Journal of Mathematical Research with Applications
May, 2017, Vol. 37, No. 3, pp. 367–378
DOI:10.3770/j.issn:2095-2651.2017.03.014
Http://jmre.dlut.edu.cn

Augmented Lagrangian Alternating Direction Method for Tensor RPCA

Ruru HAO, Zhixun SU*
Department of Mathematics, Dalian University of Technology, Liaoning 116024*, P. R. China*

Abstract Tensor robust principal component analysis (TRPCA) problem aims to separate a low-rank tensor and a sparse tensor from their sum. This problem has recently attracted considerable research attention due to its wide range of potential applications in computer vision and pattern recognition. In this paper, we propose a new model to deal with the TRPCA problem by an alternation minimization algorithm along with two adaptive rank-adjusting strategies. For the underlying low-rank tensor, we simultaneously perform low-rank matrix factorizations to its all-mode matricizations; while for the underlying sparse tensor, a soft-threshold shrinkage scheme is applied. Our method can be used to deal with the separation between either an exact or an approximate low-rank tensor and a sparse one. We established the subsequence convergence of our algorithm in the sense that any limit point of the iterates satisfies the KKT conditions. When the iteration stops, the output will be modified by applying a high-order SVD approach to achieve an exactly low-rank final result as the accurate rank has been calculated. The numerical experiments demonstrate that our method could achieve better results than the compared methods.

Keywords tensor RPCA; alternating direction method; augmented Lagrangian function; high-order SVD

MR(2010) Subject Classification 90C90; 94A08; 94A12

1. Introduction

A tensor is a multidimensional array. It is the higher order generalization of vector and matrix, which has many applications in information sciences, computer vision, graph analysis, and traffic data analysis. In the real world, as the size and the amount of redundancy of the data increase fast and nearly all of the existing high-dimensional real world data either have the natural form of tensor (e.g., multichannel images) or can be grouped into the form of tensor. Challenges come up in many scientific areas when someone confronts with the high-dimensional real world data.

1.1. Notation and preliminary knowledge

Following [1,2], we use the notations as follows. For vectors we use low-case letters $\mathbf{x}, \mathbf{y}, \ldots$, for matrices we use bold upper-case letters $\mathbf{X}, \mathbf{Y}, \ldots$, while for tensors we use bold calligraphic

Received February 3, 2017; Accepted March 24, 2017
Supported by the National Natural Science Foundation of China (Grant Nos. 61572099; 61320106008; 91230103) and National Science and Technology Major Project (Grant Nos. 2013ZX04005021; 2014ZX04001011).
* Corresponding author
E-mail address: hao@mail.dlut.edu.cn (Ruru HAO); zxsu@dlut.edu.cn (Zhixun SU)

letters $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}, \ldots$. We denote the $(i_1, \ldots, i_N)$-th component of an N-way tensor $\boldsymbol{\mathcal{X}}$ as $x_{i_1 \ldots i_N}$. Like the situation of vector and matrix, the inner product of $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is defined as

$$\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \rangle = \sum_{i_1=1}^{I_1} \cdots \sum_{i_N=1}^{I_N} x_{i_1 \ldots i_N} y_{i_1 \ldots i_N}.$$

The Frobenius norm of $\boldsymbol{\mathcal{X}}$ is defined as $\|\boldsymbol{\mathcal{X}}\|_F = \sqrt{\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{X}} \rangle}$.

A fiber of $\boldsymbol{\mathcal{X}}$ is a vector obtained by fixing all indices of $\boldsymbol{\mathcal{X}}$ except one, and a slice of $\boldsymbol{\mathcal{X}}$ is a matrix by fixing all indices of $\boldsymbol{\mathcal{X}}$ except two. The mode-n matricization (also called unfolding) of $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is denoted as $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times \Pi_{j \neq n} I_j}$, which is a matrix with columns being the mode-n fibers of $\boldsymbol{\mathcal{X}}$ in the lexicographical order. Relating to the matricization process, we define $\mathbf{unfold}_n(\boldsymbol{\mathcal{X}}) = \mathbf{X}_{(n)}$ and $\mathbf{fold}_n$ to reverse the process, i.e., $\mathbf{fold}_n(\mathbf{unfold}_n(\boldsymbol{\mathcal{X}})) = \boldsymbol{\mathcal{X}}$.

The n-rank of an N-way tensor $\boldsymbol{\mathcal{X}}$, denoted as $\mathrm{rank}_n(\boldsymbol{\mathcal{X}})$, is the rank of $\mathbf{X}_{(n)}$, and we define the rank of $\boldsymbol{\mathcal{X}}$ as an array: $\mathrm{rank}(\boldsymbol{\mathcal{X}}) = (\mathrm{rank}(\mathbf{X}_{(1)}), \ldots, \mathrm{rank}(\mathbf{X}_{(N)}))$. Out definition is related to the Tucker decomposition [3]. In this paper, we say $\boldsymbol{\mathcal{X}}$ is (approximately) low-rank if $\mathbf{X}_{(n)}$ is (approximately) low-rank for all n.

1.2. Related work

The recent proposed Robust Principal Component Analysis (RPCA) [4] is an algorithm to decompose a data matrix $\mathbf{X} \in \mathbb{R}^{n_1 * n_2}$ into two components such that $\mathbf{X} = \mathbf{L} + \mathbf{S}$, where $\mathbf{L}$ is a low-rank matrix and $\mathbf{S}$ is a sparse matrix. The RPCA has strong performance guarantees. It is shown that if the singular vectors of $\mathbf{L}$ satisfy some incoherent conditions, $\mathbf{L}$ and $\mathbf{S}$ can be recovered with high probability. RPCA and its extensions have been successfully applied for background modeling [4], image alignment [5], video restoration [6] et al. One major shortcoming of RPCA is that it can only handle 2-way data.

