Neural Computing and Applications (2021) 33:17369–17387
https://doi.org/10.1007/s00521-021-06325-8

ORIGINAL ARTICLE

Sparse nonnegative tensor decomposition using proximal algorithm and inexact block coordinate descent scheme

Deqing Wang[1,2] · Zheng Chang[2,3] · Fengyu Cong[1,2,4,5]

Received: 14 October 2020 / Accepted: 8 July 2021 / Published online: 4 October 2021

Abstract
Nonnegative tensor decomposition is a versatile tool for multiway data analysis, by which the extracted components are nonnegative and usually sparse. Nevertheless, the sparsity is only a side effect and cannot be explicitly controlled without additional regularization. In this paper, we investigated the nonnegative CANDECOMP/PARAFAC (NCP) decomposition with the sparse regularization item using l_1-norm (sparse NCP). When high sparsity is imposed, the factor matrices will contain more zero components and will not be of full column rank. Thus, the sparse NCP is prone to rank deficiency, and the algorithms of sparse NCP may not converge. In this paper, we proposed a novel model of sparse NCP with the proximal algorithm. The subproblems in the new model are strongly convex in the block coordinate descent (BCD) framework. Therefore, the new sparse NCP provides a full column rank condition and guarantees to converge to a stationary point. In addition, we proposed an inexact BCD scheme for sparse NCP, where each subproblem is updated multiple times to speed up the computation. In order to prove the effectiveness and efficiency of the sparse NCP with the proximal algorithm, we employed two optimization algorithms to solve the model, including inexact alternating nonnegative quadratic programming and inexact hierarchical alternating least squares. We evaluated the proposed sparse NCP methods by experiments on synthetic, real-world, small-scale, and large-scale tensor data. The experimental results demonstrate that our proposed algorithms can efficiently impose sparsity on factor matrices, extract meaningful sparse components, and outperform state-of-the-art methods.

Keywords Tensor decomposition · Nonnegative CANDECOMP/PARAFAC decomposition · Sparse regularization · Proximal algorithm · Inexact block coordinate descent

1 Introduction

1.1 Background

Nonnegative tensor decomposition is a powerful tool in signal processing and machine learning [10, 35]. Nonnegative CANDECOMP/PARAFAC (NCP), as an important decomposition method, has been widely applied to processing multiway data, such as hyperspectral data [39], electroencephalograph (EEG) data [11], fluorescence excitation-emission matrix (EEM) data [13], neural data [46], and many other multiway tensor data [30]. In many cases, the extracted components by NCP are not only nonnegative but also sparse. For example, the spectral components from EEG tensor decomposition are usually very sparse, representing the narrow-band frequencies of some brain activities [11]. For another example, after decomposing EEM tensor, a component in the sample mode denotes the concentrations of a compound in all samples [5], which is sometimes also sparse. The nonnegative constraint in NCP will naturally lead to sparse results. However, this sparsity is only a side effect, which cannot be controlled to a certain level [18]. Without properly controlling the sparsity, the intrinsic components in the data cannot be extracted precisely, especially in low signal-to-noise ratio conditions. Therefore, in order to extract meaningful and accurate sparse components, additional sparse regularization is necessary for NCP tensor decomposition.

The design of NCP decomposition with explicit sparse regularization (sparse NCP) will benefit a lot from the methods in nonnegative matrix factorization (NMF) cases.

Extended author information available on the last page of the article

On the one hand, an early study of NMF [18] proposed the method of projecting components into sparse vectors at some sparsity level. However, this method keeps all components at the same fixed sparsity level, which is not in line with the true sparsity of different components in the data. On the other hand, incorporating sparse regularization items into the optimization model is a popular method. The l_1-norm is a conventional and effective regularizer to impose sparsity for signal processing [6]. The reason is that, for most underdetermined linear equations, the optimization problem with l_1-norm regularization can yield strong sparsity [12]. More information about the sparse regularization can be found in [2, 34, 50].

Many works have been devoted to the tensor decomposition with sparse regularization, but only a few can be found for NCP. The works of [1, 21] and [29] studied the sparse regularization for tensor decomposition using l_1-norm and trace norm, but they only focused on the unconstrained CP model without the nonnegative constraint. The works of [14, 28, 31] and [47] proposed the methods of imposing sparsity by the l_1-norm on nonnegative Tucker decomposition. However, these methods are not suitable for large-scale problems [47], and their effectiveness is unknown to NCP. Kim et al. considered solving sparse NCP using ANLS [23]. Nevertheless, ANLS seriously suffers from rank deficiency caused by high sparsity or zero components in the factor matrices. Recently, Huang et al. have proposed an alternating optimization-based ADMM (AO-ADMM) method, which can handle the l_1-norm regularization item in NCP [19]. Nevertheless, there is no experimental evaluation on the sparse NCP in [19]. The work [32] proposed a sparse NCP algorithm, which is targeted at the multiway co-clustering. In practical applications, the sparse NCP may face the following two major challenges.

One challenge is that when the tensor data are highly sparse or strongly sparse regularization is imposed on the decomposition, more and more zero components will appear in the factor matrices. Thus, the factor matrices are not of full column rank, which will cause the rank deficiency problem. The rank deficiency will further cause a poor convergence of the tensor decomposition algorithm. It is introduced that the proximal algorithm is an excellent method to improve the convergence of a mathematical optimization method [4]. In an optimization problem by iterations, the proximal algorithm is constructed by adding a proximal regularization item to the original model. This proximal item is the squared Frobenius norm of the difference between the current variable and its value in previous iteration [4]. The proximal algorithm can naturally be incorporated into tensor decomposition [26].

The other challenge is that, for large-scale tensor data, the process of sparse NCP decomposition might be inefficient. It is reported that the inexact block coordinate descent scheme could accelerate the convergence and is very beneficial to the large-scale problem [15, 40]. Hence, the inexact scheme can be employed in the sparse NCP problem.

1.2 Contribution

Firstly, in this paper, we propose a novel sparse NCP method with the l_1-norm and the proximal algorithm. The proposed sparse NCP will overcome the rank deficiency and guarantee the decomposition to converge to a stationary point. The block coordinate descent (BCD) is one of the main techniques for tensor decomposition, especially the constrained one [24]. In BCD framework, each factor matrix is updated as a subproblem alternatively while other factor matrices are fixed. By the proximal algorithm, the proximal regularization item can make the subproblems strongly convex [26] and can provide a full column rank condition for the sparse NCP.

Secondly, we develop an inexact BCD scheme for the novel sparse NCP model. The inexact scheme will speed up the computation of the sparse NCP, especially in large-scale cases. Specifically, in the inexact BCD scheme, the subproblem of the sparse NCP is iterated multiple times for updating a factor matrix.

Thirdly, in order to prove the viability of the sparse NCP model with the proximal algorithm and the inexact scheme, we employ two efficient optimization algorithms to solve the model, including inexact alternating nonnegative quadratic programming and inexact hierarchical alternating least squares. We evaluate the proposed sparse NCP methods on synthetic, real-world, small-scale and large-scale tensor data. By properly selecting and tuning the sparse regularization, the effectiveness and efficiency of the sparse NCP methods are demonstrated to impose sparsity on factor matrices.

1.3 Organization

The rest of this paper is organized as follows. Section 2 introduces some preliminaries. In Sect. 3, we describe the mathematical model of sparse NCP with the proximal algorithm and inexact BCD scheme. Section 4 elucidates the solutions to the sparse NCP model using the optimization methods. Section 5 describes the detailed experiments on synthetic and real-world datasets. Some critical observations are discussed in Sect. 6. Finally, we conclude our paper in Sect. 7.

2 Preliminaries

In this paper, operator $\circ$ represents the outer product of vectors, $\odot$ represents the Khatri-Rao product, $*$ represents the Hadamard product that is the elementwise matrix product, $\langle\ \rangle$ represents the inner product, $[\![\ ]\!]$ represents Kruskal operator and $[\ ]_+$ represents nonnegative projection. $||\ ||_F$ denotes Frobenius norm, and $||\ ||_1$ denotes l_1-norm. Basics of tensor computation and multi-linear algebra can be found in review papers [25, 35].

2.1 Nonnegative CP decomposition

Given an Nth-order nonnegative tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ and a positive number R, nonnegative CANDECOMP/PARAFAC (NCP) is to solve the following minimization problem:

$$\min_{\boldsymbol{A}^{(1)},\ldots,\boldsymbol{A}^{(N)}} \frac{1}{2}||\boldsymbol{\mathcal{X}} - [\![\boldsymbol{A}^{(1)},\ldots,\boldsymbol{A}^{(N)}]\!]||_F^2 \quad \text{s.t. } \boldsymbol{A}^{(n)} \geqslant 0 \text{ for } n = 1,\ldots,N, \tag{1}$$

where $\boldsymbol{A}^{(n)} \in \mathbb{R}^{I_n\times R}$ for $n = 1,\ldots,N$ are the estimated factor matrices in different modes, I_n is the size in mode-n, and R is the initial number of components. We use $\mathscr{F}_{\text{tensor}}(\boldsymbol{A}) = \mathscr{F}_{\text{tensor}}(\boldsymbol{A}^{(1)},\ldots,\boldsymbol{A}^{(N)})$ to denote the objective function in (8). The estimated factor matrices in Kruskal operator can be represented by the sum of R rank-1 tensors in outer product form:

$$[\![\boldsymbol{A}^{(1)},\ldots,\boldsymbol{A}^{(N)}]\!] = \sum_{r=1}^{R} \boldsymbol{\mathcal{Y}}_r = \sum_{r=1}^{R} \boldsymbol{a}_r^{(1)} \circ \cdots \circ \boldsymbol{a}_r^{(N)}, \tag{2}$$

where $\boldsymbol{a}_r^{(n)}$ represents the rth column of $\boldsymbol{A}^{(n)}$.

Let $\boldsymbol{X}_{(n)} \in \mathbb{R}^{I_n\times\prod_{\tilde{n}=1,\tilde{n}\neq n}^{N} I_{\tilde{n}}}$ represent the mode-n unfolding (matricization) of original tensor $\boldsymbol{\mathcal{X}}$. The mode-n unfolding of the estimated tensor in Kruskal operator $[\![\boldsymbol{A}^{(1)},\ldots,\boldsymbol{A}^{(N)}]\!]$ can be written as $\boldsymbol{A}^{(n)}\left(\boldsymbol{B}^{(n)}\right)^T$, in which

$$\boldsymbol{B}^{(n)} = \left(\boldsymbol{A}^{(N)} \odot \cdots \odot \boldsymbol{A}^{(n+1)} \odot \boldsymbol{A}^{(n-1)} \odot \cdots \odot \boldsymbol{A}^{(1)}\right) \in \mathbb{R}^{\prod_{\tilde{n}=1,\tilde{n}\neq n}^{N} I_{\tilde{n}}\times R}$$

. In BCD framework, factor $\boldsymbol{A}^{(n)}$ is updated alternatively by a subproblem in every iteration, which is equal to the following minimization problem:

$$\min_{\boldsymbol{A}^{(n)}} \mathscr{F}\left(\boldsymbol{A}^{(n)}\right) = \frac{1}{2}||\boldsymbol{X}_{(n)} - \boldsymbol{A}^{(n)}\left(\boldsymbol{B}^{(n)}\right)^T||_F^2 \quad \text{s.t. } \boldsymbol{A}^{(n)} \geqslant 0. \tag{3}$$

The partial gradient (or partial derivative) of $\mathscr{F}\left(\boldsymbol{A}^{(n)}\right)$ with respect to $\boldsymbol{A}^{(n)}$ is

$$\frac{\partial}{\partial \boldsymbol{A}^{(n)}} \mathscr{F}\left(\boldsymbol{A}^{(n)}\right) = \boldsymbol{A}^{(n)}\left[\left(\boldsymbol{B}^{(n)}\right)^T \boldsymbol{B}^{(n)}\right] - \boldsymbol{X}_{(n)}\boldsymbol{B}^{(n)}. \tag{4}$$

In (4), the item $\boldsymbol{X}_{(n)}\boldsymbol{B}^{(n)}$ is called the *Matricized Tensor Times Khatri-Rao Product* (MTTKRP) [35]. The item $\left(\boldsymbol{B}^{(n)}\right)^T\boldsymbol{B}^{(n)}$ can be computed efficiently by

$$\begin{aligned}\left(\boldsymbol{B}^{(n)}\right)^T\boldsymbol{B}^{(n)} = &\left[\left(\boldsymbol{A}^{(N)}\right)^T\boldsymbol{A}^{(N)}\right] * \cdots \\ &* \left[\left(\boldsymbol{A}^{(n+1)}\right)^T\boldsymbol{A}^{(n+1)}\right] * \left[\left(\boldsymbol{A}^{(n-1)}\right)^T\boldsymbol{A}^{(n-1)}\right] \\ &* \cdots * \left[\left(\boldsymbol{A}^{(1)}\right)^T\boldsymbol{A}^{(1)}\right].\end{aligned} \tag{5}$$

