Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 11, No. 2, pp. 385–402, April 2015

RECOVERING LOW CP/TUCKER RANKED TENSORS, WITH APPLICATIONS IN TENSOR COMPLETION

Xiang Gao*, Bo Jiang† and Shaozhe Tao

Abstract: In this paper, we propose a method to find tensor decompositions with low CP rank. In our approach, no prior knowledge is assumed regarding the rank of the tensor to be recovered. The key underlying idea is to take advantage of group sparsity structure. The $L_{1,2}$-norm regularization and the $L_{1,\infty}$-norm regularization are proposed to achieve the group sparse effect. Then the problem reduce to the multi-block optimization, which can be efficiently solved by the so-called Block Coordinate Decent (BCD) method. We also show that our approach can be used to solve the tensor low CP/Tucker rank completion problem. Simulation results demonstrate that our new method performs well numerically, especially when the tensor to be recovered indeed has a low rank structure.

Key words: *tensor, low rank, group sparsity, tensor completion*

Mathematics Subject Classification: *15A69, 15A03, 65F99*

1 Introduction

A tensor is a multidimensional array which can be regarded as an higher dimensional analogy to matrix. Accordingly, an N-way or Nth-order tensor in the real space $\mathbf{R}^{I_1\times I_2\times\cdots\times I_N}$ refers to the entity often denoted as $\mathcal{A} = (\mathcal{A}_{i_1 i_2\cdots i_N})_{I_1\times I_2\times\cdots\times I_N}$. As in the matrix case, decomposing a given tensor is an important topic and has been studied quite a lot over the past decades. We refer the interested readers to the nice survey [16] for various applications of tensors. Different from the decomposition of a matrix, which is clear and straightforward, there are more than one way to decompose a given tensor. Among them, there are two popular decompositions, known as *CP decomposition* and *Tucker decomposition* respectively.

The CP decomposition factorizes a tensor $\mathcal{X} \in \mathbf{R}^{I_1\times I_2\times\cdots\times I_N}$ into a sum of rank-one tensors:

$$\mathcal{X} = \sum_{r=1}^{R} a^{1,r} \circ a^{2,r} \circ \cdots \circ a^{N,r} := [\![A_1, A_2, \cdots, A_N]\!], \tag{1.1}$$

where "$\circ$" denotes the outer product of vectors and $A_n = [a^{n,1}, a^{n,2}, \cdots, a^{n,R}] \in \mathbf{R}^{I_n\times R}$ for $1 \le n \le N$. The CP rank for a tensor $\mathcal{X}$ denoted by rank_{CP}, which is a natural

*Research of this author was supported in part by the National Science Foundation under Grant Number CMMI-1161242.

†The corresponding author. Research of this author was supported in part by the National Science Foundation under Grant Number CMMI-1161242 and National Natural Science Foundation of China under Grant Number 11401364.

generalization of the rank for matrix, is associated with the *smallest* CP decomposition and is defined as follows.

$$\text{rank}_{CP}(\mathcal{X}) = \min\left\{ R \;\middle|\; \mathcal{X} = \sum_{r=1}^{R} a^{1,r} \circ a^{2,r} \circ \cdots \circ a^{N,r} \right\} \tag{1.2}$$

The Tucker decomposition is a sort of compression of high-order tensor. It decomposes a tensor into a core tensor multiplied by a matrix along each mode. For an N-way tensor $\mathcal{X} \in \mathbf{R}^{I_1 \times I_2 \times \cdots \times I_N}$, we have the Tucker decomposition as follows

$$\begin{aligned} \mathcal{X} &= \mathcal{G} \times_1 A_1 \times_2 A_2 \cdots \times_N A_N \\ &= \sum_{i_1=1}^{r_1} \sum_{i_2=1}^{r_2} \cdots \sum_{i_N=1}^{r_N} \mathcal{G}_{i_1 i_2 \cdots i_N} a^{1,i_1} \circ a^{2,i_2} \circ \cdots \circ a^{N,i_N} := [\![\mathcal{G}; A_1, \cdots, A_N]\!], \end{aligned} \tag{1.3}$$

where $\mathcal{G} \in \mathbf{R}^{r_1 \times r_2 \times \cdots \times r_N}$ is the so-called core tensor, $A_n = [a^{n,1}, a^{n,2}, \cdots, a^{n,r_n}] \in \mathbf{R}^{I_n \times r_n}$ and "$\times_n$" denotes the n-mode product of the tensor for $1 \le n \le N$. The Tucker rank of a tensor denoted by rank_{TK} is a vector corresponding to the size of the core tensor associated with the *smallest* Tucker decomposition. Typically, a Tucker rank $(r_1, r_2, \cdots, r_N)$ means that the size of the core tensor is $r_1 \times r_2 \times \cdots \times r_N$. It has been shown in the literature [16] that, if $A_1, \cdots, A_N$ are all orthogonal, the smallest Tucker decomposition can be accomplished in polynomial time.

However, to find the smallest CP decomposition of a given tensor is a very challenging problem. Notice that the problem of determining the CP rank, i.e. the smallest number of rank-one terms in CP decomposition, is already NP-complete [5]. Even for a specific $9 \times 9 \times 9$ tensor, we only know its CP rank lies in between 18 and 23 (cf. [8]). That motivates people to study the CP low-rank decomposition (also referred to as CP low-rank approximation in this paper). Its goal is to factorize a tensor into a sum of rank-one tensors with as less terms as possible, i.e. lower CP rank, that approximates the original tensor well.

In the past, people always try a brutal procedure to find a fairly good CP approximation by repeating the decomposition with a fixed number of rank-one components until reach a particular number which is satisfactory. When the number of components is fixed, there is a well-known alternating least squares (ALS) algorithm to compute the CP decomposition, and that is first proposed by Carroll and Chang [9] and Harshman [14]. Moreover, there are several other algorithms regarding the CP decomposition which had also been compared with ALS, including derivative-based method(dGN, PMF3) [13] and ASD [7]. Although they perform better than ALS to some extent in term of convergence properties, much more computational difficulties are involed. It should be noticed that among all these methods, the number of components, R, of the CP decomposition (1.1) is always a prerequisite.

In this paper, we are trying to find a good CP approximation in one shot. As we would show afterwards, obtaining the CP/Tucker low-rank decomposition of a tensor can be transformed into the problem of finding the factor matrices with as many zero columns as possible. Suggested by this observation, we can take advantage of the group sparsity structure to achieve the low rank effect. The so-called group sparsity has been studied widely in statistics and machine learning area. It is well known that L_1 norm can promote the sparsity solution. In recent years, techniques for incorporating group information using mixed-norm regularization have been studied. Particularly, Yuan and Lin and others [11,12] studied $L_{1,2}$-norm regularization to promote group sparsity, known as the group LASSO scheme. Since there is no evidence that shows the L_2-norm is the only valid norm to be used in the group sparsity scheme [4], we further study the possibility of using other L_p-norms

($p > 2$) in the group sparsity framework which will in fact lead to a $L_{1,p}$ regularization. In our method, if the columns of the factor matrices are treated as the groups, then we are able to acquire low rank effect by utilizing the group sparsity structure.

The completion problem is a missing value estimation problem. The so-called low-rank tensor completion is to recover a low-rank tensor from partial information. As we have already seen at the very beginning, there are multiple ways to define the tensor rank. The current literatures are mainly devoted to the low-n-rank completion [10, 15]. Another focus of this paper is CP/Tucker low-rank completion problem. Actually, one may find out that completion problems are natural applications of the decomposition.

Essentially, we can obtain the low rank decomposition and completion of a tensor via solving a multi-block optimization problem. Accordingly, the objective function consists of two parts. One is the least square term for the traditional tensor decomposition, while the other one is the group sparse regularization term that would lead to a low rank solution. On the computational perspective, due to the embedded block structure of the problem, the block updating strategy is naturally considered. Specifically, we adopt the Block Coordinate Descent (BCD) method in our computation, which updates blocks cyclically. For effective implementation of the BCD algorithm, we adopt the so-called prox-linear update rule which is efficient for this particular regularized multi-convex problems [17]. Our numerical results show that, by incorporating our new idea of group sparsity, the low rank decomposition can be obtained.