Tensor can be viewed as the extension of matrix. It is a more proper way to express the real world data compared with matrix, for the real world data are generally in multi-dimensional way. In recent years, plenty works based on tensor recovery have been proposed, such as sparsity measure [7], recovery models [8–10], completion [11–15], denoising [16–19]. [20] studied the Tensor Robust Principal Component Analysis (TRPCA) which aims to exactly recover a three-way low-rank tensor corrupted by sparse errors via convex optimization. [21] designed a tensor RPCA model for the task of background subtraction from compressive measurements (BSCM) over the video frames.

1.3. Organization

The rest of the paper is organized as follows. Section 2 shows our tensor robust principal component analysis (TRPCA) model and the alternating minimization algorithm with two different rank-adjusting strategies; the convergence analysis and a novel high-order SVD trick are also included in this section. In Section 3, we compare the experimental results of the proposed method with those of some state-of-the-art methods for image and video separation. Section 4 concludes this paper.

2. Model and algorithm

We introduce a low-rank matrix factorization based model and an augmented Lagrangian alternating direction method to solve the Tensor-RPCA problem with two rank-adjusting schemes. For the convergence analysis part, we claim that any limit point of the iteration is the KKT point of the problem. A high order SVD trick is provided to modify the final result.

2.1. Tensor-RPCA model

We develop an iterative approach to solve the Tensor-RPCA problem. The problem is to separate a low rank tensor $\boldsymbol{\mathcal{L}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and a sparse tensor $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ from their given sum $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, i.e., to recover a low-rank $\boldsymbol{\mathcal{L}}$ and a sprase $\boldsymbol{\mathcal{S}}$ simultaneously such that

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{S}}. \tag{1}$$

As is now well known, $l1$-norm minimization has been used to recover sparse signals in compressive sensing, it is very reasonable for us to propose our tensor approximation model using the following $l1$-norm data fidelity form. Assuming that the n-rank of $\boldsymbol{\mathcal{L}}$ does not exceed a prescribed estimate $\gamma \in \mathbb{R}^N$, we first consider the model:

$$\begin{aligned} \min_{\boldsymbol{\mathcal{L}}} \quad & \|\boldsymbol{\mathcal{T}} - \boldsymbol{\mathcal{L}}\|_1 \\ \text{s.t.} \quad & \text{rank}_n(\boldsymbol{\mathcal{L}}) < \boldsymbol{\gamma} \end{aligned} \tag{2}$$

where $\boldsymbol{\gamma}$ represents an upper bound of the $\text{rank}_n(\boldsymbol{\mathcal{L}})$.

We apply low-rank matrix factorization to each mode unfolding of $\boldsymbol{\mathcal{L}_n}$ by finding matrices $\mathbf{X}_n \in \mathbb{R}^{I_n \times r_n}, \mathbf{Y}_n \in \mathbb{R}^{r_n \times \Pi_{j \neq n} I_j}$ such that $\mathbf{L}_{(n)} \approx \mathbf{X}_n \mathbf{Y}_n$ for $n = 1, \ldots, N$, where r_n is the estimated rank, either fixed or adaptively updated.

$$\begin{aligned} \min_{\mathbf{X}_n, \mathbf{Y}_n, \boldsymbol{\mathcal{L}}} \quad & \|\boldsymbol{\mathcal{T}} - \boldsymbol{\mathcal{L}}\|_1 \\ \text{s.t.} \quad & \mathbf{L}_{(n)} - \mathbf{X}_n \mathbf{Y}_n = 0, n = 1, \ldots, N. \end{aligned} \tag{3}$$

2.2. Alternating minimization

We present an augmented Lagrangian alternating direction method for solving (3). The augmented Lagrangian function of (3) is defined as follows:

$$L_\beta(\mathbf{X}, \mathbf{Y}, \boldsymbol{\mathcal{L}}, \Lambda) = \|\boldsymbol{\mathcal{T}} - \boldsymbol{\mathcal{L}}\|_1 + \sum_{n=1}^{N} \langle \Lambda_n, \mathbf{L}_{(n)} - \mathbf{X}_n \mathbf{Y}_n \rangle + \frac{\beta_n}{2} \|\mathbf{L}_{(n)} - \mathbf{X}_n \mathbf{Y}_n\|_F^2 \tag{4}$$

where each $\beta_n > 0$ is the penalty parameter and $\Lambda_1, \ldots, \Lambda_n$ are the Lagrange multipliers corresponding to the constraints $\mathbf{L}_{(n)} - \mathbf{X}_n \mathbf{Y}_n = 0$, $n = 1, \ldots, N$, and $\langle \mathbf{U}, \mathbf{V} \rangle$ denotes the usual inner product between matrices $\mathbf{U}$ and $\mathbf{V}$ of equal sizes. We denote $(\mathbf{X}_1, \ldots, \mathbf{X}_N)$, $(\mathbf{Y}_1, \ldots, \mathbf{X}_N)$, $(\Lambda_1, \ldots, \Lambda_N)$ and $\beta_1, \ldots, \beta_N$ by $\mathbf{X}$, $\mathbf{Y}$, Λ and β, respectively.