2.2 Sparse regularization with l_1-norm

In order to impose sparsity to the factor matrices, it is natural to incorporate the sparse regularization items using l_1-norm [9, 47] into the objective function in (1), which leads to the following basic sparse NCP problem:

$$\min_{\boldsymbol{A}^{(1)},\ldots,\boldsymbol{A}^{(N)}} \frac{1}{2}||\boldsymbol{\mathcal{X}} - [\![\boldsymbol{A}^{(1)},\ldots,\boldsymbol{A}^{(N)}]\!]||_F^2 + \sum_{n=1}^{N}\beta_n\sum_{r=1}^{R}||\boldsymbol{a}_r^{(n)}||_1 \quad \text{s.t. } \boldsymbol{A}^{(n)} \geqslant 0 \text{ for } n = 1,\ldots,N, \tag{6}$$

where β_n are positive sparse regularization parameters in parameter vectors $\boldsymbol{\beta} \in \mathbb{R}^{N\times 1}$. The subproblem can be written as the following optimization problem

$$\min_{\boldsymbol{A}^{(n)}} \mathscr{F}_0\left(\boldsymbol{A}^{(n)}\right) = \frac{1}{2}\left\|\boldsymbol{X}_{(n)} - \boldsymbol{A}^{(n)}\left(\boldsymbol{B}^{(n)}\right)^T\right\|_F^2 + \beta_n\sum_{r=1}^{R}\left\|\boldsymbol{a}_r^{(n)}\right\|_1 \quad \text{s.t. } \boldsymbol{A}^{(n)} \geqslant 0. \tag{7}$$

In the objective function of the subproblem, the sparse regularization is imposed on the factor matrix $\boldsymbol{A}^{(n)}$ by the l_1-norm.

3 The proposed sparse NCP model

3.1 Sparse NCP with proximal algorithm

The basic sparse NCP in (6) has a serious drawback. When strongly sparse regularization is imposed in (6), many zero columns will appear in the factor matrices $\boldsymbol{A}^{(n)}$. Thus, both $\boldsymbol{A}^{(n)}$ and $\boldsymbol{B}^{(n)}$ cannot guarantee to be of full column rank. Therefore, the basic sparse NCP model in (6) will suffer from rank deficiency and cannot guarantee to converge.

In order to overcome the above drawback, we propose the following sparse NCP model with proximal algorithm (a proximal regularization item using squared Frobenius norm):

$$\min_{A^{(1)},\ldots,A^{(N)}} \left\{ \frac{1}{2} \left\| \mathcal{X} - [\![A^{(1)},\ldots,A^{(N)}]\!] \right\|_F^2 + \sum_{n=1}^{N} \frac{\alpha_n}{2} \left\| \widetilde{A}^{(n)} - A^{(n)} \right\|_F^2 + \sum_{n=1}^{N} \beta_n \sum_{r=1}^{R} \left\| a_r^{(n)} \right\|_1 \right\} \quad (8)$$
$$\text{s.t. } A^{(n)} \geqslant 0 \text{ for } n = 1,\ldots,N,$$

where $\widetilde{A}^{(n)}$ is the value of the factor $A^{(n)}$ in previous iteration during updating and α_n are positive regularization parameters in vectors $\boldsymbol{\alpha} \in \mathbb{R}^{N\times1}$.

In BCD framework, the subproblem of model (8) can be written in the following minimization problem:

$$\min_{A^{(n)}} \mathscr{F}_{\mathrm{PROX}}\left(A^{(n)}\right) = \left\{ \frac{1}{2} \left\| X_{(n)} - A^{(n)} \left(B^{(n)}\right)^T \right\|_F^2 + \frac{\alpha_n}{2} \left\| \widetilde{A}^{(n)} - A^{(n)} \right\|_F^2 + \beta_n \sum_{r=1}^{R} \left\| a_r^{(n)} \right\|_1 \right\}$$
$$\text{s.t. } A^{(n)} \geqslant 0. \quad (9)$$

The objective function $\mathscr{F}_{\mathrm{PROX}}\left(A^{(n)}\right)$ can be further represented by the following form:

$$\mathscr{F}_{\mathrm{PROX}}\left(A^{(n)}\right) = \frac{1}{2} \left\| \begin{pmatrix} X_{(n)}^T \\ \sqrt{\alpha_n}\left(\widetilde{A}^{(n)}\right)^T \end{pmatrix} - \begin{pmatrix} B^{(n)} \\ \sqrt{\alpha_n} I_R \end{pmatrix} \left(A^{(n)}\right)^T \right\|_F^2 + \beta_n \sum_{r=1}^{R} \left\| a_r^{(n)} \right\|_1. \quad (10)$$

In (10), it is clear to see that the item $\begin{pmatrix} B^{(n)} \\ \sqrt{\alpha_n} I_R \end{pmatrix}$ must be of full column rank even though $B^{(n)}$ is not of full column rank. Thus, the proposed sparse NCP with the proximal algorithm can successfully overcome the rank deficiency problem in the objective function.

3.2 Inexact block coordinate descent scheme

The BCD is a main framework to solve tensor decomposition. It is reported that the inexact BCD scheme could accelerate the computation [15, 40]. Specifically, the factor matrices $A^{(n)}, n = 1,\ldots,N$, are updated alternatively in outer iterations; meanwhile, in the subproblem (9), the factor $A^{(n)}$ is also updated several times in inner iterations. The procedures of the inexact scheme are listed in Algorithm 1.

Algorithm 1: Inexact BCD Scheme for Sparse NCP

```
1 Initialization and preparation steps;
2 repeat
3     for $n = 1$ to $N$ do
4         Basic computations before updating $A^{(n)}$;
5         repeat
6             Update $A^{(n)}$ by $A^{(n)} = \operatorname{argmin}_{A^{(n)} \geqslant 0} \mathscr{F}_{\mathrm{PROX}}\left(A^{(n)}\right)$;
7         until finite number of inner iterations is reached;
8     end
9 until outer termination criterion is reached;
```

3.3 Convergence analysis

The proposed sparse NCP method in (8) can guarantee to converge to a stationary point.

Proposition 1 *Every limit point of the sequence $\left\{A_k^{(1)},\ldots,A_k^{(N)}\right\}_{k=1}^{\infty}$ generated by the sparse NCP in Algorithm 1 is a stationary point of (6).*

Proof The objective function $\mathscr{F}_{\mathrm{PROX}}\left(A^{(n)}\right)$ in (9) with the proximal regularization item is strictly convex [4]. Moreover, $\mathscr{F}_{\mathrm{PROX}}\left(A^{(n)}\right)$ is a proximal upper bound [17, 33] of the objective function $\mathscr{F}_0\left(A^{(n)}\right)$ in (7). Using the inexact block coordinate descent scheme, the subproblem in Algorithm 1 is updated by a finite number of inner iterations. According to the Theorem 2 in [49], every limit point of the sequence $\left\{A_k^{(1)},\ldots,A_k^{(N)}\right\}_{k=1}^{\infty}$ generated by the sparse NCP in Algorithm 1 is a stationary point of (6). □

4 Optimization methods for solving sparse NCP

In order to prove the viability and effectiveness of the novel sparse NCP with the proximal algorithm and inexact scheme, we employ the following two optimization methods to solve the model.

4.1 Alternating nonnegative quadratic programming

First, we utilize a method that is based on a general form of the alternating nonnegative least squares (ANLS). The classical ANLS is an important tool for NMF and NCP [24]. Many efficient optimization algorithms were proposed to solve the nonnegative least squares (NNLS) subproblems, such as active-set (AS) [20] and block principal pivoting (BPP) [22]. However, there are two limitations to the application of ANLS to sparse NCP. The first limitation is that ANLS is very prone to rank deficiency. The proximal algorithm can tackle this limitation in our sparse NCP model. The second limitation is that the subproblem of our proposed sparse NCP model cannot be represented in a least squares form due to the l_1-norm regularization, which can be clearly seen in (10). Therefore, some new forms of the objective function in (8) should be considered.

Inspired by [27], the subproblem of the proposed sparse NCP in (9) can be represented in the nonnegative quadratic programming (NNQP) form as the following problem:

$$\min_{\boldsymbol{A}^{(n)}} \sum_{i=1}^{I_n} \left\{ \frac{1}{2} \left[\boldsymbol{A}^{(n)}\right]_{(i,:)} \boldsymbol{M} \left[\boldsymbol{A}^{(n)}\right]^T_{(i,:)} + \boldsymbol{N}_{(i,:)} \left[\boldsymbol{A}^{(n)}\right]^T_{(i,:)} + \frac{1}{2} \left[\boldsymbol{X}_{(n)}\right]_{(i,:)} \left[\boldsymbol{X}_{(n)}\right]^T_{(i,:)} + \frac{\alpha_n}{2} \left[\widetilde{\boldsymbol{A}}^{(n)}\right]_{(i,:)} \left[\widetilde{\boldsymbol{A}}^{(n)}\right]^T_{(i,:)} \right\}$$
$$\text{s.t. } \boldsymbol{A}^{(n)} \geqslant 0, \tag{11}$$

where $[\,]_{(i,:)}$ represents the ith row of a matrix, $\boldsymbol{M} = \left(\boldsymbol{B}^{(n)}\right)^T \boldsymbol{B}^{(n)} + \alpha_n \boldsymbol{I}_R$, $\boldsymbol{N} = \beta_n \boldsymbol{E} - \boldsymbol{X}_{(n)} \boldsymbol{B}^{(n)} - \alpha_n \widetilde{\boldsymbol{A}}^{(n)}$ and $\boldsymbol{E}$ is a matrix of all ones. In fact, NNQP is a general form of NNLS.

The above-mentioned optimization methods for NNLS can also be used to solve NNQP problem. In this study, we only use block principal pivoting (BPP) [22] as the NNQP solver, which has been proven to be a very efficient method [22, 24]. The solver of BPP contains multiple inner iterations. We limited the inner iterations by several times in the inexact scheme. We name the method of solving tensor decomposition using NNQP as alternating nonnegative quadratic programming (ANQP). Furthermore, we abbreviated the method of solving the sparse NCP with the proximal algorithm using ANQP as PROX-ANQP. Algorithm 2 explicates the PROX-ANQP method.

Algorithm 2: PROX-ANQP for sparse NCP

```
Input  : $\mathcal{X}, R, \boldsymbol{\alpha}, \boldsymbol{\beta}$
Output: $\boldsymbol{A}^{(n)}, n = 1, \dots, N$
1  Initialize $\boldsymbol{A}^{(n)} \in \mathbb{R}^{I_n \times R}, n = 1, \dots, N$, using nonnegative
   random numbers;
2  repeat
3      for $n = 1$ to $N$ do
4          Make mode-$n$ unfolding of $\mathcal{X}$ as $\boldsymbol{X}_{(n)}$ and compute
           MTTKRP $\boldsymbol{X}_{(n)} \boldsymbol{B}^{(n)}$;
5          Compute $\left(\boldsymbol{B}^{(n)}\right)^T \boldsymbol{B}^{(n)}$ based on (5);
6          $\boldsymbol{X}_{(n)} \boldsymbol{B}^{(n)} \leftarrow \boldsymbol{X}_{(n)} \boldsymbol{B}^{(n)} + \alpha_n \widetilde{\boldsymbol{A}}^{(n)} - \beta_n \boldsymbol{E}$;
7          $\left(\boldsymbol{B}^{(n)}\right)^T \boldsymbol{B}^{(n)} \leftarrow \left(\boldsymbol{B}^{(n)}\right)^T \boldsymbol{B}^{(n)} + \alpha_n \boldsymbol{I}_R$;
8          repeat
9              Update factor $\boldsymbol{A}^{(n)}, n = 1, \dots, N$, using NNQP
               based on BPP method:
10             $\boldsymbol{A}^{(n)} = \operatorname{argmin}_{\boldsymbol{A}^{(n)} \geq 0} \mathcal{F}_{\text{PROX}}\left(\boldsymbol{A}^{(n)}\right)$
11             $= \texttt{NNQP\_BPP}\left(\boldsymbol{X}_{(n)} \boldsymbol{B}^{(n)}, \left(\boldsymbol{B}^{(n)}\right)^T \boldsymbol{B}^{(n)}\right)$;
12         until finite number of inner iterations is reached;
13     end
14 until outer termination criterion is reached;
15 return $\boldsymbol{A}^{(n)}, n = 1, \dots, N$.
```

4.2 Inexact hierarchical alternating least squares

Second, we employ an inexact hierarchical alternating least squares (iHALS) method for solving the sparse NCP with the proximal algorithm. The conventional HALS is an efficient method of updating each factor column by column [7, 9]. However, the HALS method has two major drawbacks to solving the sparse NCP.

