM assumptions:

Assumption 1 (i): $r_{\mathcal{E}}(.)$ and $r_{\mathcal{E}}(\boldsymbol{\lambda})$ in (B.1) are proper and lower semi-continuous (lsc). **Justification:** $r_{\mathcal{E}}(.)$ and $r_{\mathcal{E}}(\boldsymbol{\lambda})$ are are proper based on definition of proper functions (cf. Theorem 1). In addition, indicator functions and ℓ_p norm are lsc (cf. the definition in [22, Definition 1.5].

(ii): f_d is a C^1 function (the class of functions with first order differentiability). **Justification:** since f_d is quadratic, therefore it is a C^2 function, and hence it is a C^1 function.

Assumption 2 (i): $\inf \Psi(.) > -\infty, \inf r_{\mathcal{E}}(.) > -\infty, \inf r_{\mathcal{E}}(\boldsymbol{\lambda}) > -\infty$. **Justification:** $f_d \geq 0$, $r_{\mathcal{E}}(.) \geq 0, r_{\mathcal{E}}(\boldsymbol{\lambda}) \geq 0$.

(ii): f_d is globally Lipschitz (see [36] for a detailed definition). **Justification:** since f_d is quadratic, it is globally Lipschitz.

(iii): The inequalities (3.5) and (3.6) from [36] are trivially fulfilled. **Justification:** Assumption 2 (ii) is satisfied (cf. [36, Remark 3 (iii)]).

(iv): The inequality (3.7) from [36] is satisfied for f_d, whenever f_d is a C^2 function (cf. [36, Remark 3 (iv)]). **Justification:** as f_d is quadratic, it is a C^2 function.

Assumption 3 $\Psi(.)$ defined in (B.1) satisfies the KL property. **Justification:** according to the above-mentioned properties of semi-algebraic functions, the indicator function of a polyhedral set (such as non-negativity and the membership to the simplex set) and ℓ_p norm ($p = 0, 1, 2, \infty$) are semi-algebraic. In addition, $\Psi(.)$ is semi-algebraic as it is the sum of a polynomial and a semi-algebraic function. Therefore, $\Psi(.)$ satisfies the KL.

Appendix B.2. The convergence of each step of AO in SFBS

Each step of AO in SFBS is Forward-Backward Splitting, whose convergence is proved in [18]. This proof is not only applicable for continuous and convex functions, but also is usable for non-smooth and non-convex ones. According to the analysis in [18], the objective function in each step, should be a proper and lsc function, which has the KL property and is bounded from below. As mentioned before, (7) is proper (cf. Theorem 1) and lsc, since $\frac{1}{2}\|\boldsymbol{T}^{(n)} - \boldsymbol{A}^{(n)}\boldsymbol{W}\|_F^2$ is continuous and the regularization term is lsc based on Assumption 1 (i). Finally, (7) is bounded from below, and satisfies KL according to the semi-algebraic function properties.

The only assumption of the convergence analysis in [18] to be discussed is generating a *bounded* sequence by SFBS. The assumptions that guarantee the boundedness of generated sequence is discussed in [37, Remark 5]. For instance, the *coercivity* [22, Definition 3.25] of the objective function is simply sufficient to obtain a bounded sequence from SFBS. A function is coercive, if it is bounded

from below on bounded sets and $\liminf_{|x|\to\infty} \frac{f(x)}{x} = \infty$. It can be observed that (7) is coercive.

[1] F. L. Hitchcock, The expression of a tensor or a polyadic as a sum of products, Journal of Mathematics and Physics 6 (1-4) (1927) 164–189.

[2] P. Comon, Tensors: a brief introduction, IEEE Sig. Proc. Magazine 31 (3) (2014) 44–53.

[3] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, C. Faloutsos, Tensor decomposition for signal processing and machine learning, IEEE Transactions on Signal Processing 65 (13) (2017) 3551–3582.

[4] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, H. A. Phan, Tensor decompositions for signal processing applications: From two-way to multiway component analysis, IEEE signal processing magazine 32 (2) (2015) 145–163.

[5] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, M. Telgarsky, Tensor decompositions for learning latent variable models, J. Machine Learning Research 15 (2014) 2773–2832.

[6] E. Sobhani, P. Comon, C. Jutten, M. Babaie-Zadeh, Text mining with constrained tensor decomposition, in: International Conference on Machine Learning, Optimization, and Data Science (LOD), Springer, 2019, pp. 219–231. doi:978-3-030-37599-7_19.

[7] A. Cichocki, R. Zdunek, A. H. Phan, S.-i. Amari, Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation, John Wiley & Sons, 2009.

[8] S.-H. Hsieh, C.-S. Lu, S.-C. Pei, 2d sparse dictionary learning via tensor decomposition, in: 2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP), IEEE, 2014, pp. 492–496.