We know that l_1-norm of a tensor is the sum of the absolute values of all its entries, and the matricization of a tensor in any mode can be viewed as a new arrangement of all the entries. So the l_1-norm of the matricization of a certain tensor in any mode is exactly the same as the

l_1-norm of the tensor. So the (4) can be rewritten as follows:

$$L_\beta(\mathbf{X},\mathbf{Y},\boldsymbol{\mathcal{L}},\Lambda)=\sum_{n=1}^{N}\alpha_n\|\mathbf{T}_{(n)}-\mathbf{L}_{(n)}\|_1+\langle\Lambda_n,\mathbf{L}_{(n)}-\mathbf{X}_n\mathbf{Y}_n\rangle+\frac{\beta_n}{2}\|\mathbf{L}_{(n)}-\mathbf{X}_n\mathbf{Y}_n\|_F^2 \tag{5}$$

where α_n, $n=1,\ldots,N$, are weights of the n-th matricization and satisfy $\sum_{n=1}^{N}\alpha_n=1$.

Since solving (5) for $\mathbf{X}$, $\mathbf{Y}$ and $\boldsymbol{\mathcal{L}}$ is difficult, following the idea in the alternating direction method, we choose to minimize the augmented Lagrangian function with respect to each block variable $\mathbf{X}$, $\mathbf{Y}$ and $\boldsymbol{\mathcal{I}}$ one at a time while fixing the other two ones at their latest values, and then update the Lagrange multiplier. We can update the multiplier Λ by the formula:

$$\Lambda^{k+1}=\Lambda^k+\beta(\mathbf{X}^{k+1}\mathbf{Y}^{k+1}-\boldsymbol{\mathcal{I}}^{k+1}).$$

So we can perform the k-th update as:

$$\mathbf{X}^{k+1}=\operatorname*{argmin}_{\mathbf{X}}L_\beta(\mathbf{X},\mathbf{Y}^k,\boldsymbol{\mathcal{L}}^k,\Lambda^k), \tag{6a}$$

$$\mathbf{Y}^{k+1}=\operatorname*{argmin}_{\mathbf{Y}}L_\beta(\mathbf{X}^{k+1},\mathbf{Y},\boldsymbol{\mathcal{L}}^k,\Lambda^k), \tag{6b}$$

$$\boldsymbol{\mathcal{L}}^{k+1}=\operatorname*{argmin}_{\boldsymbol{\mathcal{L}}}L_\beta(\mathbf{X}^{k+1},\mathbf{Y}^{k+1},\boldsymbol{\mathcal{L}},\Lambda^k), \tag{6c}$$

$$\Lambda^{k+1}=\Lambda^k+\gamma\beta(\mathbf{X}^{k+1}\mathbf{Y}^{k+1}-\boldsymbol{\mathcal{L}}^{k+1}). \tag{6d}$$

Note that both (6a) and (6b) can be decomposed into N independent least square problems, which can be solved in parallel. The updates in (6) can be explicitly written as

$$\mathbf{X}_n^{k+1}=(\mathbf{L}_{(n)}^k-\frac{\Lambda_n^k}{\beta_n})(\mathbf{Y}_n^k)^\top\big(\mathbf{Y}_n^k(\mathbf{Y}_n^k)^\top\big)^\dagger,\ n=1,\ldots,N, \tag{7a}$$

$$\mathbf{Y}_n^{k+1}=\big((\mathbf{X}_n^{k+1})^\top\mathbf{X}_n^{k+1}\big)^\dagger(\mathbf{X}_n^{k+1})^\top\big(\mathbf{L}_{(n)}^k-\frac{\Lambda_n^k}{\beta_n}\big),\ n=1,\ldots,N, \tag{7b}$$

$$\boldsymbol{\mathcal{L}}^{k+1}=\sum_{n=1}^{N}\mathbf{fold}_n\mathcal{S}\big((\mathbf{X}_n^{k+1}\mathbf{Y}_n^{k+1}-\frac{\Lambda_n^k}{\beta_n}-\mathbf{T}_{(n)}),\frac{\alpha_n}{\beta_n}\big)+\mathbf{T}_n, \tag{7c}$$

$$\Lambda_n^{k+1}=\Lambda_n^k+\gamma\beta_n(\mathbf{X}^{k+1}\mathbf{Y}^{k+1}-\boldsymbol{\mathcal{L}}^{k+1}),\ n=1,\ldots,N. \tag{7d}$$

In the above equations (7a) and (7b), $\mathbf{A}^\dagger$ denotes the Moore-Penrose pseudo-inverse of $\mathbf{A}$. No matter how $\mathbf{X}_n$ is computed, only the products $\mathbf{X}_n\mathbf{Y}_n, n=1,\ldots,N$, affect $\boldsymbol{\mathcal{L}}$ and thus the final recovery result. Hence, we shall update $\mathbf{X}$ in the following more efficient way

$$\mathbf{X}_n^{k+1}=(\mathbf{L}_{(n)}^k-\frac{\Lambda_n^k}{\beta_n})(\mathbf{Y}_n^k)^\top, n=1,\ldots,N. \tag{8}$$

In (7c), $\mathcal{S}(x,\tau)$ is the soft-threshold operator, and $\mathcal{S}(x,\tau)=\operatorname{sign}(x)\max(|x|-\tau,0)$.

2.3. Rank adjusting schemes

In the iterations mentioned above (6), the rank of the underlying tensor $\boldsymbol{\mathcal{L}}$ is generally unknown, which can be bounded by the size of $\mathbf{X}_n$ and $\mathbf{Y}_n$. A good estimation for the rank of $\boldsymbol{\mathcal{L}}$, denoted by $(\operatorname{rank}(\mathbf{L}_{(1)}),\ldots,\operatorname{rank}(\mathbf{L}_{(N)}))$ is essential to the final results given by the algorithm. A smaller rank estimation can cause underfitting and a large recovery error, while a larger rank estimation can cause overfitting and large deviation to the underlying tensor $\boldsymbol{\mathcal{L}}$. Since we do

not have the priori knowledge of rank($\boldsymbol{\mathcal{L}}$), we provide two schemes to dynamically adjust the rank estimation $r_1, \ldots, r_N$. In our algorithm, we use parameter ξ_n to decide which scheme we adopt. If $\xi_n = -1$, the rank-decreasing scheme is applied to r_n in each iteration; if $\xi_n = 1$, the rank-increasing scheme is applied to r_n; otherwise, r_n is fixed to the initial input.