First, HALS is also very prone to rank deficiency. Specifically, if a column of the factor matrix $\boldsymbol{A}^{(n)}$ becomes a zero vector, the HALS will break down [22]. One practical remedy is to replace the zero elements with a small positive value [9], such as 10^{-16}. However, by this modification, the obtained factor matrices are not sparse anymore.

Second, HALS suffers from the caveat problem (see Section 5.2 in [22]). Specifically, the unbalanced scales will appear in different columns and factors. For example, one column in the first factor might have a scale of 10^{-8} and the corresponding column in the second factor might have a scale of 10^{8}. At the same time, another column in the first factor might have a scale of 10^{16} and the corresponding column in the third factor might have a scale of 10^{-16}. One common method of controlling the unbalanced scales is to normalize all columns to unit vectors in the factors [9]. However, by factor normalization, the factor columns will never become zeros vectors. Hence it is impossible to impose sparsity efficiently.

The proximal algorithm in our sparse NCP will overcome the above drawbacks. We have mentioned that the proximal will guarantee the full column rank in the model. Moreover, the proximal regularization item in sparse NCP can keep all columns in factors on a balanced scale.

Next, we will introduce the solution of the model in (8) using the iHALS method. For the sake of simplification, we use $\boldsymbol{a}_r$ and $\boldsymbol{b}_r$ instead of $\boldsymbol{a}_r^{(n)}$ and $\boldsymbol{b}_r^{(n)}$ in this part, which are the rth column of $\boldsymbol{A}^{(n)}$ and $\boldsymbol{B}^{(n)}$, respectively. We also use $\left[\boldsymbol{A}^{(n)}\right]_{(:,r)} = \boldsymbol{a}_r \in \mathbb{R}^{I_n \times 1}$ to represent the column of a matrix, and $\left[\boldsymbol{A}^{(n)}\right]_{(i,r)} = a_{ir}^{(n)}$ to represent an element in a matrix.

The objective function in (9) can be further represented as

$$\mathcal{F}\left(\boldsymbol{A}^{(n)}\right) = \frac{1}{2} \left\| \boldsymbol{X}_{(n)} - \sum_{r=1}^{R} \boldsymbol{a}_r \boldsymbol{b}_r^T \right\|_F^2 + \frac{\alpha_n}{2} \sum_{r=1}^{R} \|\boldsymbol{a}_r - \widetilde{\boldsymbol{a}}_r\|_2^2 + \beta_n \sum_{r=1}^{R} \|\boldsymbol{a}_r\|_1, \tag{12}$$

where $\widetilde{\boldsymbol{a}}_r$ is the rth column of $\widetilde{\boldsymbol{A}}^{(n)}$. The minimization problem for (12) can be solved iteratively by columnwise subproblems:

$$\min_{\boldsymbol{a}_r} \mathscr{F}_r = \frac{1}{2}\left\|\boldsymbol{Z}_r - \boldsymbol{a}_r\boldsymbol{b}_r^T\right\|_F^2 + \frac{\alpha_n}{2}||\boldsymbol{a}_r - \widetilde{\boldsymbol{a}}_r||_2^2 + \beta_n||\boldsymbol{a}_r||_1 \quad \text{s.t. } \boldsymbol{a}_r \geqslant 0, \tag{13}$$

for $r = 1, \ldots, R$, in which

$$\boldsymbol{Z}_r = \boldsymbol{X}_{(n)} - \sum_{\tilde{r}=1,\tilde{r}\neq r}^{R} \boldsymbol{a}_{\tilde{r}}\boldsymbol{b}_{\tilde{r}}^T. \tag{14}$$

The partial derivative of $\mathscr{F}_r$ with respect to $\boldsymbol{a}_r$ is

$$\begin{aligned}\frac{\partial \mathscr{F}_r}{\partial \boldsymbol{a}_r} &= \left(\boldsymbol{a}_r\boldsymbol{b}_r^T - \boldsymbol{Z}_r\right)\boldsymbol{b}_r + \alpha_n\boldsymbol{a}_r - \alpha_n\widetilde{\boldsymbol{a}}_r + \beta_n\boldsymbol{1}, \\ &= \left(\boldsymbol{b}_r^T\boldsymbol{b}_r + \alpha_n\right)\boldsymbol{a}_r - \left(\boldsymbol{Z}_r\boldsymbol{b}_r + \alpha_n\widetilde{\boldsymbol{a}}_r - \beta_n\boldsymbol{1}\right),\end{aligned} \tag{15}$$

where $\boldsymbol{1} \in \mathbb{R}^{I_n\times 1}$ is a vector with all elements equal to 1. When $\frac{\partial \mathscr{F}_r}{\partial \boldsymbol{a}_r} = 0$, nonnegative column vector $\boldsymbol{a}_r$ can be updated as

$$\boldsymbol{a}_r \leftarrow \left[\frac{\boldsymbol{Z}_r\boldsymbol{b}_r + \alpha_n\widetilde{\boldsymbol{a}}_r - \beta_n\boldsymbol{1}}{\boldsymbol{b}_r^T\boldsymbol{b}_r + \alpha_n}\right]_+, \tag{16}$$

which is a closed form solution of (13) according to the Theorem 2 in [24].

A fast HALS method was utilized to solve the large-scale problem [7, 24]. We use the same idea to solve the sparse NCP problem. $\boldsymbol{Z}_r$ in (14) can also be represented as

$$\boldsymbol{Z}_r = \boldsymbol{X}_{(n)} - \sum_{\tilde{r}=1}^{R} \boldsymbol{a}_{\tilde{r}}\boldsymbol{b}_{\tilde{r}}^T + \widetilde{\boldsymbol{a}}_r\boldsymbol{b}_r^T. \tag{17}$$

Replacing $\boldsymbol{Z}_r$ in (16) by (17), we obtain the new update rule for $\boldsymbol{a}_r$ as shown in (18).

$$\begin{aligned}\boldsymbol{a}_r &\leftarrow \left[\frac{\left(\boldsymbol{X}_{(n)} - \sum_{\tilde{r}=1}^{R}\boldsymbol{a}_{\tilde{r}}\boldsymbol{b}_{\tilde{r}}^T + \widetilde{\boldsymbol{a}}_r\boldsymbol{b}_r^T\right)\boldsymbol{b}_r + \alpha_n\widetilde{\boldsymbol{a}}_r - \beta_n\boldsymbol{1}}{\boldsymbol{b}_r^T\boldsymbol{b}_r + \alpha_n}\right]_+ \\ &= \left[\widetilde{\boldsymbol{a}}_r + \frac{\boldsymbol{X}_{(n)}\boldsymbol{b}_r - \sum_{\tilde{r}=1}^{R}\boldsymbol{a}_{\tilde{r}}\boldsymbol{b}_{\tilde{r}}^T\boldsymbol{b}_r - \beta_n\boldsymbol{1}}{\boldsymbol{b}_r^T\boldsymbol{b}_r + \alpha_n}\right]_+ \\ &= \left[\widetilde{\boldsymbol{a}}_r + \frac{\left[\boldsymbol{X}_{(n)}\boldsymbol{B}^{(n)}\right]_{(:,r)} - \boldsymbol{A}^{(n)}\left[\left(\boldsymbol{B}^{(n)}\right)^T\boldsymbol{B}^{(n)}\right]_{(:,r)} - \beta_n\boldsymbol{1}}{\left[\left(\boldsymbol{B}^{(n)}\right)^T\boldsymbol{B}^{(n)}\right]_{(r,r)} + \alpha_n}\right]_+\end{aligned} \tag{18}$$

We implement the above procedures using the inexact scheme. We use PROX-iHALS to denote the inexact hierarchical alternating least squares method for solving the sparse NCP with the proximal algorithm. The PROX-iHALS is illustrated in Algorithm 3.

Algorithm 3: PROX-iHALS for sparse NCP

```
   Input  : $\mathcal{X}$, $R$, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$
   Output: $\boldsymbol{A}^{(n)}, n = 1, \ldots, N$
1  Initialize $\boldsymbol{A}^{(n)} \in \mathbb{R}^{I_n\times R}$, $n = 1, \ldots, N$, using nonnegative
    random numbers;
2  repeat
3      for $n = 1$ to $N$ do
4          Make mode-$n$ unfolding of $\mathcal{X}$ as $\boldsymbol{X}_{(n)}$ and compute
            MTTKRP $\boldsymbol{X}_{(n)}\boldsymbol{B}^{(n)}$;
5          Compute $\left(\boldsymbol{B}^{(n)}\right)^T\boldsymbol{B}^{(n)}$ based on (5);
6          repeat
7              for $r = 1$ to $R$ do
8                  Update $\boldsymbol{a}_r^{(n)}$ using (18);
9              end
10         until finite number of inner iterations is reached;
11     end
12 until outer termination criterion is reached;
13 return $\boldsymbol{A}^{(n)}, n = 1, \ldots, N$.
```

4.3 Stopping conditions

4.3.1 Stopping condition for outer loop

We terminate the outer loop according to the change of relative error during iteration. Relative error is related to data fitting. In the kth outer iteration, the relative error [48] of tensor decomposition is defined by

$$\text{RelErr}_k = \frac{\|\mathcal{X} - [\![\boldsymbol{A}_k^{(1)}, \ldots, \boldsymbol{A}_k^{(N)}]\!]\|_F}{\|\mathcal{X}\|_F}. \tag{19}$$

Based on the relative error, we terminate the outer loop using the following stopping condition

$$|\text{RelErr}_{k-1} - \text{RelErr}_k| < \varepsilon. \tag{20}$$

The threshold of ε can be set by a very small positive value, such as $1e-8$.

In addition, we also set a maximum running time for the outer loop.

4.3.2 Stopping condition for inner loop

In the lth inner iteration, we define the relative residual of the nth factor matrix $\boldsymbol{A}^{(n)}$ as

$$r_l^{(n)} = \frac{\|\boldsymbol{A}_l^{(n)} - \boldsymbol{A}_{l-1}^{(n)}\|_F}{\|\boldsymbol{A}_l^{(n)}\|_F}. \tag{21}$$

For the PROX-iHALS, we terminate the inner loop by the stopping condition of $r_l^{(n)} < \delta^{(n)}$, where $\delta^{(n)}$ is a dynamic positive threshold. If there is only one iteration in the inner loop, we update $\delta^{(n)}$ by $\delta^{(n)} = \delta^{(n)}/10$. We set the initial value by $\delta^{(n)} = 0.01$. For the PROX-ANQP, the inner loop is terminated according to the columns in the feasible region of the BPP algorithm [22].

Since we employ the inexact BCD framework, we also set a maximum number of inner iterations (`MAX_INNER_ITER`) to terminate the inner loop.

We summarize the stopping conditions for both of the outer and inner loop in Algorithm 4.

Algorithm 4: The stopping conditions

```
1  Set $\delta^{(n)} = 0.01, n = 1, \ldots, N$;
2  for $k = 1, 2, \ldots$ do
       % The outer loop starts here.
3      for $n = 1$ to $N$ do
4          for $l = 1, 2, \ldots$ do
               % The inner loop starts here.
5              $A^{(n)} = \underset{A^{(n)} \geqslant 0}{\operatorname{argmin}} \mathscr{F}(A^{(n)})$;
6              if $r_l^{(n)} < \delta^{(n)}$ or $l \geq$ MAX_INNER_ITER then
7                  Terminate the inner loop;
8              end
               % The inner loop ends here.
9          end
10         if $l == 1$ then
11             $\delta^{(n)} = \delta^{(n)}/10$;
12         end
13     end
14     if $|\mathrm{RelErr}_{k-1} - \mathrm{RelErr}_k| < \varepsilon$ then
15         Terminate the outer loop.
16     end
       % The outer loop ends here.
17 end
```

4.4 Remarks on convergence

The PROX-ANQP and PROX-iHALS methods in the inexact BCD framework have outstanding convergence properties. In Sect. 3.3, we have mentioned that proposed sparse NCP using the proximal algorithm and inexact BCD scheme can guarantee to converge to a stationary point. The subproblem with the proximal algorithm in (9) is strongly convex, which can yield a unique minimum [4]. Furthermore, the optimization methods of ANQP and iHALS can stably decrease the subproblem. According to the Proposition 3.7.1 in [4], both the PROX-ANQP and PROX-iHALS can converge to stationary points.

5 Experiments and results

We carried out the experiments on synthetic, real-world, dense, sparse, small-scale, and large-scale tensors. We compared the proposed **PROX-ANQP** and **PROX-iHALS** methods with three sparse NCP methods listed below.