Throughout, we denote $\mathbf{R}^n$ to be the n-dimensional Euclidean space. We denote the high order tensor by calligraphic letters, e.g. $\mathcal{X}$. The *order* N of a tensor is the number of dimensions, also known as *ways*. Matrices, vectors and scalars are denoted by capital letters, boldface lowercase letters and non-bold lowercase letters respectively. The *Frobenius norm* of a tensor $\mathcal{X} \in \mathbf{R}^{I_1 \times I_2 \times \cdots \times I_N}$, by notation $\|\cdot\|$, is the square root of the sum of the squares of all its elements, i.e.,

$$\|\mathcal{X}\| = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \cdots i_N}^2}$$

The following products of vectors or matrices will be very useful in our discussion. The *outer product* of vectors $a^1, a^2, \cdots, a^N$ with $a^n \in \mathbf{R}^{I_n}$ is denoted by $a^1 \circ a^2 \circ \cdots a^N$ such that

$$(a^1 \circ a^2 \circ \cdots \circ a^N)_{i_1 i_2 \cdots i_N} = a_{i_1}^1 a_{i_2}^2 \cdots a_{i_N}^N.$$

We call tensor $\mathcal{X}$ is rank-one, if it is the outer product of some vectors. The *n-mode (matrix) product* of a tensor $\mathcal{X} \in \mathbf{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with a matrix $U \in \mathbf{R}^{J \times I_n}$ is denoted by $\mathcal{X} \times_n U$ and is of size $I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N$. Elementwise, we have

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

The symbol "$\otimes$" denotes the standard *Kronecker product* of matrices. The *Khatri-Rao product* of matrices $A \in \mathbf{R}^{I \times K}$ and $B \in \mathbf{R}^{J \times K}$, is denoted by $A \odot B$. The result is a matrix of size $(IJ) \times (K)$ defined by

$$A \odot B = [a_1 \otimes b_1 \ a_2 \otimes b_2 \cdots a_K \otimes b_K]$$

The remainder of the paper is organized as follows. Section 2 describes the relationship between the low-rank decomposition of a tensor and the group sparsity property of its factor

matrices. In Section 3, we discuss the problem formulation and solution methods of CP low rank decomposition. Section 4 is devoted to CP/Tucker low rank completion. In Section 5, we provide some numerical results to justify our algorithms. Finally, we summarize our research in Section 6.

2 Sufficient Conditions for Low-Rank Decomposition

In this section, we show how the low-rank decomposition in the sense of CP/Tucker relates to the group sparsity property of certain matrices. In fact, our research is motivated by the following key observations.

Proposition 2.1.

(i) *In CP decomposition* (1.1)*, suppose there is a zero colume of matrix* A_n *for some* $1 \leq n \leq N$*, then this decomposition actually admits at most* $R-1$ *rank-one terms. Therefore, obtaining a low-rank CP decomposition can be achieved by generating a decomposition with as many zero column vectors as possible in either* $A_1, A_2, \cdots, A_N$*.*

(ii) *In Tucker decompostion* (1.3)*, suppose there is a zero colume of matrix* A_n *for some* $1 \leq n \leq N$*, then this decomposition actually admits a core with size at most* $(r_1, \cdots, r_{n-1}, r_n - 1, r_{n+1}, \cdots, _N)$*. Therefore, obtaining a low-rank Tucker decomposition can be achieved by generating a decomposition with all the matrices* $A_1, A_2, \cdots, A_N$ *have as many zero columns as possible.*

Proof. We assume that the j-th column of A_n is a zero vector for some $1 \leq j \leq R$ and $1 \leq n \leq N$ in CP decomposition (1.1), namely $a^{n,j} = 0$. Then

$$\mathcal{X} = \sum_{r=1}^{R} a^{1,r} \circ a^{2,r} \circ \cdots \circ a^{N,r} = \sum_{r=1, r\neq j}^{R} a^{1,r} \circ a^{2,r} \circ \cdots \circ a^{N,r},$$

where the number of rank-one terms is at most $R-1$. Therefore, if there are many zero columns in either $A_1, A_2, \cdots, A_N$, then $\mathcal{X}$ is of low CP rank.

In Tucker decomposition (1.3), without loss of generality, we assume $a^{1,1} = 0$, which means the first column of A_1 is a zero vector. Then

$$\begin{aligned} \mathcal{X} &= \sum_{i_1=1}^{r_1} \sum_{i_2=1}^{r_2} \cdots \sum_{i_N=1}^{r_N} \mathcal{G}_{i_1 i_2 \cdots i_N} a^{1,i_1} \circ a^{2,i_2} \circ \cdots \circ a^{N,i_N} \\ &= \sum_{i_1=2}^{r_1} \sum_{i_2=1}^{r_2} \cdots \sum_{i_N=1}^{r_N} \mathcal{G}_{i_1 i_2 \cdots i_N} a^{1,i_1} \circ a^{2,i_2} \circ \cdots \circ a^{N,i_N}. \end{aligned}$$

By constructing $\hat{\mathcal{G}} \in \mathbf{R}^{(r_1-1)\times r_2 \times \cdots \times r_N}$ and $\hat{A}_1 \in \mathbf{R}^{I_1 \times (r_1-1)}$ such that

$$\hat{\mathcal{G}}_{i_1, i_2, \cdots, i_N} = \mathcal{G}_{i_1+1, i_2, \cdots, i_N} \quad \text{and} \quad (\hat{A}_1)_{j_1, j_2} = (A_1)_{j_1+1, j_2},$$

we have that $[\hat{\mathcal{G}}; \hat{A}_1, \cdots, A_N]$ is a valid Tucker decomposition of $\mathcal{X}$ with core size $(r_1 - 1, r_2, \cdots, r_N)$. Therefore, if all $A_1, A_2, \cdots, A_N$ have many zero column vectors, then $\mathcal{X}$ is of low Tucker rank. □

Now let's recall the concept of group sparsity. That is the target candidate has natural grouping of its elements, and the elements within each group are likely to be either *all zeros*

or *all nonzeros*. In our case, the matrix A_n in either CP decomposition (1.1) or Tucker Decomposition (1.3) is the target candidate, and we treat each column of this matrix as a group. Thus, from Proposition 2.1, we conclude that the group sparsity of $A_1, A_2, \cdots, A_N$ implies the CP low-rank decomposition and the Tucker low-rank decomposition respectively.

To further take advantage of the group sparsity structure, we wish to write the CP low-rank decomposition or the Tucker low-rank decomposition *explicitly* in terms of matrices $A_1, A_2, \cdots, A_N$. To this end, we introduce the so-called matricization technique, also known as unfolding or flattening, which is the process of reordering the elements of an N-way array into a matrix. For instance, a $2 \times 3 \times 4$ tensor can be arranged as a 6×4 matrix or a 3×8 matrix, and so on. The mode-n matricization of a tensor $\mathcal{X} \in \mathbf{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is denoted by $X_{(n)}$ and arranges the mode-n fibers to be the columns of the resulting matrix. Though conceptually simple, the formal notation is complicated. Tensor element $(i_1, i_2, \cdots, i_N)$ maps to matrix element (i_n, j), where

$$j = 1 + \sum_{\substack{k=1 \\ k \neq n}}^{N} (i_k - 1) J_k \qquad \text{with} \qquad J_k = \prod_{\substack{m=1 \\ m \neq n}}^{k-1} I_m$$

Using this notation as well as the Khatri-Rao product "$\odot$" of matrices, we manage to rewrite the CP decomposition (1.1) equivalently as:

$$X_{(n)} = A_n (A_N \odot \cdots A_{n+1} \odot A_{n-1} \cdots A_1)^\top, \qquad \forall n = 1, 2, \ldots, N. \tag{2.1}$$

Similarly, we can also rewrite the Tucker decomposition (1.3) in a matricized form like below:

$$X_{(n)} = A_n G_{(n)} (A_N \otimes \cdots A_{n+1} \otimes A_{n-1} \cdots A_1)^\top, \qquad \forall n = 1, 2, \ldots, N, \tag{2.2}$$

where "$\otimes$" is the standard Kronecker product of matrices.