2.3.1. Rank-decreasing scheme

In this scheme, the initial rank input is an overestimated value, i.e., $r_n > \text{rank}_n(\boldsymbol{\mathcal{L}})$. Following [15,22], we calculate the eigenvalues of $\mathbf{X}_n^\top \mathbf{X}_n$ after each iteration, which are assumed to be sorted in a descent order as $\lambda_1^n \geq \lambda_2^n \geq \cdots \geq \lambda_{r_n}^n$. Then we compute the quotients between the two eigenvalues sequentially $\bar{\lambda}_i^n = \lambda_i^n / \lambda_{i+1}^n$, $i = 1, \ldots, r_n - 1$. Suppose

$$\hat{r}_n = \underset{1 \leq i \leq r_n - 1}{\text{argmax}} \ \bar{\lambda}_i.$$

If

$$\text{gap}_n = \frac{(r_n - 1)\bar{\lambda}_{\hat{r}_n}}{\sum_{i \neq \hat{r}_n} \bar{\lambda}_i} \geq 10, \tag{9}$$

which implies there exists a gap "big" enough between $\lambda_{\hat{r}_n}$ and $\lambda_{\hat{r}_n+1}$, we thus reduce r_n to $\hat{r}_n$. Assume that the SVD of $\mathbf{X}_n \mathbf{Y}_n$ is $\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$. Then we update $\mathbf{X}_n$ to $\mathbf{U}_{\hat{r}_n}\boldsymbol{\Sigma}_{\hat{r}_n}$ and $\mathbf{Y}_n$ to $\mathbf{V}_{\hat{r}_n}^\top$, where $\mathbf{U}_{\hat{r}_n}$ is a submatrix of $\mathbf{U}$ containing $\hat{r}_n$ columns corresponding to the largest $\hat{r}_n$ singular values, and $\boldsymbol{\Sigma}_{\hat{r}_n}$ and $\mathbf{V}_{\hat{r}_n}$ are obtained in a similar way.

In our numerical experiments, we can find that this scheme generally works well when the underlying tensor is exactly a low-rank one. For this kind of tensors, there is a large "gap" which is easy to be identified, the true rank can be typically obtained after just one step of rank decreasing. While for approximately low-rank tensors, a large gap may or may not exist. When a large gap cannot be found by the rule introduced before, the overestimated rank will not decrease. For these tensors, the rank-increasing scheme below has a better performance.

2.3.2. Rank-increasing scheme

Different from the rank-decreasing scheme introduced in the last subsection, this rank-increasing scheme starts with an underestimated rank, i.e., $r_n \leq \text{rank}_n(\boldsymbol{\mathcal{L}})$. Following [14,21], we increase r_n to $\min(r_n + \Delta r_n, r_n^{\max})$ at iteration $k+1$ if

$$\left| 1 - \frac{\left\| \boldsymbol{\mathcal{T}} - \mathbf{fold}_n(\mathbf{X}_n^{k+1}\mathbf{Y}_n^{k+1}) \right\|_F}{\left\| \boldsymbol{\mathcal{T}} - \mathbf{fold}_n(\mathbf{X}_n^{k}\mathbf{Y}_n^{k}) \right\|_F} \right| \leq 10^{-2}, \tag{10}$$

which means the progress of the iteration is very limited in the r_n dimensional space along the n-th mode, so we should increase the rank estimation by Δr_n if the new estimation is smaller than $r_n^{\max}$. Here, Δr_n is a positive integer, and $r_n^{\max}$ is the maximal rank estimate. We adopt the economy QR factorization of $(\mathbf{Y}_n^{k+1})^\top$ to decide how to increase the rank estimation.

Let the economy QR factorization of $(\mathbf{Y}_n^{k+1})^\top$ be $\mathbf{QR}$. We augment $\mathbf{Q} \leftarrow [\mathbf{Q}, \hat{\mathbf{Q}}]$ where $\hat{\mathbf{Q}}$ has Δr_n randomly generated columns and then orthonormalize $\mathbf{Q}$. Next, we update $\mathbf{Y}_n^{k+1}$ to $\mathbf{Q}^\top$ and $\mathbf{X}_n^{k+1} \leftarrow [\mathbf{X}_n^{k+1}, \mathbf{0}]$, where $\mathbf{0}$ is an $I_n \times \Delta r_n$ *zero* matrix. Since we update the variables in the order of $\mathbf{X}, \mathbf{Y}, \boldsymbol{\mathcal{L}}$, appending any matrix of appropriate size after $\mathbf{X}_n$ does not make any difference. Small Δr_n usually gives better solution while needs more time.

In general, if we concern the solution quality, we can choose the rank-increasing scheme, because it works no worse than the rank-decreasing scheme in this term no matter the underlying tensor is exactly low-rank or approximately low-rank. If the convergence rate is a priority, and the underlying tensor is an exactly low-rank one, the rank-increasing scheme usually gives acceptable solutions at the cost of less running time for it takes only one decreasing adjustment.