- *AO-ADMM*: This is the sparse NCP method using AO-ADMM algorithm [19], which includes multiple inner iterations. The l_1-norm is handled by a proximal operator.
- *iAPG*: We extend the APG method in sparse Tucker decomposition [47] to the sparse NCP problem in (6). In order to make a fair comparison, we implement APG in the inexact scheme using multiple inner iterations, which is abbreviated as iAPG [42]. The l_1-norm is handled by a proximal operator.
- *iMU*: This is the sparse NCP method using the classical MU algorithm [9]. We implement MU in the inexact scheme using multiple inner iterations, which is abbreviated as iMU.

The above three methods can be directly applied to solve the sparse NCP in (6). The l_1-norm can be handled by the proximal operator in AO-ADMM and iAPG. Due to the proximal operator, AO-ADMM and iAPG do not suffer from the rank deficiency. Using the multiplicative updating rule, MU does not suffer from the rank deficiency.

In Table 1, we summarized the computational complexity of all the sparse NCP methods. Only the multiplicative operations were counted for mode-n in one outer iteration. The main time cost of these algorithms was spent on the calculation of MTTKRP $\boldsymbol{X}_{(n)}\boldsymbol{B}^{(n)}$, which consists of two parts: Khatri-Rao product $\boldsymbol{B}^{(n)}$ and matrix product of $\boldsymbol{X}_{(n)}$ and $\boldsymbol{B}^{(n)}$. The computational complexity of $\boldsymbol{B}^{(n)}$ reaches $R\prod_{\tilde{n}=1,\tilde{n}\neq n}^{N} I_{\tilde{n}}$ and that of $\boldsymbol{X}_{(n)}\boldsymbol{B}^{(n)}$ reaches $R\prod_{n=1}^{N} I_n$. Item $\left(\boldsymbol{B}^{(n)}\right)^T\boldsymbol{B}^{(n)}$ can be calculated efficiently by (5), whose complexity is $R^2\sum_{\tilde{n}=1,\tilde{n}\neq n}^{N} I_{\tilde{n}}$. For the inner loop of the subproblem, $\bar{K}$ is assumed to be the average iteration number. In Table 1, we can find that the complexity of these algorithms is highly comparable to each other. It can be inferred that the time of convergence is highly related to the number of iterations.

Many experimental parameters and settings will affect the performances of a sparse NCP method. Since our purpose in the experiments is only to test the ability to impose sparsity, we fix the following settings for all methods.

- Initialization. For PROX-ANQP, PROX-iHALS, AO-ADMM and iAPG, all factor matrices were initialized using nonnegative random numbers by MATLAB function `max(0,randn(`I_n, R`))`. Only the iMU was

Table 1 Computational Complexity of Multiplicative Operations for Subproblem (9)

Method	$\boldsymbol{X}_{(n)}\boldsymbol{B}^{(n)}$	$\left(\boldsymbol{B}^{(n)}\right)^T\boldsymbol{B}^{(n)}$	Inner loop
PROX-ANQP			$\bar{K}(I_nR^2+R^3)$
PROX-iHALS			$\bar{K}I_nR^2$
AO-ADMM	$R\prod_{\tilde{n}=1,\tilde{n}\neq n}^{N} I_{\tilde{n}}$	$R^2\sum_{\tilde{n}=1,\tilde{n}\neq n}^{N} I_{\tilde{n}}$	$\bar{K}(I_nR^2+R^3)$
iAPG	$+R\prod_{\tilde{n}=1}^{N} I_{\tilde{n}}$		$\bar{K}I_nR^2$
iMU			$\bar{K}I_nR^2$

$\bar{K}$ is assumed to be the average inner iteration number initialized by max(0,randn(I_n, R))+0.1. All initialized factors were scaled by $A_0^{(n)} = \frac{A_0^{(n)}}{||A_0^{(n)}||_F} \times \sqrt[N]{||\mathcal{X}||_F}$.

- The factor updating order was fixed by $1, 2, \ldots, N$.
- The maximum inner iteration `MAX_INNER_ITER` was fixed by 5 according to the default setting in AO-ADMM [19].
- For the PROX-ANQP and PROX-iHALS method, the proximal regularization parameter α_n was fixed by 1e-4 [45].

The l_1-norm regularization parameters of $\beta_n, n = 1, \ldots, N$, in sparse NCP are the key elements to impose sparsity, which are the most crucial testing parameters in the experiments. We selected a sequence of β_n values in ascending order for each tensor by manual testing. For synthetic tensors, we stop the increase of β_n when the true sparse components are recovered, while for real-world tensors, we stop the increase of β_n when the number of nonzero components is reduced to less than half of the initial number. In order to make it convenient to select and test the parameters, we kept $\beta_n, n = 1, \ldots, N$, the same in all modes of the tensor. After choosing the β_n, we calculated and evaluated the sparsity level [44] of the factor matrices by

$$\text{Sparsity}_{A^{(n)}} = \frac{\#\{A_{i,r}^{(n)} < T_s\}}{I_n \times R}, \tag{22}$$

where T_s is a small positive number and $\#\{\cdot\}$ denotes the number of elements that are smaller than the threshold T_s in factor matrix $A^{(n)}$.

In the synthetic tensor experiments, we used prior sparse matrices to construct the data. After decomposition, the accuracy of the recovered sparse signals should be evaluated. Let $S^{(n)} = [s_1, \ldots, s_R] \in \mathbb{R}^{L \times R}$ denote the mode-n prior sparse matrix, where R is the real number of components and L is the length of a component. Let $T^{(n)} = [t_1, \ldots, t_{\tilde{R}}] \in \mathbb{R}^{L \times \tilde{R}}$ represent the mode-n estimated sparse matrix, where the $\tilde{R}$ is the estimated number of nonzero components. We evaluate the accuracy of the estimated matrix $T^{(n)}$ compared with original sparse signals $S^{(n)}$ by peak-signal-to-noise ratio (PSNR, see Chapter 3 in [9])

$$\text{PSNR} = \frac{1}{\tilde{R}} \sum_{r=1}^{\tilde{R}} 10\log_{10} \frac{L}{||\hat{t}_r - \hat{s}_c||_2^2}, \tag{23}$$

where $\hat{t}_r$ is the rth normalized estimated sparse signal, and $\hat{s}_c$ is the normalized reference sparse signal. $\hat{s}_c$ comes from $S^{(n)}$, which has the highest correlation coefficient with $\hat{t}_r$.

All the experiments were conducted on the computer with Intel Core i5-4590 3.30 GHz CPU, 8 GB memory, 64-bit Windows 10 and MATLAB R2016b. The fundamental tensor computation was based on Tensor Toolbox 2.6 [3]. The codes are available on the author's website http://deqing.me/.

5.1 Synthetic tensor data

5.1.1 Size $1000 \times 100 \times 100 \times 5$ with one sparse factor

In this experiment, we constructed a synthetic fourth-order tensor by 10 channels of simulated sparse and nonnegative signals, as shown in Fig. 1a. The signals come from the file `VSparse_rand_10.mat` in NMFLAB [8]. There are 1000 points in each channel, so the sparse signal matrix is

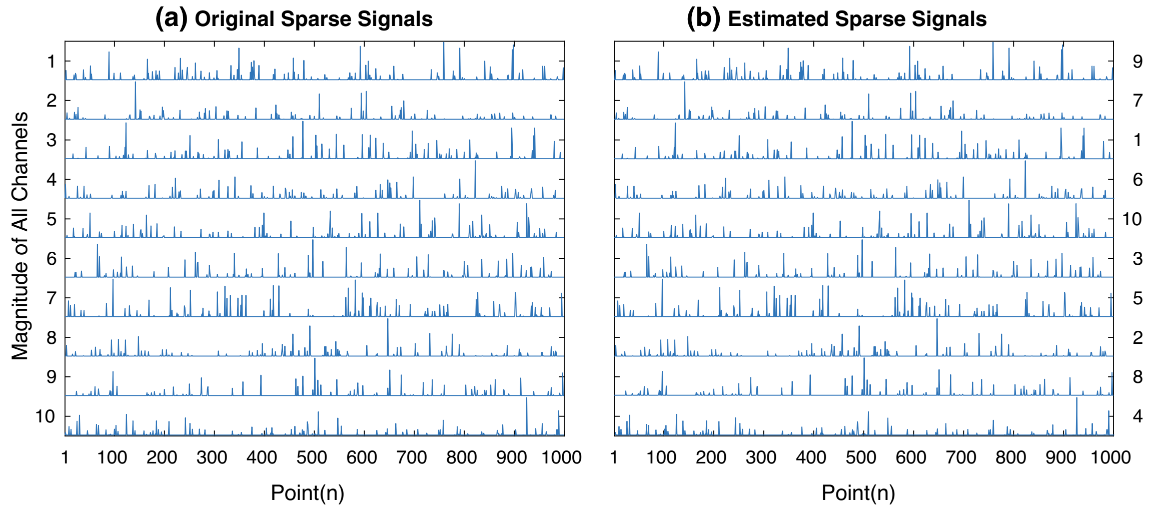

Fig. 1 Sparse and nonnegative signals used in synthetic tensor. **a** shows the original ten channels of signals. **b** shows the estimated ten channels of signals from the synthetic tensor $\mathcal{X}_{\text{SYN1}}$ by sparse NCP based on the PROX-ANQP method with $\beta_n = 5$. The PSNR is 90.2698 according to (23)

Table 2 Comparison of Sparse NCPs on $\mathcal{X}_{\text{SYN1}} \in \mathbb{R}^{1000\times100\times100\times5}$

Method	β_n	Obj	RelErr	Time(s)	Iter	NNC	Spars_1	PSNR
PROX-ANQP	0	1.51e+02	0.0082	69.2	33.5	20.00	0.394	74.66
	1	6.90e+03	0.0088	548.3	275.5	10.00	0.787	84.76
	2	8.75e+03	0.0089	803.2	403.1	10.00	0.810	87.25
	3	9.90e+03	0.0089	907.4	456.2	10.00	0.837	88.23
	4	1.15e+04	0.0089	959.3	480.3	**10.00**	**0.852**	**88.67**
	5	1.28e+04	0.0089	991.6	496.0	**10.00**	**0.855**	**88.79**
PROX-iHALS	0	1.51e+02	0.0082	103.0	51.4	20.00	0.474	79.34
	1	8.16e+03	0.0090	394.8	198.3	11.63	0.800	80.59
	2	9.64e+03	0.0090	656.4	327.9	10.63	0.824	84.05
	3	1.03e+04	0.0090	868.8	436.8	10.23	0.845	87.14
	4	1.20e+04	0.0090	859.2	431.9	**10.27**	**0.861**	**86.93**
	5	1.31e+04	0.0090	947.1	474.7	**10.17**	**0.864**	**87.66**
AO-ADMM	0	1.59e+02	0.0084	254.5	124.8	20.00	0.151	65.74
	1	7.41e+03	0.0089	357.7	179.4	12.33	0.782	80.64
	2	8.42e+03	0.0089	801.9	401.8	11.10	0.819	87.93
	3	1.00e+04	0.0089	855.2	427.6	10.63	0.854	88.22
	4	1.17e+04	0.0090	908.3	455.9	10.57	0.869	88.45
	5	1.34e+04	0.0090	876.7	438.8	10.63	**0.871**	**87.78**
iAPG	0	1.56e+02	0.0084	140.9	70.6	20.00	0.122	63.19
	1	6.67e+03	0.0088	354.5	177.9	10.37	0.764	86.72
	2	8.04e+03	0.0088	681.6	339.0	10.17	0.791	88.57
	3	9.72e+03	0.0089	781.7	391.2	10.07	0.828	89.26
	4	1.21e+04	0.0090	792.5	394.9	**10.27**	**0.849**	**88.98**
	5	1.33e+04	0.0089	871.7	433.8	**10.00**	**0.856**	**89.33**
iMU	0	1.58e+02	0.0084	712.0	352.5	20.00	0.210	66.64
	1	6.30e+03	0.0089	862.2	427.4	12.90	0.693	75.63
	2	8.62e+03	0.0091	868.9	430.5	11.53	0.788	79.04
	3	1.06e+04	0.0093	836.4	411.2	11.10	0.847	81.37
	4	1.26e+04	0.0094	822.8	403.5	11.23	0.864	81.00
	5	1.39e+04	0.0094	838.4	408.2	10.77	**0.882**	82.32

Spars_1 = Sparsity level of the first factor

Ground truth: Spars_1= 0.9

NNC = Number of nonzero components

$\boldsymbol{S}^{(1)} = [\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{10}] \in \mathbb{R}^{1000\times10}$. Three uniformly distributed random matrices $\boldsymbol{A}^{(2)}, \boldsymbol{A}^{(3)} \in \mathbb{R}^{100\times10}$ and $\boldsymbol{A}^{(4)} \in \mathbb{R}^{5\times10}$ were employed as mixing matrices, which were generated by `rand` function in MATLAB. Afterwards, we synthesized a third-order tensor by

$$\mathcal{X}_{\text{SNY1}} = [\![\boldsymbol{S}^{(1)}, \boldsymbol{A}^{(2)}, \boldsymbol{A}^{(3)}, \boldsymbol{A}^{(4)}]\!] \in \mathbb{R}^{1000\times100\times100\times5}$$

. Next, nonnegative noise was added to the tensor with SNR of 40dB, which was generated by code max(0,randn(size($\mathcal{X}$))).