[9] B. W. Bader, T. G. Kolda, et al., Matlab tensor toolbox version 2.6, Available online (February 2015).
URL http://www.sandia.gov/~tgkolda/TensorToolbox/

[10] K. Huang, N. D. Sidiropoulos, A. P. Liavas, A flexible and efficient algorithmic framework for constrained matrix and tensor factorization, IEEE Transactions on Signal Processing 64 (19) (2016) 5052–5065.

[11] Y. Xu, W. Yin, A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion, SIAM Journal on imaging sciences 6 (3) (2013) 1758–1789.

[12] Y. Zhang, G. Zhou, Q. Zhao, A. Cichocki, X. Wang, Fast nonnegative tensor factorization based on accelerated proximal gradient and low-rank approximation, Neurocomputing 198 (2016) 148–154.

[13] X. Vu, C. Chaux, N. Thirion-Moreau, S. Maire, A proximal approach for nonnegative tensor decomposition, in: International Conference on Latent Variable Analysis and Signal Separation, Springer, 2017, pp. 201–210.

[14] Y. Wang, W. Yin, J. Zeng, Global convergence of admm in nonconvex nonsmooth optimization, Journal of Scientific Computing 78 (1) (2019) 29–63.

[15] X. Fu, S. Ibrahim, H.-T. Wai, C. Gao, K. Huang, Block-randomized stochastic proximal gradient for low-rank tensor factorization, IEEE Transactions on Signal Processing 68 (2020) 2170–2185.

[16] P. L. Combettes, J.-C. Pesquet, Proximal splitting methods in signal processing, in: Fixed-point algorithms for inverse problems in science and engineering, Springer, 2011, pp. 185–212, ch.10.

[17] N. Parikh, S. Boyd, Proximal algorithms, Foundations and Trends in optimization 1 (3) (2014) 127–239.

[18] H. Attouch, J. Bolte, B. F. Svaiter, Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized gauss–seidel methods, Mathematical Programming 137 (1-2) (2013) 91–129.

[19] P. Comon, Tensor decompositions—state of the art and applications, keynote address in ima conf, Mathematics in Signal Processing, Warwick, UK 375.

[20] L. Chiantini, G. Ottaviani, N. Vannieuwenhoven, An algorithm for generic and low-rank specific identifiability of complex tensors, SIAM Journal on Matrix Analysis and Applications 35 (4) (2014) 1265–1287.

[21] Y. Censor, S. A. Zenios, et al., Parallel optimization: Theory, algorithms, and applications, Oxford University Press on Demand, 1997.

[22] R. T. Rockafellar, R. J.-B. Wets, Variational analysis, Vol. 317, Springer Science & Business Media, 2009.

[23] J.-J. Moreau, Fonctions convexes duales et points proximaux dans un espace hilbertien, Comptes Rendus Acad. Sciences Paris Serie A Math. 255 (1962) 2897–2899.

[24] R. Rockafellar, Convex analysis in the calculus of variations, in: Advances in Convex Analysis and Global Optimization, Springer, 2001, pp. 135–151.

[25] P. L. Combettes, V. R. Wajs, Signal recovery by proximal forward-backward splitting, Multiscale Modeling & Simulation 4 (4) (2005) 1168–1200.

[26] A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM journal on imaging sciences 2 (1) (2009) 183–202.

[27] L. Condat, Fast projection onto the simplex and the ℓ_1 ball, Mathematical Programming 158 (1-2) (2016) 575–585.

[28] K. Huang, AO-ADMM-Code @ONLINE, https://www.catalyzex.com/paper/arxiv:1506.04209#clicktoread/ (Oct. 2015).

[29] Y. Xu, W. Yin, APG-Code @ONLINE, https://xu-yangyang.github.io/BCD/ (2013).

[30] R. Bro, H. A. Kiers, A new efficient method for determining the number of components in parafac models, Journal of Chemometrics: A Journal of the Chemometrics Society 17 (5) (2003) 274–286.

[31] H. W. Kuhn, The hungarian method for the assignment problem, Naval research logistics quarterly 2 (1-2) (1955) 83–97.

[32] E. Sobhani, P. Comon, C. Jutten, M. Babaie-Zadeh, Corrindex: a permutation invariant performance index, Signal Processing (2022) 108457.

[33] K. Nigam, 20 Newsgroups data sets @ONLINE, http://www.cs.cmu.edu/~TextLearning/datasets.html (Feb. 2000).

[34] S. Prabhakaran, Pre-processing steps @ONLINE, https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/#20topicdistributionacrossdocuments (Mar. 2018).

[35] C. Battaglino, G. Ballard, T. G. Kolda, A practical randomized cp tensor decomposition, SIAM Journal on Matrix Analysis and Applications 39 (2) (2018) 876–901.

[36] J. Bolte, S. Sabach, M. Teboulle, Proximal alternating linearized minimization for nonconvex and nonsmooth problems, Mathematical Programming 146 (1-2) (2014) 459–494.

[37] H. Attouch, J. Bolte, P. Redont, A. Soubeyran, Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the kurdyka-łojasiewicz inequality, Mathematics of operations research 35 (2) (2010) 438–457.