2.4. Dynamic weights and stopping rules

If we have no prior knowledge about the low-rankness of each mode of the underlying low-rank tensor, we can set the parameters $\alpha_1, \ldots, \alpha_N$ in (3) uniformly to $\frac{1}{N}$ at the beginning. During the iterations, we either fix them or dynamically update them according to the fitting error

$$\mathbf{fit}_n(\mathbf{X}_n\mathbf{Y}_n) = \Big\| \sum_{n=1}^{N} \alpha_n \mathbf{fold}_n(\mathbf{X}_n\mathbf{Y}_n) - \boldsymbol{\mathcal{T}} \Big\|_F.$$

Suppose in some mode, the low-rankness is more significant, weight of the objective in that mode should be increased. It means the smaller $\mathbf{fit}_n(\mathbf{X}_n\mathbf{Y}_n)$ is, the larger α_n should be. Specifically, if the current iterate is $(\mathbf{X}^k, \mathbf{Y}^k, \boldsymbol{\mathcal{Z}}^k)$, we set

$$\alpha_n^k = \frac{\left[\mathbf{fit}_n(\mathbf{X}_n^k\mathbf{Y}_n^k)\right]^{-1}}{\sum_{i=1}^{N}\left[\mathbf{fit}_i(\mathbf{X}_i^k\mathbf{Y}_i^k)\right]^{-1}}, \quad n = 1, \ldots, N. \tag{11}$$

As demonstrated below, dynamic updating α_n's can improve the recovery quality for tensors that have better low-rankness in one mode than others.

The iteration should be terminated if one of the following conditions was satisfied for some k, where k demonstrates the k-th way,

$$\frac{\left|\sum_{n=1}^{N}\mathbf{fit}_n(\mathbf{X}_n^k\mathbf{Y}_n^k) - \sum_{n=1}^{N}\mathbf{fit}_n(\mathbf{X}_n^{k+1}\mathbf{Y}_n^{k+1})\right|}{1+\sum_{n=1}^{N}\mathbf{fit}_n(\mathbf{X}_n^k\mathbf{Y}_n^k)} \le tol, \tag{12}$$

and

$$\frac{\sum_{n=1}^{N}\alpha_n^k \cdot \mathbf{fit}_n(\mathbf{X}_n^{k+1}\mathbf{Y}_n^{k+1})}{\|\boldsymbol{\mathcal{T}}\|_F} \le tol, \tag{13}$$

where *tol* is a small positive value specified below. The relative change of the overall fitting is denoted by the left side of (12) while the weighted fitting is denoted by the left side of (13). When both of them are satisfied, we hold that the performance is good enough.

2.5. Pseudocode

The above discussions are summarized in Algorithm 1. After the algorithm terminates with output $(\mathbf{X}, \mathbf{Y}, \boldsymbol{\mathcal{L}}, \Lambda)$, we adopt $\sum_{n=1}^{N}\alpha_n\mathbf{fold}_n(\mathbf{X}_n\mathbf{Y}_n)$ to estimate the underlying tensor $\boldsymbol{\mathcal{L}}$. During the iterations, we can have different rank-adjusting schemes for different modes by setting different ξ_n. For simplicity we set ξ_n uniformly for all modes in our experiments.

2.6. Convergence analysis

There is no established convergence theory, to the best of our knowledge, for ALADM algorithms applied to non-convex problems or even to convex problems with more than two blocks of variables as we have in Algorithm 1. On the other hand, empirical evidence suggests

Input: Tensor $\boldsymbol{\mathcal{T}}$, and $\alpha_n \geq 0, n = 1, \ldots, N$ with $\sum_{n=1}^{N} \alpha_n = 1$.
Parameters: $r_n, \Delta r_n, r_n^{\max}, \xi_n, n = 1, \ldots, N$.
Initialization: $(\mathbf{X}^0, \mathbf{Y}^0, \boldsymbol{\mathcal{L}}^0, \Lambda^0)$.
for $k = 0, 1, \ldots$ **do**
  $\mathbf{X}^{k+1} \leftarrow$ (8), $\mathbf{Y}^{k+1} \leftarrow$ (7b), $\boldsymbol{\mathcal{L}}^{k+1} \leftarrow$ (7c),and $\Lambda^{k+1} \leftarrow$(7d).
  if *stopping criterion is satisfied* **then**
    Output $(\mathbf{X}^{k+1}, \mathbf{Y}^{k+1}, \boldsymbol{\mathcal{L}}^{k+1})$.
  end
  for $n = 1, \ldots, N$ **do**
    if $\xi_n = -1$ **then**
      Apply rank-decreasing scheme to $\mathbf{X}_n^{k+1}$ and $\mathbf{Y}_n^{k+1}$ in Section 1
    end
    else if $\xi_n = 1$ **then**
      Apply rank-increasing scheme to $\mathbf{X}_n^{k+1}$ and $\mathbf{Y}_n^{k+1}$ in Section 2
    end
  end
end

Algorithm 1 Low-rank Tensor Approximation

that Algorithm 1 has very strong convergence behavior. In this subsection, we give a weak convergence result for Algorithm 1 that under mild conditions any limit point of the iteration sequence generated by Algorithm 1 is a KKT point. Although far from being satisfactory, this result nevertheless provides an assurance for the behavior of the algorithm. The proof of the theorem is inspired by [23]. Further theoretical studies in this direction are certainly desirable.