For all sparse NCP methods, we set $\varepsilon = 1e-8$ as the threshold of outer stopping condition in (20). We set $T_s = 1e-3$ in (22). The maximum running time was set by 180 seconds. We selected 20 as the number of components for tensor decomposition[1]. The reason is that we intend to recover the ten channels of the true signal just by imposing sparse regularization during decomposition, even though the exact optimal number of components is unknown. We selected the values of $\beta_n = 0, 0.1, 0.5, 1, 2, 3$ for all the optimization methods to evaluate their abilities to impose sparsity. The selection of sparse regularization parameters depends on the tensor data. After tensor decomposition, the values of objective function value (Obj), relative error (RelErr), running time in seconds (in wall-clock time),

[1] Since ten channels of signals are mixed in the tensor, it is natural to select 10 as the optimal component number. The number of components might also be estimated by some classical methods, such as DIFFIT [38]. However, we selected 20 to test the performances of sparse regularization.

iteration number (Iter), the number of nonzero components (NNC), sparsity level (Spars) and PSNR of the estimated signal factor matrix were recorded as the performance evaluation criteria. For all optimization methods with each β_n, the sparse NCP was run 30 times, and the average values of all criteria were computed. The results are shown in Table 2. We emphasized the outstanding performances of the sparse NCP algorithms in bold in the tables.

From Table 2, it can be found that all methods are able to impose sparsity with proper sparse regularization parameter β_n. When β_n increases, the sparsity level of the mode-1 factor matrix also increases. After properly tuning the sparse regularization parameter β_n, weak components will be removed (set to 0), weak elements in strong components will be prohibited, and the true ten channels of sparse signals will be recovered.

When β_n increases to a proper value, the PSNR increases significantly. In this experiment, the higher the value of PSNR is, the better an algorithm performs to recover original sparse components. In Table 2, it is clear to see that PROX-ANQP, PROX-iHALS and iAPG have higher PSNR with larger sparse regularization parameters, for example $\beta_n = 4, 5$. This means that these three methods recover the ten channels of sparse signals more precisely. One of the recovered sparse signal matrix from $\mathcal{X}_{\mathrm{SYN1}}$ by PROX-ANQP is shown in Fig. 1b.

For the synthetic data, the objective function values and relative errors are very similar with the same β_n value. The convergence speed can be concluded from Table 2. iMU performs slowly compared with other methods. AO-ADMM performs slowly with $\beta_n = 0$, but it becomes fast with $\beta_n > 0$. All PROX-ANQP, PROX-iHALS and iAPG methods perform very well. It can also be concluded from Table 2 that the running time is highly related to the number of outer iterations.

5.1.2 Size 500 × 500 × 500 with two sparse factors

For this third-order tensor, the factor matrices were generated using the following codes.

Factor	Code	Zeros (%)
$\boldsymbol{S}^{(1)} \in \mathbb{R}^{500\times100}$	`max(0,rand(500,100)*10-9);`	90
$\boldsymbol{S}^{(2)} \in \mathbb{R}^{500\times100}$	`max(0,rand(500,100)*2-1);`	50
$\boldsymbol{A}^{(3)} \in \mathbb{R}^{500\times100}$	`rand(500,100);`	0

Afterwards, a third-order tensor was synthesized by $\mathcal{X}_{\mathrm{SNY2}} = [\![\boldsymbol{S}^{(1)}, \boldsymbol{S}^{(2)}, \boldsymbol{A}^{(3)}]\!]$, whose true number of components was 100 (rank = 100). Noise with SNR of 40dB was added.

Table 3 Comparison of Sparse NCPs on $\mathcal{X}_{\mathrm{SYN2}} \in \mathbb{R}^{500\times500\times500}$

Method	β_n	Time(s)	Iter	NNC	Spars_1	Spars_2	Spars_3
PROX-ANQP	0	42.0	19.0	200.00	0.300	0.512	0.027
	0.1	51.6	25.5	100.93	0.705	0.749	0.502
	0.5	49.9	25.6	100.27	0.828	0.750	0.505
	1	173.6	97.6	**100.00**	**0.872**	**0.749**	**0.506**
	3	241.3	137.5	**100.00**	**0.933**	**0.725**	**0.505**
	5	282.7	158.9	**100.00**	**0.945**	**0.682**	**0.505**
PROX-iHALS	0	56.9	32.4	200.00	0.438	0.506	0.014
	0.1	80.5	46.3	159.07	0.615	0.623	0.218
	0.5	114.5	65.6	123.13	0.746	0.702	0.393
	1	131.1	75.2	110.47	0.790	0.728	0.454
	3	305.1	174.9	**100.23**	**0.863**	**0.735**	**0.503**
	5	345.3	195.4	**100.03**	**0.891**	**0.706**	**0.504**
AO-ADMM	0	141.5	80.7	200.00	0.344	0.482	0.086
	0.1	125.8	72.7	195.77	0.613	0.471	0.078
	0.5	138.7	79.6	128.93	0.821	0.522	0.064
	1	125.3	73.7	110.53	0.875	0.547	0.054
	3	165.9	96.7	100.37	0.939	0.646	0.138
	5	247.3	144.0	**100.00**	**0.948**	0.690	0.359
iAPG	0	121.7	72.7	200.00	0.156	0.495	0.342
	0.1	122.0	72.1	200.00	0.566	0.552	0.372
	0.5	140.4	83.2	151.77	0.706	0.621	0.444
	1	145.2	85.2	107.80	0.719	0.695	0.466
	3	288.9	171.3	**100.07**	**0.766**	**0.722**	**0.504**
	5	440.1	263.5	**100.00**	**0.823**	**0.699**	**0.505**
iMU	0	600	356.8	200.00	0.330	0.508	0.011
	0.1	595.3	353.6	199.63	0.374	0.506	0.012
	0.5	600	332.9	149.07	0.524	0.638	0.265
	1	600	345.1	146.27	0.535	0.635	0.280
	3	595.2	345.1	127.97	0.626	0.672	0.369
	5	593.7	340.5	118.03	0.672	0.686	0.418

Ground truth levels: Spars_1= 0.95, Spars_2= 0.75 and Spars_3= 0.5

Spars_n = Sparsity level of the mode-n estimated factor

NNC = Number of nonzero components

We set the outer stopping condition by $\varepsilon = 1e - 6$ and the maximum running time by 600 seconds. 200 was selected as the initial number of components. The average performances of all sparse NCP methods after 30 runs were computed. We only show running time in seconds, iteration number (Iter), number of nonzero components (NNC) and the sparsity level (Spars) of all estimated factors in Table 3.

Table 3 shows that all methods are able to impose sparsity on all factors matrices. PROX-ANQP, PROX-iHALS, and iAPG methods perform very well to extract the true 100 sparse components. Interestingly, the sparsity levels of all the extracted factor matrix by PROX-ANQP, PROX-iHALS, and iAPG methods are also very close to

Table 4 Comparison of Sparse NCPs on Ongoing EEG Tensor $\mathcal{X}_{\mathrm{EEG}} \in \mathbb{R}^{64\times146\times510}$

Method	β_n	Obj	RelErr	Time(s)	Iter	NNC	Spars$_1$	Spars$_2$	Spars$_3$
PROX-	0	2.98e+10	0.2847	**19.2**	444.4	40.00	0.085	0.381	0.104
ANQP	1e+5	3.35e+10	0.2858	**34.9**	824.1	39.73	0.124	0.475	0.129
	5e+5	4.59e+10	0.2957	**21.4**	540.4	37.30	0.241	0.779	0.292
	10e+5	5.86e+10	0.3145	**17.6**	483.9	30.53	0.383	0.872	0.495
	15e+5	6.79e+10	0.3342	**15.5**	448.4	23.57	0.530	0.904	0.633
	20e+5	7.46e+10	0.3515	**10.5**	310.6	**17.57**	**0.638**	**0.927**	**0.723**
PROX-	0	2.98e+10	0.2848	**14.2**	466.9	40.00	0.086	0.376	0.103
iHALS	1e+5	3.34e+10	0.2852	**18.6**	601.6	39.97	0.119	0.469	0.122
	5e+5	4.58e+10	0.2952	**15.1**	488.4	37.47	0.238	0.778	0.293
	10e+5	5.86e+10	0.3158	**13.1**	424.2	29.83	0.399	0.872	0.503
	15e+5	6.79e+10	0.3343	**15.5**	456.7	23.40	0.533	0.906	0.631
	20e+5	7.47e+10	0.3514	**10.8**	355.7	**17.53**	**0.637**	**0.927**	**0.720**
AO-	0	2.98e+10	0.2849	**23.0**	810.4	40.00	0.081	0.385	0.102
ADMM	1e+5	3.34e+10	0.2853	**20.2**	685.2	39.93	0.124	0.460	0.124
	5e+5	4.60e+10	0.2978	**16.0**	546.6	35.93	0.271	0.775	0.326
	10e+5	5.84e+10	0.3218	**15.3**	523.7	26.23	0.468	0.860	0.547
	15e+5	6.75e+10	0.3394	**14.2**	486.6	20.93	0.582	0.900	0.662
	20e+5	7.43e+10	0.3552	**9.9**	339.5	**15.93**	**0.669**	**0.925**	**0.738**
iAPG	0	2.99e+10	0.2851	**9.5**	301.4	40.00	0.084	0.370	0.101
	1e+5	3.34e+10	0.2857	**21.4**	710.9	39.80	0.132	0.475	0.127
	5e+5	4.73e+10	0.3117	**18.4**	621.0	28.37	0.426	0.783	0.445
	10e+5	5.92e+10	0.3378	**19.5**	655.6	19.63	0.598	0.865	0.631
	15e+5	6.76e+10	0.3579	**15.8**	532.8	14.13	0.704	0.904	0.736
	20e+5	7.42e+10	0.3681	**12.9**	436.7	**12.20**	**0.745**	**0.924**	**0.781**
iMU	0	2.99e+10	0.2853	106.3	2790.8	40.00	0.077	0.380	0.115
	1e+5	3.34e+10	0.2857	101.7	2708.2	40.00	0.104	0.457	0.137
	5e+5	4.58e+10	0.2928	89.7	1269.9	39.77	0.191	0.761	0.272
	10e+5	5.87e+10	0.3132	103.0	1034.6	31.70	0.364	0.860	0.493
	15e+5	6.79e+10	0.3339	94.5	946.6	23.97	0.525	0.901	0.634
	20e+5	7.48e+10	0.3505	81.5	883.3	**18.10**	**0.635**	**0.922**	**0.720**

Spars$_1$, Spars$_2$ and Spars$_3$ are the sparsity levels of spatial, spectral and temporal factor
NNC = Number of nonzero components

the ground-truth[2] values with some β_n. iMU and AO-ADMM do not always work well to reach the ground-truth factor sparsity levels. Moreover, iMU shows slow convergence compared with other methods.

5.2 Third-order dense tensor data

In this experiment, we used a real-world third-order ongoing EEG tensor. The original data were collected from fourteen right-handed and healthy subjects of adults in a music listening experiment. The music stimulus was a piece of 8.5 minutes of modern tango Adiós Nonino by the composer Astor Piazzolla. Short-Time Fourier Transform (STFT) with the window size of 3 seconds and the hop size of 1 second was used to transform the data of each subject into a third-order tensor. Details of data collection and preprocessing can be found in [43, 44]. We only employ the tensor data of one subject in this experiment[3]. The size of this tensor is $channel \times frequency \times time = 64 \times 146 \times 510$, in which 64 channel points represent 64 electrodes on the scalp, 146 frequency points represent the spectrum in 1-30Hz, and 510 time points represent the duration of a stimulus of about 8.5 minutes. Since the spectra from EEG tensor are usually sparse, we wish to recover the sparse spectral components by sparse regularization.

[2] Since we use double number of true sparse components as the initial number of tensor decomposition, the ground truth sparsity of the factor matrix is computed by $(x\% + 1)/2$. $x\%$ is the percentage of zeros in a simulated matrix.