Consequently, we are able to show later that the problems of tensor low-rank decomposition and the tensor completion can both be casted as matrix optimization problems. As mentioned that the smallest Tucker decomposition could be accomplished in polynomial time, we here focus on the CP low-rank approximation.

3 Tensor CP Low-Rank Approximation

In this section, we shall present how to obtain a tensor low-rank approximation in the sense of CP via optimization scheme. As illustrated in last section, our method is motivated by the group sparsity framework. From the most general perspective, here we provide the idea and formulation to find the CP low-rank approximation of a N-way tensor. Particularly, suppose we are given a tensor $\mathcal{X} \in \mathbf{R}^{I_1 \times \cdots \times I_N}$ and we do not know its exact rank, what we would do is to start with a relatively large rank and implement our new approach of group sparsity to decompose $\mathcal{X}$.

3.1 Problem formulation

In CP low-rank approximation, we want to factorize a tensor into a sum of rank-one tensors with as less terms as possible, which approximates the original tensor well. Consequently, we put both criteria in the objective and get the following formulation.

$$\min_{A_1, A_2, \cdots, A_N} \frac{1}{2} \|\mathcal{X} - [\![A_1, A_2, \cdots, A_N]\!]\|^2 + \rho \, \mathrm{rank}_{CP}([\![A_1, A_2, \cdots, A_N]\!]), \tag{3.1}$$

where ρ is the parameter to control the degree of low-rank. From the discussion in Section 2, we know that the group sparsity of matrices $A_1, A_2, \cdots, A_N$ can imply the low-rank structure of tensor $[\![A_1, A_2, \cdots, A_N]\!]$.

It is well known that the L_1-norm is a popular quantifying term for standard sparsity structure. The techniques for incorporating group information using mixed-norm regularization have been studied widely in machine learning and statistics. In particular, Yuan and Lin [12] proposes $L_{1,2}$-norm regularization term and name it group LASSO, which can lead to group sparsity effect if data are organized as groups. The group LASSO proposed by Yuan and Lin solves the convex problems that have the following form:

$$\min_{\beta}(\|y - \sum_{i=1}^{R} X_i\beta_i\|^2 + \rho \sum_{i=1}^{R} \|\beta_i\|_2)$$

where y is the given data, X_i represents the predictor corresponding to the ith group and β_i is the corresponding vector.

In our model, we treat each column as a group and attempt to take the advantage of the group LASSO to achieve the low rank effect. By replacing the rank function in (3.1) with $L_{1,2}$-norm regularization, we formulate our CP low-rank decomposition problem as below:

$$\min_{A_1, A_2, \cdots, A_N} \frac{1}{2}\|\mathcal{X} - [\![A_1, A_2, \cdots, A_N]\!]\|^2 + \rho \left(\sum_{t=1}^{R} \sum_{i=1}^{N} \|a^{i,t}\|_2 \right) \tag{3.2}$$

where $a^{i,t}$ is the t-th column of matrix A_i for $1 \leq i \leq N$ and $1 \leq t \leq R$.

Besides the $L_{1,2}$-norm regularization, there is a natural consideration arises from the question about whether the $L_{1,2}$-norm can be replaced by some other $L_{1,p}$-norms for which $p > 2$. Since it is the L_1-norm that really promotes the sparsity structure, the choice of the L_2-norm in the group LASSO might not be necessarily unchangeable. In practice, another popular choice is the $L_{1,\infty}$-norm regularization [4]. If this regularization is applied, our CP low-rank decomposition problem can be formulated as below

$$\min_{A_1, A_2, \cdots, A_N} \frac{1}{2}\|\mathcal{X} - [\![A_1, A_2, \cdots, A_N]\!]\|^2 + \rho \left(\sum_{t=1}^{R} \sum_{i=1}^{N} \|a^{i,t}\|_\infty \right) \tag{3.3}$$

where $a^{i,t}$ is the t-th column of matrix A_i for $1 \leq i \leq N$ and $1 \leq t \leq R$.

Note that once the t-th column $a^{i,t}$ of matrix A_i is zero, then whether the t-th column $a^{k,t}$ of any A_k with $k \neq i$ is zero or not will not make any difference on the approximation in (3.2). This fact can be easily seen from the proof of Proposition 2.1. As shown in the proof, if $a^{i,t} = 0$, then the term $a^{1,t} \circ a^{2,t} \circ \cdots \circ a^{N,t}$ in $[\![A_1, A_2, \cdots, A_N]\!]$ will always be zero whatever the other $a^{k,t}$ $(k \neq i)$ is. However, we enforce the t-th column of every A_k in (3.2) to be zero, to make the shrinkage effect of the regularization more balanced among different modes. The same logic is also applied to tensor low rank completion problem in Section 4.

3.2 Regularized multiconvex optimization framework

Notice that the objective in either (3.2) or (3.3) has multiple block variables $A_1, A_2, \cdots, A_N$. This multi-block structure can be really helpful in the algorithm design, and the multi-block optimization problem has been carefully studied. For instance, recently, Xu and Yin [17] considered the following regularized multiconvex optimization.

$$\min_{x} f(x_1, \cdots, x_N) + \sum_{i=1}^{N} r_i(x_i),$$

where $x_1, \cdots, x_N$ is the decision variables, f is assumed to be a differentiable and *block multiconvex* function, that is f is a convex function of x_i while all the other blocks are fixed, and $r_i, i = 1, \cdots, N$ are convex functions.

In our formulation (3.2) and (3.3) , we let $f(A_1, A_2, \cdots, A_N) := \frac{1}{2}\|\mathcal{X} - [\![A_1, A_2, \cdots, A_N]\!]\|^2$, $r_i(A_i) = \rho \sum_{t=1}^{R} \|a^{i,t}\|_p$ for $1 \le i \le N$, $p \in \{2, \infty\}$ and consider the problem:

$$\min_{A_1, A_2, \cdots, A_N} f(A_1, A_2, \cdots, A_N) + \sum_{i=1}^{N} r_i(A_i). \tag{3.4}$$

Obviously $r_i(A_i)$ is convex with respect to A_i. Moreover, from (2.1), we can see that

$$\begin{aligned} f(A_1, A_2, \cdots, A_N) &= \frac{1}{2}\|X_{(i)} - A_i(A_N \odot \cdots A_{i+1} \odot A_{i-1} \cdots A_1)^\top\|^2 \\ &:= f_i(A_1, A_2, \cdots, A_N),\ \forall\ 1 \le i \le N, \end{aligned}$$

implying that $f(A_1, A_2, \cdots, A_N)$ is a differentiable and block multiconvex function. Moreover, the i-th block-partial gradient of f is given by

$$\nabla_{A_i} f = \nabla_{A_i} f_i = (A_i(A_N \odot \cdots A_{i+1} \odot A_{i-1} \cdots A_1)^\top - X_{(i)})(A_N \odot \cdots A_{i+1} \odot A_{i-1} \cdots A_1), \tag{3.5}$$

and the partial gradient function $\nabla_{A_i} f$ is Lipschitz continuous. In conclusion, the problem (3.4) can fit into the regularized multiconvex optimization framework in [17] and the algorithms therein can be applied.

Moreover, note that the convergence results of the algorithm in [17] still holds here, as long as the objective function in (3.4) satisfies the so-called Kurdyka-Łojasiewicz (KL) property, which is defined as follows.