It is straightforward to derive the KKT conditions for (3):

$$\Lambda_n \mathbf{Y}_n = 0,\ \mathbf{X}_n^\top \Lambda_n = 0,\ \mathbf{L}_{(n)} - \mathbf{X}_n \mathbf{Y}_n = 0,\ \Lambda_n \in \partial_{\mathbf{L}_{(n)}}(\mathbf{L}_{(n)} - \mathbf{T}_{(n)}),\quad n = 1, \ldots, N$$

where, for any $\beta_n > 0$, the last group relation is equivalent to

$$\mathbf{L}_{(n)} - \mathbf{T}_{(n)} - \frac{\Lambda_n}{\beta_n} \in \partial_{\mathbf{L}_{(n)}}(\|\mathbf{T}_{(n)} - \mathbf{L}_{(n)}\|_1) + \mathbf{T}_{(n)} - \mathbf{L}_{(n)} \triangleq \mathcal{Q}_{\beta_n}(\mathbf{T}_{(n)} - \mathbf{L}_{(n)}) \tag{14}$$

with the scalar function $\mathcal{Q}_\beta(t) \triangleq \frac{1}{\beta}\partial|t| + t$ applied element-wise to $\mathbf{T}_{(n)} - \mathbf{L}_{(n)}$. It is easy to verify that $\mathcal{Q}_\beta$ is monotone so that $\mathcal{Q}_\beta^{-1} \triangleq \mathcal{S}(t, \frac{1}{\beta})$. Applying $\mathcal{Q}_\beta^{-1}(\cdot)$ to both sides of (14) and invoking the equation $\mathbf{L}_{(n)} = \mathbf{X}_n \mathbf{Y}_n$, we arrive at:

$$\mathbf{T}_{(n)} - \mathbf{L}_{(n)} = \mathcal{Q}_{\beta_n}^{-1}(\mathbf{L}_{(n)} - \mathbf{T}_{(n)} - \frac{\Lambda_n}{\beta_n}) \equiv \mathcal{S}(\mathbf{T}_{(n)} - \mathbf{L}_{(n)} + \frac{\Lambda}{\beta}, \frac{1}{\beta}) \equiv \mathcal{S}(\mathbf{T}_{(n)} - \mathbf{X}_n \mathbf{Y}_n + \frac{\Lambda}{\beta}, \frac{1}{\beta}).$$

Therefore, the KKT conditions for (3) can be written as: for $\beta_n > 0$, $n = 1, \ldots, N$,

$$\Lambda_n \mathbf{Y}_n = 0,\ \mathbf{X}_n^\top \Lambda_n = 0,\ \mathbf{L}_{(n)} - \mathbf{X}_n \mathbf{Y}_n = 0, \tag{15a}$$

and

$$\mathbf{T}_{(n)} - \mathbf{L}_{(n)} = \mathcal{S}(\mathbf{T}_{(n)} - \mathbf{X}_n \mathbf{Y}_n + \frac{\Lambda_n}{\beta_n}, \frac{1}{\beta_n}). \tag{15b}$$

Theorem 2.1 *Let $\{(\mathbf{X}^k, \mathbf{Y}^k, \boldsymbol{\mathcal{L}}^k)\}$ be a sequence generated by Algorithm 1 with fixed r_n's and fixed positive α_n's. Let $\boldsymbol{\Lambda}^k = \boldsymbol{\mathcal{T}} - \big(\sum_n \alpha_n \cdot \mathbf{fold}_n(\mathbf{X}_n^k \mathbf{Y}_n^k)\big)$. Then any limit point of $\{(\mathbf{X}^k, \mathbf{Y}^k, \boldsymbol{\mathcal{L}}^k, \boldsymbol{\Lambda}^k)\}$ satisfies the KKT conditions in (3).*

Proof It follows from (7), (8), and the identities

$$(\mathbf{X}_n^{k+1} - \mathbf{X}_n^k)\mathbf{Y}_n^k(\mathbf{Y}_n^k)^\top = \Big(\mathbf{L}_{(n)} - \frac{\Lambda_n^k}{\beta_n} - \mathbf{X}_n^k\mathbf{Y}_n^k\Big)(\mathbf{Y}_n^k)^\top, \tag{16a}$$

$$(\mathbf{X}_n^{k+1})^\top\mathbf{Y}_n^{k+1}(\mathbf{Y}_n^{k+1} - \mathbf{Y}_n^k) = (\mathbf{X}^{k+1})^\top\Big(\mathbf{L}_{(n)} - \frac{\Lambda_n^k}{\beta_n} - \mathbf{X}_n^{k+1}\mathbf{Y}_n^k\Big), \tag{16b}$$

$$\mathbf{L}_{(n)}^{k+1} - \mathbf{L}_{(n)}^k = \mathcal{S}\Big(\mathbf{X}_n^{k+1}\mathbf{Y}_n^{k+1} - \mathbf{T}_{(n)} + \frac{\Lambda_n^k}{\beta_n}, \frac{1}{\beta_n}\Big) \tag{16c}$$

$$\Lambda_n^{(k+1)} - \Lambda_n^k = \beta_n(\mathbf{X}_n^{k+1}\mathbf{Y}_n^{k+1} - \mathbf{L}_{(n)}^{k+1}). \tag{16d}$$

Hence a limited point of $\{(\mathbf{X}^k, \mathbf{Y}^k, \boldsymbol{\mathcal{L}}^k, \boldsymbol{\Lambda}^k)\}$ implies that the both sides of (16) all tend to zero as k goes to infinity. Consequently,

$$\Lambda_n^k\mathbf{Y}_n^k \to 0, (\mathbf{X}_n^k)^\top\Lambda_n^k \to 0, \mathbf{L}_{(n)}^k - \mathbf{X}_n^k\mathbf{Y}_n^k \to 0, \tag{17a}$$

and

$$\mathbf{T}_{(n)} - \mathbf{L}_{(n)}^k - \mathcal{S}(\mathbf{T}_{(n)} - \mathbf{X}_n^k\mathbf{Y}_n^k + \frac{\Lambda_n^k}{\beta_n}, \frac{1}{\beta_n}) \to 0 \tag{17b}$$

where the the first limit in (17a) is used to derive other limits. That is, the sequence $\{(\mathbf{X}^k, \mathbf{Y}^k, \boldsymbol{\mathcal{L}}^k, \boldsymbol{\Lambda}^k)\}$ asymptotically satisfies the KKT conditions (15), from which the conclusions of the proposition follow readily. This completes the proof. □