[3] https://github.com/wangdeqing/Ongoing_EEG_Tensor_Decomposition.

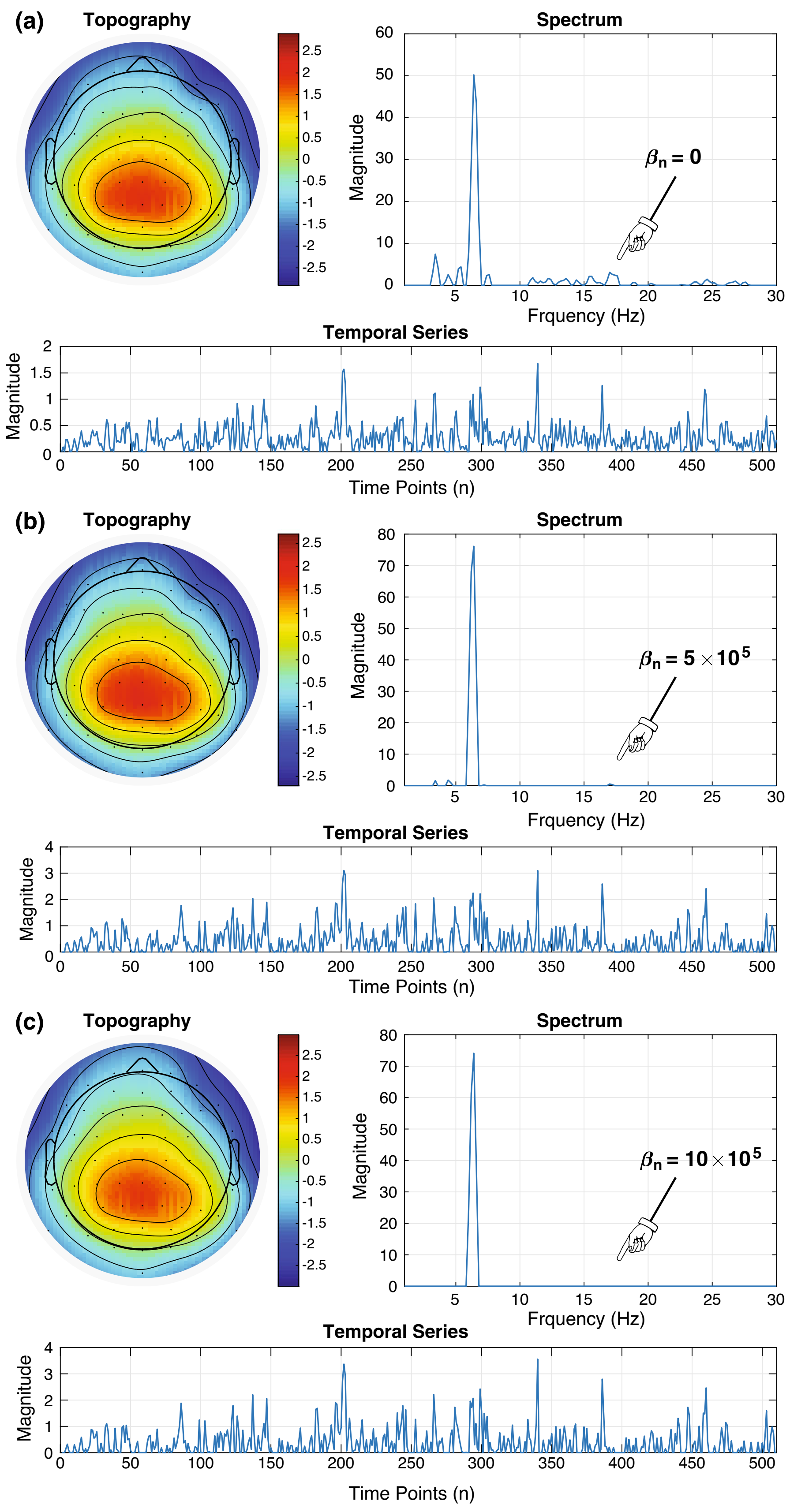

Fig. 2 Selected groups of components from the ongoing EEG tensor using the PROX-ANQP algorithm. All groups reveal the same brain activity. In the decomposed EEG data, the spatial component is topography, the spectral component is the spectrum, and the temporal component is the energy evolution series. The components in **a** were extracted with $\beta_n = 0$, **b** with $\beta_n = 5 \times 10^5$ and **c** with $\beta_n = 10 \times 10^5$

Table 5 Comparison of Sparse NCPs on Facebook Wall Posts Tensor $\mathcal{X}_{\mathrm{Sp1}} \in \mathbb{R}^{46952\times 46951\times 1952}$

Method	β_n	Obj	RelErr	Time(s)	Iter	NNC	Spars$_1$	Spars$_2$	Spars$_3$
PROX-	0	6.48e+05	0.9510	**535.4**	42.8	100.00	0.997	0.989	0.798
ANQP	0.01	6.49e+05	0.9519	**379.1**	39.8	99.33	0.998	≈ 1	0.874
	0.05	6.70e+05	0.9668	**400.8**	42.6	59.40	0.999	≈ 1	0.931
	0.1	6.90e+05	0.9814	**213.7**	22.6	**18.53**	$\approx \boldsymbol{1}$	$\approx \boldsymbol{1}$	**0.975**
PROX-	0	6.48e+05	0.9513	**512.3**	51.7	100.00	0.997	0.990	0.803
iHALS	0.01	6.50e+05	0.9525	**487.6**	48.4	99.87	0.998	≈ 1	0.874
	0.05	6.69e+05	0.9664	**552.0**	54.7	63.90	0.999	≈ 1	0.930
	0.1	6.89e+05	0.9808	**217.4**	21.5	**20.27**	$\approx \boldsymbol{1}$	$\approx \boldsymbol{1}$	**0.974**
AO-	0	6.48e+05	0.9514	690.7	71.6	100.00	0.997	0.989	0.803
ADMM	0.01	6.50e+05	0.9522	684.5	69.9	99.97	0.998	≈ 1	0.873
	0.05	6.74e+05	0.9698	620.7	63.3	56.33	≈ 1	≈ 1	0.938
	0.1	6.93e+05	0.9835	239.4	24.5	**15.50**	$\approx \boldsymbol{1}$	$\approx \boldsymbol{1}$	**0.979**
iAPG	0	6.49e+05	0.9520	536.2	58.9	100.00	0.997	0.991	0.797
	0.01	6.51e+05	0.9533	533.8	57.5	99.83	0.998	≈ 1	0.874
	0.05	6.74e+05	0.9699	548.1	59.2	52.03	≈ 1	≈ 1	0.940
	0.1	6.93e+05	0.9831	331.6	35.7	**15.53**	$\approx \boldsymbol{1}$	$\approx \boldsymbol{1}$	**0.979**
iMU	0	6.49e+05	0.9521	497.2	58.5	100.00	0.998	0.989	0.800
	0.01	6.53e+05	0.9544	592.7	64.3	100.00	0.999	≈ 1	0.872
	0.05	6.84e+05	0.9767	676.7	74.6	39.87	≈ 1	≈ 1	0.957
	0.1	7.01e+05	0.9893	273.7	30.2	**7.07**	$\approx \boldsymbol{1}$	$\approx \boldsymbol{1}$	**0.990**

Spars$_n$ = Sparsity level of the mode-n estimated factor

Spars$_n$ ≈ 1 means that the factor is very close to a zero matrix

NNC = Number of nonzero components

We set $\varepsilon = 1e-8$ in (20) and $T_s = 1e-6$ in (22). The maximum running time was set by 120 seconds. The initial number of components was set by 40 according to previous studies [11, 44]. We tested the values of $\beta_n = 0, 1e5, 5e5, 10e5, 15e5, 20e5$ all methods. All methods were run 30 times. The averages of performance criteria are recorded in Table 4. The results show that all methods are effective to impose sparsity with β_n. The iMU is slower than the other four methods.

We selected three groups of extracted components using the PROX-ANQP method with $\beta_n = 0, 5e5, 10e5$, respectively, as shown in Fig. 2. These three groups show the same brain activity. It is obvious to see that the spectra become sparser and sparser when the sparse regularization parameter increases. With $\beta_n = 5e5, 10e5$, more and more unnecessary elements are removed in the spectra, and only the most prominent frequency band is retained. Figure 2 demonstrates that our methods are effective to extract meaningful sparse components that are related to some brain activities.

5.3 Third-order large-scale sparse tensor data

In this experiment, we tested the sparse NCP algorithms on a third-order large-scale sparse social network tensor. The data contain Facebook wall posts[4] information among 46, 952 users in 1952 days [41]. The size of this sparse tensor $\mathcal{X}_{\mathrm{Sp1}}$ is 46,952 $\times 46,951$ $\times$1952, and the number of non-zero elements is 737, 928.

We set the outer stopping condition by $\varepsilon = 1e-6$ in (20) and the sparsity threshold by $T_s = 1e-6$ in (22). The maximum running time was set by 1200 seconds, and the initial number of components was set by 100. We tested the values of $\beta_n = 0, 0.01, 0.05, 0.1$ for all methods. The average values of performance criteria after 30 runs are recorded in Table 5. We recorded the objective function values of all methods within the first 600 seconds in Fig. 3.

5.4 Fourth-order large-scale sparse tensor data

In this experiment, we evaluated the sparse NCP algorithms on a fourth-order large-scale sparse tensor of NIPS publications[5] [16]. The size of this sparse tensor $\mathcal{X}_{\mathrm{Sp2}}$ is $2482 \times 2862 \times 14{,}036 \times 17$. The values represent 2482 papers, 2862 authors, 14,036 words and 17 years. There are 3,101,609 nonzero elements.

[4] http://konect.cc/networks/facebook-wosn-wall/.

[5] http://frostt.io/tensors/nips/.

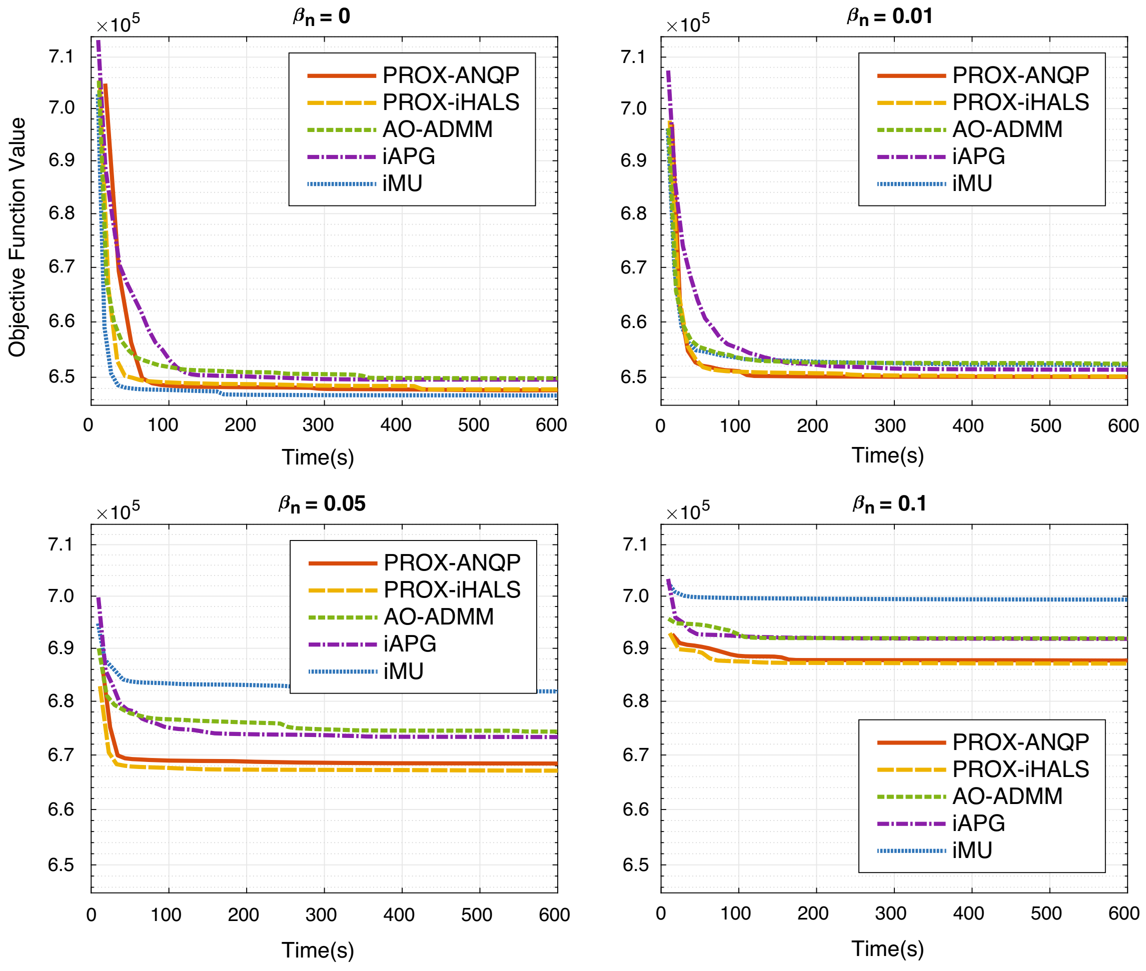

Fig. 3 The Objective Function Value Curves of Sparse NCPs on Third-order Facebook Wall Posts Tensor

The maximum running time was set by 1800 seconds. Other settings are the same as the previous third-order case. We tested the values of $\beta_n = 0, 0.1, 0.3, 0.5$ for all methods. The average values of performance criteria after 30 runs are recorded in Table 6. Figure 4 illustrates the objective function values of all methods within the first 1200 seconds.