Definition 3.1. A function $\psi(x)$ satisfies the Kurdyka-Łojasiewicz (KL) property at point $\bar{x} \in \mathrm{dom}(\partial\psi)$ if there exist $\phi(s) = cs^{1-\theta}$ for some $c > 0$ and $\theta \in [0, 1)$, and a certain neighborhood U of $\bar{x}$, such that the KL inequality holds

$$\phi'(|\psi(x) - \psi(\bar{x})|)\mathrm{dist}(0, \partial\psi(x)) \ge 1, \quad \forall x \in \mathrm{U} \cap \mathrm{dom}(\partial\psi) \ \ \textit{and} \ \ \psi(x) \ne \psi(\bar{x}), \tag{3.6}$$

where $\mathrm{dom}(\partial\psi) \triangleq \{x : \partial\psi(x) \ne \varnothing\}$ and $\mathrm{dist}(0, \partial\psi(x)) \triangleq \min\{\|y\| : y \in \partial\psi(x)\}$.

The functions which have the KL property contain a large amount of classes of functions in applications, for instance, semi-algebraic, subanalytic and strongly convex functions all have the KL property (see the discussion in Section 2.2 of [17] for more details). Our objective function in (3.4) only contains three types of functions, namely, $\|\cdot\|_2^2$, $\|\cdot\|_2$ and $\|\cdot\|_\infty$. Notice that all those three functions are semi-algebraic function, and the function in (3.4) are linear compositions of $A_i, i = 1, 2 \ldots, N$ or their components products, we readily know that the objective function in (3.4) is also semi-algebraic, hence it has the KL property and the convergence of the algorithm can be guaranteed.

For general multiblock optimization problems, a commonly used strategy is to update the the block variables iteratively. Then, in each iteration, how to update the block variables becomes a key issue. In the following we shall present the block updating rules which is called *block coordinate descent* (BCD) method.

3.3 BCD updating rule of Gauss-Seidel type

The BCD of Gauss-Seidel type is to cyclically update each of $A_1, \cdots, A_N$ while fixing the remaining blocks at their last updated values. To be more specific, in the k-th iteration, updating scheme of BCD is described as follows:

$$A_i^{(k)} := \underset{A_i}{\operatorname{argmin}}\, g_i(A_1^{(k)}, \cdots, A_{i-1}^{(k)}, A_i, A_{i+1}^{(k-1)}, \cdots, A_N^{(k-1)}), \tag{3.7}$$

where g_i is certain multiblock function for $1 \le i \le N$, which will be specified later . For example, a natural choice of g_i is

$$g_i(A_1^{(k)}, \cdots, A_{i-1}^{(k)}, A_i, A_{i+1}^{(k-1)}, \cdots, A_N^{(k-1)}) = f(A_1^{(k)}, \cdots, A_{i-1}^{(k)}, A_i, A_{i+1}^{(k-1)}, \cdots, A_N^{(k-1)}) + r_i(A_i).$$

In [17], three alternative choices of g_i were provided under the scheme of BCD, and the authors proved that the sequence generated by BCD converges to critical point of (3.4) under some mild conditions. For practical purpose, we adopt the following prox-linear calculating rule:

$$A_i^{(k)} = \underset{A_i}{\operatorname{argmin}} \langle H_i^{(k)}, A_i - \hat{A}_i^{(k-1)} \rangle + \frac{L_i^{(k-1)}}{2} \|A_i - \hat{A}_i^{(k-1)}\|^2 + \rho \sum_{t=1}^{R} \|a^{i,t}\|_p \tag{3.8}$$

where $p \in \{2, \infty\}$, $H_i^{(k)} = \nabla f_{A_i}(A_1^{(k)}, \cdots, \hat{A}_i^{(k-1)}, A_{i+1}^{(k-1)}, \cdots, A_N^{(k-1)})$ is the block-partial gradient of f at $\hat{A}_i^{(k-1)}$ and $\hat{A}_i^{(k-1)}$ is given by the following formula:

$$\hat{A}_i^{(k-1)} = A_i^{(k-1)} + \omega_i^{(k-1)}(A_i^{(k-1)} - A_i^{(k-2)})$$

with $\omega_i^{(k-1)} \ge 0$ being the extrapolation weight. It is reported in [17] that this particular calculating rule appears to be very efficient in many tests.

3.3.1 Specify the parameters

In the following, we illustrate how to set the parameters $L_i^{(k-1)}$ and $\omega_i^{(k-1)}$ in order to satisfy the conditions in [17]. For the i-th block, let

$$P_i^{(k-1)} = A_N^{(k-1)} \odot \cdots A_{i+1}^{(k-1)} \odot A_{i-1}^{(k)} \cdots A_1^{(k)}, \tag{3.9}$$

$$L_i^{(k-1)} = \max\left\{ l^{k-2}, \|(P_i^{(k-1)})^\top P_i^{(k-1)}\| \right\} \text{ for } 1 \le i \le N, \tag{3.10}$$

where $l^{k-2} = \min_{1 \le i \le N} L_i^{(k-2)}$.

We take $t_0 = 1$, $t_k = \frac{1}{2}\left(1 + \sqrt{(1 + 4t_{k-1}^2)}\right)$ and $\hat{\omega}_{k-1} = \frac{t_{k-1}-1}{t_k}$; then

$$\omega_i^{(k-1)} = \min\left(\hat{\omega}_{k-1}, \delta_\omega \sqrt{\frac{l^{k-2}}{L_i^{(k-1)}}}\right) \tag{3.11}$$

where $\delta_\omega < 1$ is pre-selected. Therefore, under the current notation, $\hat{A}_i^{(k-1)} = A_i^{(k-1)} + \omega_i^{(k-1)}(A_i^{(k-1)} - A_i^{(k-2)})$ and

$$H_i^{(k)} = \left(\hat{A}_i^{(k-1)}(P_i^{(k-1)})^\top - X_{(i)}\right) P_i^{(k-1)}, \quad \forall 1 \le i \le N. \tag{3.12}$$

3.3.2 Closed form solution for the subproblem

Now suppose $H_i^{(k)}$, $L_i^{(k-1)}$ and $\hat{A}_i^{(k-1)}$ are given, let's investigate how to solve (3.8). We first consider the case when $p = 2$. Notice that this problem is separable in columns, so it can be transformed as

$$\begin{aligned}(a^t)^{(k)} &= \underset{a^t}{\operatorname{argmin}} \langle (h^t)^{(k)}, a^t - (\hat{a}^t)^{(k-1)} \rangle + \frac{L_i^{(k-1)}}{2} \|a^t - (\hat{a}^t)^{(k-1)}\|^2 + \rho \|a^t\|_2 \\ &= \underset{a^t}{\operatorname{argmin}} \frac{L_i^{(k-1)}}{2} \left\| a^t - (\hat{a}^t)^{(k-1)} + \frac{(h^t)^{(k)}}{L_i^{(k-1)}} \right\|^2 + \rho \|a^t\|_2\end{aligned}$$

where $a^t, (a^t)^{(k)}, (\hat{a}^t)^{(k-1)}, (h^t)^{(k)}$ are the t-th column of $A_i, A_i^{(k)}, \hat{A}_i^{(k-1)}, H_i^{(k)}$ respectively. This update can be calculated explicitly by using *soft-thresholding operator* [3]:

$$(a^t)^{(k)} = \max\{\|(b^t)^{(k)}\|_2 - \rho, 0\} \frac{(b^t)^{(k)}}{L_i^{(k-1)} \|(b^t)^{(k)}\|_2}, \tag{3.13}$$

where

$$(b^t)^{(k)} = L_i^{(k-1)} (\hat{a}^t)^{(k-1)} - (h^t)^{(k)}.$$

Now let's investigate how to solve (3.8) when $p = \infty$. Similar to the discussion above, suppose $a^t, (a^t)^{(k)}, (\hat{a}^t)^{(k-1)}, (h^t)^{(k)}$ are the t-th column of $A_i, A_i^{(k)}, \hat{A}_i^{(k-1)}, H_i^{(k)}$ respectively, and the subproblem can be transformed as

$$\begin{aligned}(a^t)^{(k)} &= \underset{a^t}{\operatorname{argmin}} \langle (h^t)^{(k)}, a^t - (\hat{a}^t)^{(k-1)} \rangle + \frac{L_i^{(k-1)}}{2} \|a^t - (\hat{a}^t)^{(k-1)}\|^2 + \rho \|a^t\|_\infty \\ &= \underset{a^t}{\operatorname{argmin}} \frac{L_i^{(k-1)}}{2} \left\| a^t - (\hat{a}^t)^{(k-1)} + \frac{(h^t)^{(k)}}{L_i^{(k-1)}} \right\|^2 + \rho \|a^t\|_\infty. \end{aligned} \tag{3.14}$$

This update can be performed by resorting to the following lemma.