2.7. High-order SVD

After the algorithm terminates with output $(\mathbf{X}, \mathbf{Y}, \boldsymbol{\mathcal{L}}, \Lambda)$, we adopt

$$\boldsymbol{\mathcal{L}} = \sum_{n=1}^{N} \alpha_n \mathbf{fold}_n(\mathbf{X}_n\mathbf{Y}_n)$$

to estimate the underlying tensor $\boldsymbol{\mathcal{L}}$. However, the rank of the output tensor is often full-rank although it is close to the exact underlying tensor. The reason is that the low-rankness of the matricization of the tensor in one mode cannot guarantee the low-rankness of the matricizations in other modes, and the weighted summation cannot even guarantee the low-rankness in any mode.

Fortunately, we have already got the n-rank of the underlying tensor when the algorithm terminates. With this knowledge of $\mathrm{rank}_n(\hat{\boldsymbol{\mathcal{L}}}) = (r_1, \ldots, r_N)$, we can modify the output by the following high-order SVD approach to achieve an exactly low-rank final result. There are three steps in this approach.

Firstly, we can perform the SVD decomposition for each matricization $\mathbf{L}_n$, $n = 1, \ldots, N$, of the tensor $\boldsymbol{\mathcal{L}}$

$$\mathbf{L}_n = \mathbf{U}_n\mathbf{S}_n\mathbf{V}_n^\top, \quad n = 1, \ldots, N. \tag{18}$$

Then we can get a series of orthogonal square matrices $\hat{\mathbf{U}}_n, n = 1, \ldots, N$ by the following formula:

$$\hat{\mathbf{U}}_n = \tilde{\mathbf{U}}_n\tilde{\mathbf{U}}_n^\top \tag{19}$$

where $\tilde{\mathbf{U}}_n$ denotes the first r_n columns of $\mathbf{U}_n$, and the rank of $\hat{\mathbf{U}}_n$, $n = 1, \ldots, N$ is r_n. At last, the exactly low-rank final result can be calculated by the production of tensor $\boldsymbol{\mathcal{L}}$ and the

matrices $\hat{\mathbf{U}}_n$, $n = 1, \ldots, N$:

$$\boldsymbol{\mathcal{L}}_{new} = \boldsymbol{\mathcal{L}} \times_1 \hat{\mathbf{U}}_1 \cdots \times_N \hat{\mathbf{U}}_N. \tag{20}$$

This approach can be summarized by the Algorithm 2.

Input: Approximated Low-rank Tensor $\boldsymbol{\mathcal{L}}$, and the n-rank of the underlying exactly low rank tensor $\hat{\boldsymbol{\mathcal{L}}}$, $\text{rank}_n(\hat{\boldsymbol{\mathcal{L}}}) = (r_1, \ldots, r_N)$.
for $n = 1, \ldots, N$ **do**
  $\mathbf{U}_n \leftarrow$ (18),
  $\hat{\mathbf{U}}_n \leftarrow$ (19),
end
$\boldsymbol{\mathcal{L}}_{new} \leftarrow$(20).

Algorithm 2 High Order SVD

3. Numerical experiments

In this section, we conduct numerical experiments to support our main results. We first investigate the ability of our algorithm for recovering tensors of various n-rank from noises of various sparsity with two rank adjustment schemes. Then we test our algorithm for the background subtraction for surveillance video.

3.1. Exact recovery for synthetic data

For simplicity, we consider the tensors of size $n \times n \times n$ with varying dimension $n = 100, 200$ and 300. We generate the low-rank tensor by multiplying a random core tensor of size $r \times r \times r$ with three random matrices of size $n \times r$, where $r = 0.1 * n$, $\boldsymbol{\mathcal{L}} = \boldsymbol{\mathcal{C}} \times_1 \mathbf{A}_1 \times_2 \mathbf{A}_2 \times_3 \mathbf{A}_3$. The core tensor is generated by MATLAB command randn(r, r, r) and the random matrices are generated by MATLAB command randn(n, r). The support set Ω (with size m) of $\boldsymbol{\mathcal{S}}$ is chosen uniformly at random.

rank-decreasing, $r = 0.1 * n$, $m = \|\boldsymbol{\mathcal{S}}\|_0 = 0.2 * n^3$						
n	r	m	n-rank($\tilde{\boldsymbol{\mathcal{L}}}$)	$\|\tilde{\boldsymbol{\mathcal{S}}}\|_0$	$\|\boldsymbol{\mathcal{L}} - \tilde{\boldsymbol{\mathcal{L}}}\|_F / \|\boldsymbol{\mathcal{L}}\|_F$	$\|\boldsymbol{\mathcal{S}} - \tilde{\boldsymbol{\mathcal{S}}}\|_F / \|\boldsymbol{\mathcal{S}}\|_F$
100	10	2e5	$10 \times 10 \times 10$	175,594	4.1e-7	1.6e-9
200	10	16e5	$20 \times 20 \times 20$	1,525,616	4.4e-7	1.9e-9
300	10	54e5	$30 \times 30 \times 30$	5,424,047	1.3e-6	1.4e-9