The experimental results of both the third-order and fourth-order large-scale sparse tensor demonstrate that all our algorithms are effective to impose sparsity on the factor matrices. From Tables 5 and 6, it is clear to see that, with sparse regularization parameter β_n increasing, the number of nonzero components (NNC) decreases gradually. The extracted factor matrices from the sparse tensors are extremely sparse, even though no additional sparse regularization is imposed ($\beta = 0$). Nevertheless, our algorithms with $\beta > 0$ can further increase the sparsity level of the factor matrices.

From the perspective of convergence speed, both the PROX-ANQP and PROX-iHALS methods run fast for the large-scale sparse tensors with different β_n values compared with other methods, which can be concluded from Tables 5, 6, Figs. 3 and 4. However, AO-ADMM and iAPG perform slowly for the large-scale sparse tensors. iMU has fast convergence speed, but it has higher objective function value with large β_n values (e.g., β_n = 0.05, 0.1 in the third-order case, and β_n = 0.3, 0.5 in the fourth-order case). The reason is that, with the same β_n value, iMU yields fewer nonzero components compared with other methods.

6 Discussion

We have proposed a novel sparse NCP model using the proximal algorithm and inexact scheme. The model can be efficiently solved by two algorithms of the PROX-ANQP and PROX-iHALS. In order to test the performance of the algorithms, we conducted experiments in different cases, including synthetic and real-world tensors, third-order and fourth-order tensors, dense and sparse tensors, and small-scale and large-scale tensors. Three state-of-the-art sparse

Table 6 Comparison of Sparse NCPs on NIPS Publications Tensor $\boldsymbol{\mathcal{X}}_{\text{Sp2}} \in \mathbb{R}^{2482\times2862\times14036\times17}$

Method	β_n	Obj	RelErr	Time(s)	Iter	NNC	Spars$_1$	Spars$_2$	Spars$_3$	Spars$_4$
PROX-	0	3.11e+07	0.9439	**1091.8**	18.5	100.00	0.999	0.998	0.920	0.941
ANQP	0.1	3.12e+07	0.9447	**1161.9**	19.9	99.57	0.999	0.998	0.925	0.941
	0.3	3.20e+07	0.9571	**990.5**	17.0	78.43	0.999	0.998	0.950	0.954
	0.5	3.38e+07	0.9841	**749.3**	12.9	**24.83**	$\approx$ ***1***	**0.999**	**0.985**	**0.985**
PROX-	0	3.11e+07	0.9439	**991.4**	17.2	100.00	0.999	0.997	0.920	0.941
iHALS	0.1	3.12e+07	0.9448	**1123.2**	19.3	99.93	0.999	0.998	0.925	0.941
	0.3	3.20e+07	0.9577	**1055.5**	18.0	79.60	0.999	0.998	0.948	0.953
	0.5	3.37e+07	0.9827	**804.5**	13.9	**28.50**	$\approx$ ***1***	**0.999**	**0.982**	**0.983**
AO-	0	3.11e+07	0.9440	1346.7	23.0	100.00	0.999	0.997	0.919	0.938
ADMM	0.1	3.11e+07	0.9446	1191.1	20.6	99.70	0.999	0.998	0.924	0.939
	0.3	3.23e+07	0.9611	1358.5	23.4	74.70	0.998	0.998	0.951	0.950
	0.5	3.40e+07	0.9865	912.5	15.5	**21.93**	**0.999**	**0.999**	**0.986**	**0.986**
iAPG	0	3.11e+07	0.9436	1263.9	21.9	100.00	0.999	0.997	0.920	0.941
	0.1	3.12e+07	0.9449	1325.0	22.8	99.47	0.999	0.998	0.925	0.940
	0.3	3.22e+07	0.9600	1587.2	27.1	73.37	0.999	0.998	0.951	0.949
	0.5	3.38e+07	0.9837	1131.5	19.5	**27.23**	$\approx$ ***1***	**0.999**	**0.982**	**0.979**
iMU	0	3.12e+07	0.9448	1126.3	19.4	100.00	0.999	0.998	0.921	0.941
	0.1	3.12e+07	0.9456	1197.4	20.5	99.67	0.999	0.998	0.927	0.941
	0.3	3.28e+07	0.9685	1317.3	22.5	60.07	0.999	0.999	0.964	0.965
	0.5	3.43e+07	0.9916	920.7	15.8	**12.73**	$\approx$ ***1***	$\approx$ ***1***	**0.992**	**0.992**

Spars$_n$ = Sparsity level of the mode-n estimated factor

Spars$_n$ ≈ 1 means that the factor is very close to a zero matrix

NNC = Number of nonzero components

NCP methods are also tested for comparison, including the AO-ADMM, iAPG and iMU. We have the following findings: (1) The PROX-ANQP and PROX-iHALS methods particularly have the fast convergence speed and excellent effects of imposing sparsity in all cases compared with other methods. The outstanding performances of the PROX-ANQP and PROX-iHALS are due to two points: the proximal algorithm that overcomes the rank deficiency; and the inexact scheme that increases the efficiency. (2) The iAPG method contains a proximal operator, which can handle the l_1-norm. With the proximal operator, iAPG does not suffer from the rank deficiency. In the experiments, we find that iAPG is very efficient to impose sparsity for small-scale dense tensor decomposition. Nevertheless, iAPG runs slowly for large-scale sparse tensor decomposition. (3) The iMU method converges very slowly for dense tensors, but it becomes fast for large-scale sparse tensor. For sparse tensor decomposition, the extracted factor matrices are extremely sparse already. Most elements in the factors are zeros. According to the multiplicative updating rule, once an element becomes zero, it will never change. This property might be the reason why iMU converges fast on the sparse tensor. (4) The AO-ADMM also contains a proximal operator that can handle l_1-norm and overcome the rank deficiency. However, AO-ADMM converges slowly compared with PROX-ANQP and PROX-iHALS in most cases, particularly in the large-scale sparse tensor case. Moreover, AO-ADMM is inferior to iAPG with $\beta_n = 0$ in many cases. In a word, our proposed PROX-ANQP and PROX-iHALS methods have the best performances for sparse NCP, and have very good generalization for the different types and scales of datasets.

In addition to the solving methods, another critical issue of sparse NCP is the selection of the sparse regularization parameter β_n. Firstly, we want to mention that one purpose of this paper is to demonstrate the effectiveness of the algorithms to impose sparsity. Therefore, in order to simplify the selection of parameters, we keep β_n the same for all factor matrices in sparse NCP. With the same β_n on all modes, the sparse NCP can still recover high sparse components and low sparse (or even dense) components in different factor matrices. In the future, it would be interesting to investigate how to separately control the sparsity levels of different factor matrices using unbalanced sparse regularization parameters. Secondly, the appropriate value of parameter β_n depends on the tensor to be decomposed. In this study, we selected β_n separately for each tensor in the experiments. When the sparse regularization parameter is larger, the extracted factor matrices are sparser, and the relative error of decomposition is also larger. The trade-off

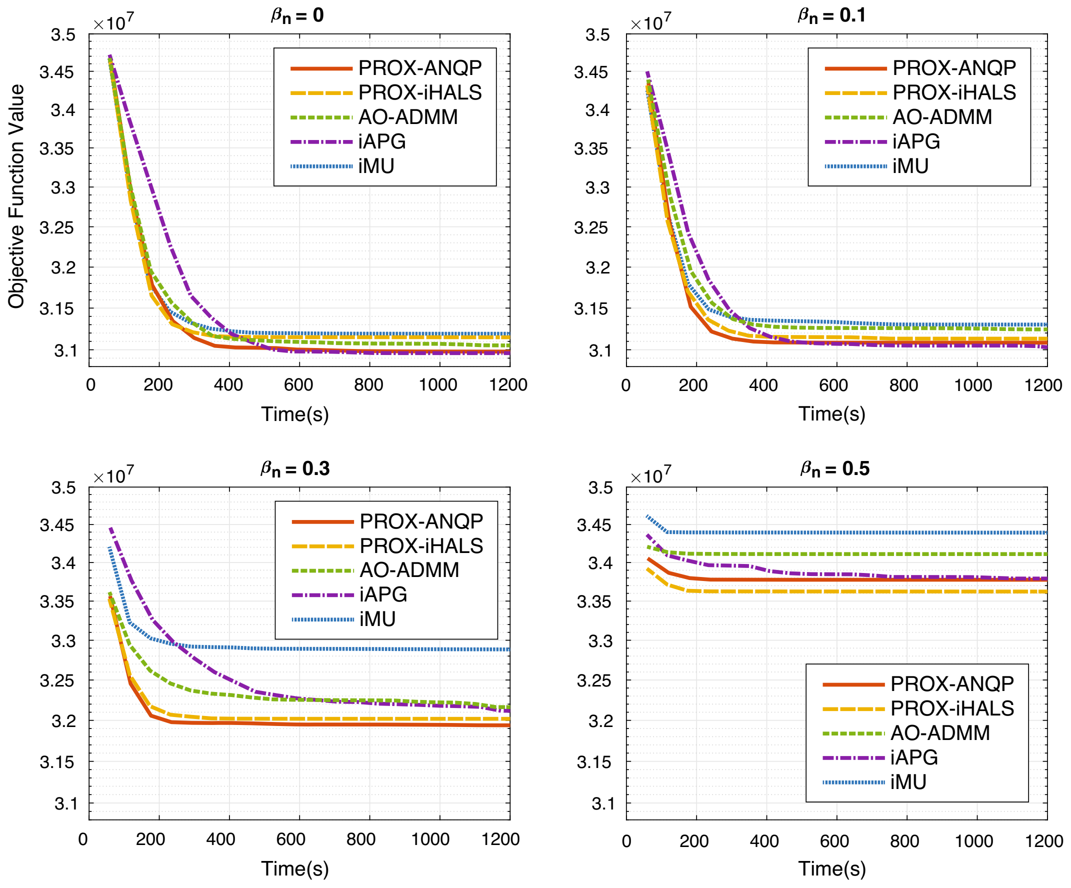

Fig. 4 The objective function value curves of sparse NCPs on fourth-order NIPS publications tensor

between the sparse level and the relative error depends on the meanings of real applications. An example of sparse regularization parameter selection of sparse NCP for ongoing EEG can be found in [44]. It is also possible to select an appropriate parameter for a concrete application using model-order selection methods [37], such as the Bayesian information criteria (BIC).

The third critical issue of sparse NCP is the sparse regularization item. In this study, we only investigated the l_1-norm item. In the future, it is worth trying to incorporate other types of sparse regularization items [2] to our sparse NCP model besides l_1-norm, such as the l_q-norm $(0 < q < 1)$ [36] and trace norm [29].

7 Conclusion

In this paper, we have investigated the nonnegative CANDECOMP/PARAFAC tensor decomposition with l_1-norm-based sparse regularization (sparse NCP). We have proposed a novel sparse NCP model using the proximal algorithm, which can guarantee the full column rank condition and the property of convergence to a stationary point. In addition, an inexact block coordinate descent scheme was presented to accelerate the computation of sparse NCP. In the inexact scheme, the subproblems are updated using multiple inner iterations. We have employed two algorithms for solving the proposed sparse NCP model with the proximal algorithm, including the inexact alternating nonnegative quadratic programming (PROX-ANQP) and the inexact hierarchical alternating least squares (PROX-iHALS). The experimental results on all synthetic, real-world, small-scale and large-scale tensors demonstrated that our sparse NCP methods can impose sparsity and extract meaningful sparse components successfully. Both PROX-ANQP and PROX-iHALS have exhibited the faster computational speed and better performances of imposing sparsity compared with other sparse NCP algorithms. The experimental results proved that the proposed sparse NCP with the proximal algorithm and inexact scheme is effective and efficient.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant No.91748105), National Foundation in China (No. JCKY2019110B009 & 2020-JCJQ-JJ-252), the

Fundamental Research Funds for the Central Universities [DUT20LAB303 & DUT20LAB308] in Dalian University of Technology in China, and the scholarship from China Scholarship Council (No. 201600090043). This study is to memorize Prof. Tapani Ristaniemi for his great help to Fengyu Cong, Zheng Chang and Deqing Wang. Prof. Tapani Ristaniemi supervised this work.