Lemma 3.2. *Suppose* $a^* = \underset{a \in \mathbf{R}^m}{\operatorname{argmin}} \frac{1}{2}\|a - b\|^2 + \gamma \|a\|_\infty$, *then we have*

$$a_i^* = \begin{cases} b_i & \text{if} \quad |b_i| \le y^* \\ y^* & \text{if} \quad |b_i| > y^* \end{cases} \tag{3.15}$$

for $i = 1, 2, \ldots, m$, *where* $y^* = \underset{y \ge 0}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{m} [(|b_i| - y)_+]^2 + \gamma y$, *and* $(x)_+ = \max(x, 0)$.

Proof. We first observe that if $\|a\|_\infty$ is determined say $\|a\|_\infty = y$, to minimize $\|a - b\|^2$ we can choose a_i (the ith component of vector a) such that the following holds

$$\|a - b\|^2 = \sum_{i=1}^{m} [(|b_i| - y)_+]^2. \tag{3.16}$$

Therefore, the target problem can be further converted to the following,

$$\begin{aligned} \min \quad & g(y) \triangleq \tfrac{1}{2} \sum_{i=1}^{m} [(|b_i| - y)_+]^2 + \gamma y \\ \text{s.t.} \quad & y \ge 0 \end{aligned} \tag{3.17}$$

Taking the derivative of the $g(y)$, we have

$$\nabla g(y) = \gamma - \sum_{i=1}^{m}(|b_i| - y)_+ = \gamma - \sum_{i\in I}(|b_i| - y), \text{ where } I = \{i \,|\, y \le |b_i|\} \tag{3.18}$$

Notice that $\nabla g(y)$ is nondecreasing function for $y \ge 0$, and when $y = 0$, $\nabla g(y) = \gamma - \sum_{i=1}^{m} |b_i|$. If we denote the y^* to be the optimal solution of (3.17), then we have the following

$$\begin{cases} y^* = 0 & \text{if} \quad \gamma \ge \sum_{i=1}^{m} |b_i| \\ y^* = \frac{\sum_{i\in I}|b_i| - \gamma}{|I|} & \text{if} \quad \gamma < \sum_{i=1}^{m} |b_i| \end{cases}. \tag{3.19}$$

Once we obtained the optimal y^*, based on (3.16) we can easily get the corresponding optimal solution a^* of the subproblem as below

$$a_i^* = \begin{cases} b_i & \text{if} \quad |b_i| \le y^* \\ y^* & \text{if} \quad |b_i| > y^* \end{cases} \tag{3.20}$$

□

Although the closed solution derived in Lemma 3.2 can also be found in [4], we keep the argument above for the convenience of the reader. Finally, we can apply the result to solving the subproblem (3.14) by letting $a = a^t$, $b = (\hat{a}^t)^{(k-1)} - \frac{(h^t)^{(k)}}{L_i^{(k-1)}}$, and $\gamma = \rho/L_i^{(k-1)}$ in the lemma's setting.

3.3.3 Detailed algorithm

Finally, we are able to summarize our BCD methods for tensor CP low rank decomposition in following table.

Algorithm 1: BCD method for solving tensor low rank decomposition

Input N-way tensor $\mathcal{X}$.
Output factor matrices $A_1, \cdots, A_N$.
Initialization randomly generate $A_1^0, \cdots, A_N^0$, with appropriate sizes.
for $k = 1, 2, \cdots,$ **do**
  for $n = 1, 2, \cdots,$ **do**
  • Compute $P_n^{(k-1)}$, $L_n^{(k-1)}$, $\omega_n^{(k-1)}$, and $H_n^{(k)}$ according to (3.9)–(3.12) respectively.
  • Let $\hat{A}_n^{(k-1)} = A_n^{(k-1)} + \omega_n^{(k-1)}(A_n^{(k-1)} - A_n^{(k-2)})$.
  • Update $A_n^{(k)}$ according to (3.8) with the closed form in (3.13) or (3.15) .
  end for
  if stopping criterion is satisfied **then**
    Return $A_1^{(k)}, \cdots, A_N^{(k)}$.
  end if
end for

4 Tensor Low Rank completion

In this section, we show that our algorithm for CP low-rank approximation can be modified to solve both CP and Tucker low-rank completion problems, which is to recover low-rank tensor from partial data. It is known that many problems in signal possessing, computer vision and MRI can be formulated into the completion problems [2,10], since sometimes part of data are missing by various reasons. There have already been some algorithms for tensor completion problems [2,10,15], but we didn't observe a method work directly on the tensor itself. Specifically, in the previous literature, the authors first unfold the tensor into some related matrix by rearranging the positions of the elements and then consider the completion problem of the resulting matrix. For instance, Liu et al [10] and Gandy et al [15] studied the low-n-rank recovery of a tensor, which is the average of the rank of all mode matrices. However, the relationship between the n-rank and the CP rank is still unclear. Recently, Jiang et al. [6] showed that for a super-symmetric tensor, it is rank one in the sense of CP if and only if its square unfolding matrix is also rank one. To our best knowledge, the work in this paper is the first attempt to study CP rank completion problem and our method can work with Tucker rank as well. Numerical results in next section show that our algorithms can recover missing data from only a small amount of samples.

4.1 CP low-rank completion

One formulation of CP low-rank completion problem is given by

$$\min_{A_1,A_2,\cdots,A_N} \frac{1}{2}\|\mathcal{P}_\Omega(\mathcal{X} - [\![A_1, A_2, \cdots, A_N]\!])\|^2 + \rho\,\mathrm{rank}_{CP}([\![A_1, A_2, \cdots, A_N]\!]) \tag{4.1}$$

where $\Omega \subset [I_1] \times [I_2] \cdots \times [I_N]$ is the index set of the observed entries of $\mathcal{X}$ and $\mathcal{P}_\Omega(\mathcal{X})$ keeps the entries of $\mathcal{X}$ in Ω and sets the remaining elements to zero. Again, we take advantage of the relationship between group sparsity and low CP rank structure, and end up with the following problem:

$$\begin{aligned}\min_{A_1,A_2,\cdots,A_N} &\frac{1}{2}\|\mathcal{P}_\Omega(\mathcal{X} - [\![A_1, A_2, \cdots, A_N]\!])\|^2 \\ &+ \rho\left(\sum_{i=1}^{N}\sum_{t=1}^{R}\|a^{i,t}\|_2\right) \triangleq f(A_1, A_2, \cdots, A_N) + \rho\left(\sum_{i=1}^{N} r_i(A_i)\right)\end{aligned} \tag{4.2}$$

Since $\mathcal{P}_\Omega$ is essentially a linear mapping, it is easy to verify that the above problem fit into the regularized multiconvex optimization framework. Indeed, Algorithm 1 can be modified to solve (4.2). The only difference lies in computing the gradient of function $f(\cdot)$ which involves projection on the index set Ω. To calculate the partial gradient with respect to A_n, we notice that

$$f(A_1, A_2, \cdots, A_N) = \frac{1}{2}\|\mathcal{P}_{\Omega(n)}(A_n(A_N \odot \cdots A_{n+1} \odot A_{n-1} \cdots A_1)^\top - X_{(n)})\|^2$$

where $\Omega(n)$ is the mode-n unfolding of Ω. After careful examination, we have the gradient for factor matrices:

$$\nabla_{A_n} f = \mathcal{P}_{\Omega(n)}(A_n(A_N \odot \cdots A_{n+1} \odot A_{n-1} \cdots A_1)^\top - X_{(n)})\,(A_N \odot \cdots A_{n+1} \odot A_{n-1} \cdots A_1).$$

We refer the reader [16] for more about projection index set. After applying Algorithm 1 with new gradient, we get the final solution.