Funding Open access funding provided by University of Jyväskylä (JYU).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Allen G (2012) Sparse higher-order principal components analysis. In: Lawrence ND, Girolami M (eds) Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, PMLR, La Palma, Canary Islands, Proceedings of Machine Learning Research, 22: 27–36
2. Bach F, Jenatton R, Mairal J, Obozinski G (2012) Optimization with sparsity-inducing penalties. Found Trends® Mach Learn 4(1):1–106. https://doi.org/10.1561/2200000015
3. Bader BW, Kolda TG et al (2015) Matlab tensor toolbox version 2.6. Available online, http://www.sandia.gov/~tgkolda/TensorToolbox/
4. Bertsekas DP (2016) Nonlinear Programming, 3rd edn. Athena Scientific, Belmont, Massachusetts
5. Bro R, Kiers HAL (2003) A new efficient method for determining the number of components in PARAFAC models. J Chemom 17(5):274–286. https://doi.org/10.1002/cem.801
6. Bruckstein AM, Donoho DL, Elad M (2009) From sparse solutions of systems of equations to sparse modeling of signals and images. SIAM Rev 51(1):34–81. https://doi.org/10.1137/060657704
7. Cichocki A, Phan AH (2009) Fast local algorithms for large scale nonnegative matrix and tensor factorizations. IEICE Trans Fundam Electron Commun Comput Sci E92–A(3):708–721. https://doi.org/10.1587/transfun.e92.a.708
8. Cichocki A, Zdunek R (2006) NMFLAB—MATLAB toolbox for non-negative matrix factorization. http://www.bsp.brain.riken.jp/ICALAB/nmflab.html. Accessed 22 Nov 2017
9. Cichocki A, Zdunek R, Phan AH, Si Amari (2009) Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation. Wiley, New York
10. Cichocki A, Mandic D, Lathauwer LD, Zhou G, Zhao Q, Caiafa C, Phan HA (2015) Tensor decompositions for signal processing applications: from two-way to multiway component analysis. IEEE Signal Process Mag 32(2):145–163. https://doi.org/10.1109/msp.2013.2297439
11. Cong F, Lin QH, Kuang LD, Gong XF, Astikainen P, Ristaniemi T (2015) Tensor decomposition of EEG signals: a brief review. J Neurosci Methods 248:59–69. https://doi.org/10.1016/j.jneumeth.2015.03.018
12. Donoho DL (2006) For most large underdetermined systems of linear equations the minimal ℓ_1-norm solution is also the sparsest solution. Commun Pure Appl Math 59(6):797–829. https://doi.org/10.1002/cpa.20132
13. Elcoroaristizabal S, Bro R, García JA, Alonso L (2015) PARAFAC models of fluorescence data with scattering: a comparative study. Chemom Intell LabD Syst 142:124–130. https://doi.org/10.1016/j.chemolab.2015.01.017
14. Friedlander MP, Hatz K (2008) Computing non-negative tensor factorizations. Optim Methods Softw 23(4):631–647. https://doi.org/10.1080/10556780801996244
15. Gillis N, Glineur F (2012) Accelerated multiplicative updates and hierarchical ALS algorithms for nonnegative matrix factorization. Neural Computation 24(4):1085–1105. https://doi.org/10.1162/NECO_a_00256
16. Globerson A, Chechik G, Pereira F, Tishby N (2007) Euclidean Embedding of Co-occurrence Data. The Journal of Machine Learning Research 8: 2265–2295, http://www.jmlr.org/papers/volume8/globerson07a/globerson07a.pdf
17. Hong M, Razaviyayn M, Luo ZQ, Pang JS (2016) A unified algorithmic framework for block-structured optimization involving big data: with applications in machine learning and signal processing. IEEE Signal Process Mag 33(1):57–77. https://doi.org/10.1109/msp.2015.2481563
18. Hoyer PO (2004) Non-negative matrix factorization with sparseness constraints. J Mach Learn Res 5(Nov):1457–1469
19. Huang K, Sidiropoulos ND, Liavas AP (2016) A flexible and efficient algorithmic framework for constrained matrix and tensor factorization. IEEE Trans Signal Process 64(19):5052–5065. https://doi.org/10.1109/tsp.2016.2576427
20. Kim H, Park H (2008) Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. SIAM J Matrix Anal Appl 30(2):713–730. https://doi.org/10.1137/07069239x
21. Kim HJ, Ollila E, Koivunen V (2013) Sparse regularization of tensor decompositions. In: 2013 IEEE international conference on acoustics, speech and signal processing. IEEE. https://doi.org/10.1109/ICASSP.2013.6638376
22. Kim J, Park H (2011) Fast nonnegative matrix factorization: an active-set-like method and comparisons. SIAM J Sci Comput 33(6):3261–3281. https://doi.org/10.1137/110821172
23. Kim J, Park H (2012) Fast nonnegative tensor factorization with an active-set-like method. In: High-performance scientific computing. Springer, London, pp 311–326. https://doi.org/10.1007/978-1-4471-2437-5_16
24. Kim J, He Y, Park H (2014) Algorithms for nonnegative matrix and tensor factorizations: a unified view based on block coordinate descent framework. J Global Optim 58(2):285–319. https://doi.org/10.1007/s10898-013-0035-4
25. Kolda TG, Bader BW (2009) Tensor decompositions and applications. SIAM Rev 51(3):455–500. https://doi.org/10.1137/07070111x
26. Li N, Kindermann S, Navasca C (2013) Some convergence results on the regularized alternating least-squares method for tensor decomposition. Linear Algebra Appl 438(2):796–812. https://doi.org/10.1016/j.laa.2011.12.002
27. Li Y, Ngom A (2013) The non-negative matrix factorization toolbox for biological data mining. Source Code Biol Med 8(1):10. https://doi.org/10.1186/1751-0473-8-10

28. Liu J, Liu J, Wonka P, Ye J (2012) Sparse non-negative tensor factorization using columnwise coordinate descent. Pattern Recogn 45(1):649–656. https://doi.org/10.1016/j.patcog.2011.05.015
29. Liu Y, Shang F, Jiao L, Cheng J, Cheng H (2015) Trace norm regularized CANDECOMP/PARAFAC decomposition with missing data. IEEE Trans Cybern 45(11):2437–2448. https://doi.org/10.1109/tcyb.2014.2374695
30. Mørup M (2011) Applications of tensor (multiway array) factorizations and decompositions in data mining. Wiley Interdiscip Rev Data Mining Knowl Discov 1(1):24–40. https://doi.org/10.1002/widm.1
31. Mørup M, Hansen LK, Arnfred SM (2008) Algorithms for sparse nonnegative tucker decompositions. Neural Comput 20(8):2112–2131. https://doi.org/10.1162/neco.2008.11-06-407
32. Papalexakis EE, Sidiropoulos ND, Bro R (2013) From *k*-means to higher-way co-clustering: Multilinear decomposition with sparse latent factors. IEEE Trans Signal Process 61(2):493–506. https://doi.org/10.1109/tsp.2012.2225052
33. Razaviyayn M, Hong M, Luo ZQ (2013) A unified convergence analysis of block successive minimization methods for nonsmooth optimization. SIAM J Optim 23(2):1126–1153. https://doi.org/10.1137/120891009
34. Selesnick I (2017) Sparse regularization via convex analysis. IEEE Trans Signal Process 65(17):4481–4494. https://doi.org/10.1109/tsp.2017.2711501
35. Sidiropoulos ND, Lathauwer LD, Fu X, Huang K, Papalexakis EE, Faloutsos C (2017) Tensor decomposition for signal processing and machine learning. IEEE Trans Signal Process 65(13):3551–3582. https://doi.org/10.1109/tsp.2017.2690524
36. Sigurdsson J, Ulfarsson MO, Sveinsson JR (2014) Hyperspectral unmixing with l_q regularization. IEEE Trans Geosci Remote Sens 52(11):6793–6806. https://doi.org/10.1109/tgrs.2014.2303155
37. Stoica P, Selen Y (2004) Model-order selection: a review of information criterion rules. IEEE Signal Process Mag 21(4):36–47. https://doi.org/10.1109/msp.2004.1311138
38. Timmerman ME, Kiers HAL (2000) Three-mode principal components analysis: choosing the numbers of components and sensitivity to local optima. British J Math Stat Psychol 53(1):1–16. https://doi.org/10.1348/000711000159132
39. Veganzones MA, Cohen JE, Farias RC, Chanussot J, Comon P (2016) Nonnegative tensor CP decomposition of hyperspectral data. IEEE Trans Geosci Remote Sens 54(5):2577–2588. https://doi.org/10.1109/tgrs.2015.2503737
40. Vervliet N, Lathauwer LD (2019) Numerical optimization-based algorithms for data fusion. In: Data handling in science and technology. Elsevier, pp 81–128. https://doi.org/10.1016/B978-0-444-63984-4.00004-1
41. Viswanath B, Mislove A, Cha M, Gummadi KP (2009) On the evolution of user interaction in facebook. In: Proceedings of the 2nd ACM workshop on Online social networks—WOSN '09. ACM Press. https://doi.org/10.1145/1592665.1592675
42. Wang D, Cong F (2021) An inexact alternating proximal gradient algorithm for nonnegative CP tensor decomposition. Sci China Technol Sci 64(9):1893–1906. https://doi.org/10.1007/s11431-020-1840-4
43. Wang D, Cong F, Zhao Q, Toiviainen P, Nandi AK, Huotilainen M, Ristaniemi T, Cichocki A (2016) Exploiting ongoing EEG with multilinear partial least squares during free-listening to music. In: IEEE 26th international workshop on machine learning for signal processing (MLSP). IEEE. https://doi.org/10.1109/mlsp.2016.7738849
44. Wang D, Wang X, Zhu Y, Toiviainen P, Huotilainen M, Ristaniemi T, Cong F (2018) Increasing stability of EEG components extraction using sparsity regularized tensor decomposition. In: Advances in neural networks – ISNN 2018. Springer, pp 789–799. https://doi.org/10.1007/978-3-319-92537-0_89
45. Wang D, Cong F, Ristaniemi T (2019) Higher-order nonnegative CANDECOMP/PARAFAC tensor decomposition using proximal algorithm. In: 2019 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, pp 3457–3461. https://doi.org/10.1109/ICASSP.2019.8683217
46. Williams AH, Kim TH, Wang F, Vyas S, Ryu SI, Shenoy KV, Schnitzer M, Kolda TG, Ganguli S (2018) Unsupervised discovery of demixed, low-dimensional neural dynamics across multiple timescales through tensor component analysis. Neuron 98(6):1099-1115.e8. https://doi.org/10.1016/j.neuron.2018.05.015
47. Xu Y (2015) Alternating proximal gradient method for sparse nonnegative tucker decomposition. Math Program Comput 7(1):39–70. https://doi.org/10.1007/s12532-014-0074-y
48. Xu Y, Yin W (2013) A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. SIAM J Imag Sci 6(3):1758–1789. https://doi.org/10.1137/120887795
49. Yang Y, Pesavento M, Luo ZQ, Ottersten B (2020) Inexact block coordinate descent algorithms for nonsmooth nonconvex optimization. IEEE Trans Signal Process 68:947–961. https://doi.org/10.1109/tsp.2019.2959240
50. Zhang H, Wang S, Xu X, Chow TWS, Wu QMJ (2018) Tree2vector: learning a vectorial representation for tree-structured data. IEEE Trans Neural Networks Learn Syst 29(11):5304–5318. https://doi.org/10.1109/tnnls.2018.2797060

Authors and Affiliations

Deqing Wang[1,2] · **Zheng Chang**[2,3] · **Fengyu Cong**[1,2,4,5]

✉ Deqing Wang
deqing.wang@foxmail.com

✉ Fengyu Cong
cong@dlut.edu.cn

Zheng Chang
zheng.chang@jyu.fi

[1] School of Biomedical Engineering, Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology, Dalian 116024, China

[2] Faculty of Information Technology, University of Jyväskylä, Jyväskylä 40100, Finland

[3] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China

[4] School of Artificial Intelligence, Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology, Dalian 116024, China

[5] Key Laboratory of Integrated Circuit and Biomedical Electronic System, Liaoning Province, Dalian University of Technology, Dalian 116024, China