4.2 Tensor Tucker completion

In Tucker low-rank completion problem, we want to obtain a low Tucker rank tensor that approximate the original tensor well from partial information. Since the Tucker rank is a vector, the low Tucker rank means that the values for all components of this vector are small. Following the discussion of group sparsity and Tucker low-rank effect, we arrive at the following formulation:

$$\begin{aligned}&\min_{\mathcal{G},A_1,A_2,\cdots,A_N} \frac{1}{2}\|\mathcal{P}_\Omega(\mathcal{X}-\mathcal{G}\times_1 A_1\times_2 A_2\cdots\times_N A_N)\|^2+\rho\left(\sum_{i=1}^{N}\sum_{t_i=1}^{R_i}\|a^{i,t_i}\|_2\right)\\ \triangleq\ & f(\mathcal{G};A_1,A_2,\cdots,A_N)+\rho\left(\sum_{i=1}^{N} r_i(A_i)\right).\end{aligned} \tag{4.3}$$

Here the first term is the standard tensor completion formulation, and the second term is the penalty term which can lead us to the low rank effect.

In fact, the idea and algorithm in CP low-rank decomposition and completion can be used for solving Tucker low-rank completion. For instance, if we want to apply Algorithm 1, the update for the factor matrices $A_1,\cdots,A_N$ is almost the same as the CP case, except we have different gradient here:

$$\nabla_{A_n} f=\mathcal{P}_{\Omega(n)}(A_nG_{(n)}(A_N\otimes\cdots A_{n+1}\otimes A_{n-1}\cdots A_1)^\top-X_{(n)})(A_N\otimes\cdots A_{n+1}\otimes A_{n-1}\cdots A_1)G_{(n)}^\top,$$

and accordingly

$$P_n^{(k-1)}=(A_N^{(k-1)}\otimes\cdots A_{n+1}^{(k-1)}\otimes A_{n-1}^{(k)}\cdots A_1^{(k)})(G_{(n)}^{(k-1)})^\top,$$

where $G_{(n)}$ is the mode-n unfolding of core tensor $\mathcal{G}$.

Different from CP situation, Tucker low-rank completion requires to update the core tensor after $A_1,\cdots,A_n$ have been updated in each iteration. One should notice that updating the core tensor $\mathcal{G}$ involves a lot of computational details. The reason is that the extrapolation way for updating factor matrices (3.8) will make $A_1,\cdots,A_N$ no longer orthonormal. Therefore, no closed form of $\mathcal{G}$ can be written out. In our approach, we need to matricize the core tensor into a matrix, but at this time it doesn't matter which mode we choose to unfold the core tensor. For simplicity, we use 1-mode unfolding $G_{(1)}$ to update G. Specifically,

$$f(G_{(1)})=\frac{1}{2}\|\mathcal{P}_{\Omega(1)}(A_1G_{(1)}(A_N\otimes\cdots A_3\otimes A_2)^\top-X_{(1)})\|^2.$$

Then

$$\nabla_{G_{(1)}} f=A_1^\top\mathcal{P}_{\Omega(1)}(A_1G_{(1)}(A_N\otimes\cdots A_3\otimes A_2)^\top-X_{(1)})(A_N\otimes\cdots A_3\otimes A_2)$$

We take $L_G^{(k-1)}=\|(A_N^{(k)})^\top A_N^{(k)}\|\times\cdots\|(A_1^{(k)})^\top A_1^{(k)}\|$. In addition, let $\hat{G}_{(1)}^{(k-1)}=G_{(1)}^{(k-1)}+\omega^{(k-1)}(G_{(1)}^{(k-1)}-G_{(1)}^{(k-2)})$, where $\omega^{(k-1)}$ is defined as (3.11). So

$$\hat{H}_{G_{(1)}}^{(k)}=(A_1^{(k)})^\top\mathcal{P}_{\Omega(1)}(A_1^{(k)}\hat{G}_{(1)}^{(k-1)}(A_N^{(k)}\otimes\cdots A_3^{(k)}\otimes A_2^{(k)})^\top-X_{(1)})(A_N^{(k)}\otimes\cdots A_3^{(k)}\otimes A_2^{(k)})$$

would be the gradient of $f(\hat{G}_{(1)}^{(k-1)})$. Then, we are able to derive new $G_{(1)}^{(k)}$ by

$$G_{(1)}^{(k)}=\hat{G}_{(1)}^{(k-1)}-\frac{\hat{H}_{G_{(1)}}^{(k)}}{L_G^{(k-1)}} \tag{4.4}$$

Finally, we transform $G_{(1)}^{(k)}$ back to the core tensor tensor $G^{(k)}$ and finish the updating. We refer the reader [17] for more about core tensor update.

5 Numerical Results

In this section, we test our algorithms for 3-way tensor low-rank decomposition and completion. All tests are performed with Tensor Toolbox of version 2.5 [1]. Our algorithms are terminated whenever the condition $\|V_k - V_{k+1}\| < tol = 0.005$ holds, where V_k is either the objective value of (3.2) for the decomposition problem or objective value of (4.2) and (4.3) for the completion problem in the kth iteration.

In the upcoming tables, we refer *Deviation* to the relative error from the original tensor, *Rank*/*R*. to the rank of the recovered tensor, *Iter* to the iteration number and *Time*/*T* to the running time of solving the problem. In the tables for completion problem, SR is referred to the percentage of the sampling data of the original tensor $\mathcal{X}$.

5.1 Tensor low-rank decomposition

For CP low-rank decomposition associated with 3-way tensor, we initially randomly generate a tensor $\mathcal{X}$ which has the dimension of $30 \times 30 \times 30$, i.e. $\mathcal{X} \in \mathbf{R}^{30\times30\times30}$. Moreover, we set the CP rank of $\mathcal{X}$ to be 4 which is reasonably low compared with the dimension of the given tensor $\mathcal{X}$. To justify our algorithm for the low-rank decomposition, we assume that we are absolutely ignorant about the initial rank of the $\mathcal{X}$ which had been set as 4. Without this knowledge, we simply begin with a randomly chosen tensor $\hat{\mathcal{X}} \in \mathbf{R}^{30\times30\times30}$, and its CP rank is 20, namely the three factor matrices $A, B, C \in \mathbf{R}^{30\times20}$. As we have proposed two different types of group sparsity regularization, namely $L_{1,2}$-norm and $L_{1,\infty}$-norm. We have applied both of them in our test, and expect that the CP rank of the final solution $\mathcal{X}^*$ that returned by our algorithm will be as small as 4. The results are provided in Table 1, in which we find the CP rank is very close to 4 and the deviation is also acceptably small. It shows both the $L_{1,2}$-norm and $L_{1,\infty}$-norm regularization algorithms for the decomposition can obtain a quite accurate tensor with respect to the original tensor $\mathcal{X}$ and possess the low rank property as well. Moreover, we can find that regarding the accuracy and processing time, the $L_{1,2}$-norm regularization slightly outperforms the $L_{1,\infty}$-norm regularization. Thus we would only focus on $L_{1,2}$-norm regularization in the remaining tests.

We also conduct another experiment comparing our algorithm of BCD scheme with the traditional ALS algorithm. In this comparison, we first randomly generate a tensor $\mathcal{X}$ which belongs to $\mathbf{R}^{20\times20\times20}$. We set the CP rank of the original tensor $\mathcal{X}$ to be within a relatively low random level which is between 2 and 8, so we do not know the exact rank of the tensor. As the previous experiment suggests, we expect our algorithm can give us a low rank decomposition. We also run ALS with the rank of either 4 or 20. Moreover, we can add a post-processing procedure to our algorithm, that is we run the ALS algorithm based on the rank that return by our algorithm when it terminated. We expect this procedure can further enhance the accuracy of our approximation, and denote this approach as ALSBCD in our table. The numerical results are summarized in Table 2. We can see that the ALS can not give us a low rank approximation, it either fails to get low rank or fails to get a satisfactory accuracy. However, combining ALS with our algorithm can indeed improve the accuracy.

5.2 Tensor low-rank completion

This subsection contributes to testing our algorithms for 3-way tensor completion problems. The general setting is quite similar to that in low-rank decomposition. As for the CP low rank completion, we again randomly generate an original tensor $\mathcal{X}$ which has the dimension of $30 \times 30 \times 30$ i.e. $\mathcal{X} \in \mathbf{R}^{30\times30\times30}$. We also set the CP rank of the original tensor $\mathcal{X}$ to be 4. Then we randomly obliterate some certain amount of elements of the tensor, so we get a tensor with some missing data. To run our algorithm, we simply began with a randomly chosen tensor $\tilde{\mathcal{X}} \in \mathbf{R}^{30\times30\times30}$ associated with the CP rank of 20. We hope the final result $\mathcal{X}^*$ obtained by our algorithm will be as close as possible to the original tensor $\mathcal{X}$.

For Tucker low-rank completion, we randomly generate an original tensor $\mathcal{Y}$ which has the dimension of $15 \times 15 \times 15$ i.e. $\mathcal{Y} \in \mathbf{R}^{15\times15\times15}$. The Tucker rank of this tensor is set to be $(3, 3, 3)$. Like the situation of CP completion, we assume there are some missing data in the original tensor $\mathcal{Y}$ and expect our algorithm will make the final result $\mathcal{Y}^*$ as close as possible to the original tensor $\mathcal{Y}$.

From the Table 3 and Table 4, we see that the deviation is small when $SR = 0.3$ and much smaller when $SR = 0.6$. It means that the performance of CP and Tucker low completion with our method is quite reliable. One may notice that it cost more CPU time for Tucker completion. This is because the core tensor update requires more effort than the matrix update. Nevertheless, since our method directly deals with the Tucker rank, it provides an way to recover a Tensor with low Tucker rank.

Recently, Liu, et al [10] proposed tensor completion using the matrix nuclear norm as a convex program.

$$\min_{\mathcal{X}} \sum_{n=1}^{N} \alpha_n \|X_{(n)}\|_* \quad \text{subject to} \quad \mathcal{P}_\Omega(\mathcal{X}) = \mathcal{P}_\Omega(\mathcal{M}) \tag{5.1}$$

where α_n are pre-determined weights satisfying $\sum_n \alpha_n = 1$ and $\|A\|_*$ is the nuclear norm of A. They proposed three algorithms to solve the above problem and its relaxed versions, including simple low-rank tensor completion (SiLRTC), fast low-rank tensor completion (FaLRTC), and high accuracy low-rank tensor completion (HaLRTC). Among those three, FaLRTC is the most stable and efficient algorithm. So we compare the Tucker Low Rank Completion with FaLRTC on a three-way tensor. Each tensor is of size $15 \times 15 \times 15$ and is generated with Tucker rank $3 \times 3 \times 3$. The sample ratios are set as 0.3 and 0.7. The performances of two methods are shown in Table 5. It suggests that our algorithm has higher accuracy when the sample ratio is large for the Tucker rank structure though it consumes more time. Investigating the efficiency of the implementation for Tucker Low Rank Completion is one of our future directions.

Moreover, we also compare the CP low rank completion with the Tucker low rank completion when they are applied to the same tensor. Namely, at each instance, we generate one original tensor, and then we applied both methods for tensor completion problems to that particular tensor. The original tensor we generate is either low CP rank or low Tucker rank, and we expect the performances of those two methods are distinct. We can find the numerical results in Table 6, where the Type of Ori. denotes the type of the low-rank structure of the original tensor. In fact, as we expected, the result shows that the methods are more suitable for the tensor which has the corresponding structure, i.e., when the tensor has CP low rank structure it is more preferable to use the CP low rank completion method. The same logic applied to the Tucker case.

Among all those numerical tests, the regularization parameter ρ appears in both decomposition and completion models. If ρ is too large, the rank of the obtained tensor will be

very low, but the error of approximation would be very high, and vise versa when ρ is too small. But there is not a clear and systematic way of choosing the parameter. In fact, a proper parameter ρ would vary from case to case. As a result, for different instances, the tuning process of ρ will consist of multiple trials of different values, and for that tuning process, one may choose the bisection method to perform the trials.

6 Conclusion

In this paper, we put forward a method of finding CP low-rank decomposition with its application of CP/Tucker completion without knowing the rank. Our main idea is to take advantage of the group sparsity and to apply BCD and MBI scheme to our problem. Based on the group sparsity, we formulate our problem as an regularized multi-convex problem, in which we applied the BCD/MBI methods. By utilizing an updating scheme proposed by Y. Xu et al [17], we can solve our problem efficiently. All the implementation details of our algorithms have been discussed. The numerical performances have shown that our approach of using group sparsity is quite reasonable and effective, especially when the given tensor has a low-rank structure. We believe our approach will have a lot of important real applications in different fields.

References

[1] B.W. Bader and T.G. Kolda, MATLAB Tensor Toolbox Version 2.5, Available online, 2012, http://www.sandia.gov/tgkolda/TensorToolbox/.

[2] D. Goldfarb and Z. Qin, Robust Low-Rank Tensor Recovery: Models and Algorithms, *SIAM J. Matrix Anal. Appl.* 35 (2014) 225–253.

[3] E. van den Berg, M. Schmidt, M. Friedlander,and K. Murphy, Group sparsity via linear-time projection, *Technical Report TR-2008-09*, Department of Computer Science, University of British Columbia, 2008.

[4] F. Bach, R. Jenatton, J. Mairal and G. Obozinski, Structured sparsity through convex optimization, *Statist. Sci.* 27 (2012) 447–608.

[5] J. Hastad, Tensor rank is NP-complete, *J. Algorithms* 11(1990) 644–654.

[6] B. Jiang, S. Ma and S. Zhang, Tensor principal component analysis via convex optimization, *Math. Program.*, published online, DOI: 10.1007/s10107-014-0774-0, 2014.

[7] J. Jiang, H. Wu, Y. Li and R. Yu, Three-way data resolution by alternating slice-wise diagonalization (ASD) method, *J. Chemometrics* 14 (2000) 15–36.

[8] J.B. Kruskal, Rank, Decomposition and uniqueness for 3-way and N-way arrays, *Mult. Data Anal.*, 1989.

[9] J.D. Carroll and J.J. Chang, Analysis of individual differences in multidimensional scaling via an N-way generalization of Eckart-Young decomposition, *Psychometrika* 35 (1970) 283–319.

[10] J. Liu, P. Musialski, P. Wonka, and J. Ye, Tensor completion for estimating missing values in visual data, *IEEE Trans. Pattern Anal. Mach. Intell.* 25 (2013) 208–220.

[11] L. Meier, S. Geer and P. Bählmann, The group lasso for logistic regression, *J. R. Stat. Soc. Ser. B Stat. Methodol.* 70 (2008) 53–71.

[12] M. Yuan and Y. Lin, Model selection and estimation in regression with grouped variables, *J. R. Stat. Soc. Ser. B Stat. Methodol.* 68 (2006) 49-67¿

[13] P. Paatero, A weighted non-negative least squares algorithm for three-way PARAFAC factor analysis, *Chemometr. Intell. Lab.* 38 (1997) 223–242.

[14] R.A. Harshman, Foundations of the PARAFAC procedure: Models and conditions for an explanatory multi-modal factor analysis, *UCLA Working Papers in Phonetics* 16 (1970) 1–84.

[15] S. Gandy, B. Recht and I. Yamada, Tensor completion and low-n-rank tensor recovery via convex optimization, *Inverse Problems*, 27 (2011).

[16] T.G. Kolda and B.W. Bader, Tensor decompositions and applications, *SIAM Rev.* 51 (2009) 455–500.

[17] Y. Xu and W. Yin, A block coordinate descent method for multi-convex optimization with applications to nonnegative tensor factorization and completion, *SIAM J. Imaging Sci.* 6 (2013) 1758–1789.

Manuscript received 10 May 2014
revised 16 November 2014, 13 January 2014
accepted for publication 13 January 2014

No.	$L_{1,\infty}$				$L_{1,2}$			
	Deviation	Rank	$Iter_{L_{1,\infty}}$	$Time_{L_{1,\infty}}$	Deviation	Rank	$Iter_{L_{1,2}}$	$Time_{L_{1,2}}$
parameter: l=m=n=30, t=4, r=20								
1	2.690e-003	4	3013	56.77	9.764e-004	5	1878	16.12
2	3.038e-003	4	1901	45.52	9.230e-004	4	2690	26.26
3	3.894e-003	4	2034	48.55	1.222e-003	4	1019	11.33
4	4.473e-003	4	2270	54.52	1.366e-003	4	996	12.10
5	4.160e-003	4	2118	52.07	1.303e-003	4	1289	12.11
6	3.916e-003	4	2863	65.63	1.416e-003	4	1240	12.14
7	3.824e-003	4	2329	68.12	1.214e-003	4	1115	9.20
8	3.038e-003	4	1901	48.62	9.230e-004	4	2690	31.14
9	2.745e-003	4	2793	67.45	8.617e-004	4	1278	14.53
10	2.690e-003	4	3013	110.11	9.764e-004	5	1878	17.84

Table 1: CP low-rank decomposition through $L_{1,\infty}$ and $L_{1,2}$-norm regularization

No.	BCD				ALS				ALSBCD			
	Deviation	R.	$Iter_{BCD}$	T_{BCD}	Deviation	R.	$Iter_{ALS}$	T_{ALS}	Deviation	R.	$Iter_{AB}$	T_{AB}
parameter: l=m=n=20, t=2+unidrnd(6), r=25												
1	9.68e-005	7	10963	147.56	5.77e-001	4	11	0.56	4.32e-006	7	580	6.89
2	8.80e-005	7	9034	124.03	5.77e-001	4	12	0.21	3.67e-006	7	31	0.47
3	1.08e-004	8	11064	145.89	5.59e-001	4	15	0.21	1.55e-005	8	12	0.25
4	9.69e-005	7	8040	109.42	5.50e-001	4	15	0.19	1.75e-005	7	14	0.26
5	8.43e-005	8	8203	111.75	5.27e-001	4	18	0.22	5.69e-006	8	14	0.28
1	8.36e-005	5	11526	116.70	2.52e-005	20	7	0.27	1.43e-005	5	16	0.16
2	7.79e-005	5	10656	108.14	2.87e-005	20	9	0.29	6.99e-006	5	16	0.15
3	1.06e-004	5	11735	117.70	2.06e-005	20	9	0.39	8.30e-006	5	255	1.81
4	1.59e-004	5	8740	87.96	4.91e-005	20	9	0.34	8.05e-006	5	14	0.14
5	8.26e-005	5	11125	110.74	6.69e-005	20	14	0.41	1.06e-005	5	541	4.42

Table 2: BCD vs ALS vs ALSBCD for CP low rank decomposition

No.	Deviation	Rank	Iter	Time
	SR=30%(70% missing data)			
1	1.538e-004	5	13820	91.00
2	1.575e-004	4	9858	65.38
3	2.622e-004	4	9667	62.02
4	1.410e-004	6	14700	92.64
5	4.180e-004	4	11161	75.46
	SR=60%(40% missing data)			
1	7.056e-005	5	16649	111.08
2	7.608e-005	4	11159	74.27
3	1.621e-004	6	10529	70.15
4	5.812e-005	6	17917	115.55
5	1.827e-004	4	11153	73.95

Table 3: CP low-rank Completion

No.	Deviation	Rank	Iter	Time
	SR=30%(70% missing data)			
1	1.324e-001	(6,5,6)	8135	177.13
2	2.895e-002	(6,5,5)	16307	354.24
3	3.103e-002	(3,3,4)	19462	397.20
4	3.892e-002	(4,4,6)	12495	266.51
5	2.371e-002	(4,6,4)	20234	434.07
	SR=60%(40% missing data)			
1	1.715e-002	(3,4,4)	24316	489.06
2	1.572e-002	(7,5,4)	16732	315.63
3	9.815e-003	(4,3,5)	14372	324.10
4	1.524e-002	(4,3,3)	22248	486.58
5	2.505e-002	(5,5,3)	14877	395.43

Table 4: Tucker low-rank Completion

No.	Time	Deviation	Time	Deviation
	SR=30%(70% missing data)			
	Tucker Completion		FaLRTC	
1	395.04	1.217e-002	0.46	1.211e-002
2	414.51	1.535e-002	0.34	1.304e-002
3	370.89	1.205e-002	0.37	9.061e-003
4	563.35	1.049e-002	0.37	1.054e-002
5	451.35	9.879e-003	0.35	9.236e-003
	SR=70%(30% missing data)			
	Tucker Completion		FaLRTC	
1	448.82	4.558e-002	0.61	9.347e-002
2	236.81	5.278e-002	0.63	1.421e-001
3	466.32	6.797e-002	0.60	3.873e-001
4	375.61	6.041e-002	0.70	2.996e-001
5	310.64	5.724e-002	0.61	3.073e-001

Table 5: Tucker low-rank Completion vs FaLRTC

Type of Ori.	No.	Deviation	Rank	Iter	Time	Deviation	Rank	Iter	Time
	SR=30%(70% missing data)								
		CP Completion				Tucker Completion			
CP low rank structure	1	1.194e-003	4	2122	4.55	2.377e-001	(6,7,6)	12156	323.38
	2	2.139e-003	4	3469	7.26	2.864e-001	(4,5,6)	19206	513.46
	3	1.895e-003	4	3087	6.45	3.100e-001	(7,5,5)	11192	307.2
Tucker low rank structure	1	1.193e-001	6	1757	3.96	3.038e-002	(9,8,9)	13751	367.07
	2	2.997e-001	5	250	0.57	6.053e-002	(6,9,8)	13927	372.72
	3	7.678e-002	7	4878	10.42	2.473e-002	(7,7,5)	16527	442.68
	SR=60%(40% missing data)								
CP low rank structure	1	6.393e-004	4	2989	6.33	1.860e-001	(7,9,6)	10074	262.64
	2	1.083e-003	5	2043	4.38	2.247e-001	(5,5,3)	11803	338.47
	3	8.944e-004	4	2989	6.54	3.056e-001	(5,3,4)	11227	303.99
Tucker low rank structure	1	4.197e-002	7	654	1.48	1.391e-002	(6,8,6)	17823	476.39
	2	3.208e-002	7	1881	4.10	7.917e-003	(6,7,5)	20768	556.28
	3	6.877e-002	6	1006	2.18	9.922e-003	(9,6,8)	13953	443.61

Table 6: CP Completion vs Tucker Completion

Xiang Gao
Department of Industrial and Systems Engineering
University of Minnesota, Minneapolis
MN 55455, USA
E-mail address: gaoxx460@umn.edu

Bo Jiang
Research Center for Management Science and Data Analytics
School of Information Management and Engineering
Shanghai University of Finance and Economics
Shanghai 200433, China
E-mail address: isyebojiang@163.com

Shaozhe Tao
Department of Industrial and Systems Engineering
University of Minnesota, Minneapolis
MN 55455, USA
E-mail address: taoxx120@umn.edu