Table 1 Recovery results for rank-decreasing scheme

rank-increasing, $r = 0.1 * n$, $m = \|\boldsymbol{\mathcal{S}}\|_0 = 0.2 * n^3$						
n	r	m	n-rank($\tilde{\boldsymbol{\mathcal{L}}}$)	$\|\tilde{\boldsymbol{\mathcal{S}}}\|_0$	$\|\boldsymbol{\mathcal{L}} - \tilde{\boldsymbol{\mathcal{L}}}\|_F / \|\boldsymbol{\mathcal{L}}\|_F$	$\|\boldsymbol{\mathcal{S}} - \tilde{\boldsymbol{\mathcal{S}}}\|_F / \|\boldsymbol{\mathcal{S}}\|_F$
100	10	2e5	$10 \times 10 \times 10$	174,455	4.1e-7	1.6e-9
200	10	16e5	$20 \times 20 \times 20$	1,539,744	4.5e-7	1.9e-9
300	10	54e5	$30 \times 30 \times 30$	5,347,546	1.6e-6	1.5e-9

Table 2 Recovery results for rank-increasing scheme

We test the two rank adjustment schemes on the synthetic data set, similarly as [20]. Table 1 gives the results of the first situation with setting $r = 0.1*n$ and $m = \|\boldsymbol{\mathcal{S}}\|_0 = 0.2*n^3$, equipped with the rank-decreasing scheme. Table 2 gives the results of the second situation with setting $r = 0.1 * n$ and $m = \|\boldsymbol{\mathcal{S}}\|_0 = 0.2 * n^3$, equipped with the rank-increasing scheme.

It can been seen that our algorithm gives the correct rank estimation r of $\|\boldsymbol{\mathcal{L}}\|$ in all cases and the relative errors, $\|\boldsymbol{\mathcal{L}} - \tilde{\boldsymbol{\mathcal{L}}}\|_F/\|\boldsymbol{\mathcal{L}}\|_F$ are less than 10^{-5}. Although the sparsity estimation is not so exact as the rank estimation, we can see that $\|\boldsymbol{\mathcal{S}}\|_0$ is much larger than m, the relative errors$\|\boldsymbol{\mathcal{S}}-\tilde{\boldsymbol{\mathcal{S}}}\|_F/\|\boldsymbol{\mathcal{S}}\|_F$ are all small, even smaller than the relative errors of the recovered low-rank tensor. These results imply the accuracy of our algorithm.

The two tables show that both of the two rank adjusting schemes perform well in our algorithm. The results of rank-decreasing scheme is a little better than that of the rank-increasing scheme, for the "gap" mentioned in Subsection 2.3.

3.2. Background subtraction of surveillance video

We choose a set of real world surveillance video from I2R data set. It is natural to model the background variations as approximately low-rank. We compared our algorithms of two rank adjusting schemes with the two models proposed in [21] which can be regarded as the state-of-art tensor RPCA method. We extract the fourth slice of each recovered low-rank tensor respectively to demonstrate the final results. Of course, the fourth frame of the video "hall" is the input frame, and the ninetieth frame is regarded as the ground truth of the background because there is no people appearing at that moment. Here is the comparison:

(a) Frame 4 (b) Frame 90 (c) H-TenRPCA

(d) PG-TenRPCA (e) TRPCADec (f) TRPCAInc

Figure 1 (a) The fourth frame of the video; (b) The 90th frame of the video; (c) The fourth slice of the result achieved by H-TenRPCA [20]; (d) The fourth slice of the result achieved by PG-TenRPCA [20]; (e) The fourth slice of the result achieved by rank decreasing Scheme 2.3.1; (f) The fourth slice of the result achieved by rank increasing Scheme 2.3.2.

From the figure above we can find a shallow shadow in the middle of the background frame subtracted by the H-TenRPCA and PG-TenRPCA. While in our results (including both of the situation related to the two different schemes), the shadow in the same position is more close to the ground truth. It means that the low-rank tensor recovered by our algorithm is reliable.

4. Conclusion

We have proposed a new algorithm for tensor robust principal component analysis (TR-PCA). Our algorithm utilizes low-rank matrix factorizations to all-mode matricizations of the underlying low-rank tensor, and describe the sparsity of the sparse tensor via l_1 norm. Two rank adjust strategies are provided. We also give the convex analysis of the iterative algorithm in the paper. An HOSVD method is applied to modify the final result. The numerical experiments show that our method consistently generates the best solutions among all compared methods.

However, how to adjust the parameters in the model is not studied in this paper. These problems will be explored in future work.

Acknowledgements We thank the referees for their time and comments.

References

[1] H. A. KIERS. *Towards a standardized notation and terminology in multiway analysis*. Journal of Chemometrics, 2000, **14**(3): 105–122.

[2] T. G. KOLDA, B. W. BADER. *Tensor decompositions and applications*. SIAM Rev., 2009, **51**(3): 455–500.

[3] L. R. TUCKER. *Some mathematical notes on three-mode factor analysis*. Psychometrika, 1966, **31**(3): 279–311.

[4] E. CANDES, Xiaodong LI, Yi MA, et al. *Robust principal component analysis?*. J. ACM, 2011, **58**(3): 1–37

[5] Yigang PENG, A. GANESH, J. WRIGHT, et al. *RASL: Robust alignment by sparse and low-rank decomposition for linearly correlated images*. IEEE Transaction on Pattern Recognition & Machine Intelligence, 2012, **34**(11): 2233–2246.

[6] Hui JI, Sibin HUANG, Zuowei SHEN, et al. *Robust video restoration by joint sparse and low rank matrix approximation*. SIAM J. Imaging Sci., 2011, **4**(4): 1122–1142.

[7] Qian ZHAO, Deyu MENG, Xu KONG, et al. *A novel sparsity measure for Tensor recovery*. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, 271–279.

[8] D. GOLDFARB, Zhiwei QIN. *Robust low-rank tensor recovery: models and algorithms*. SIAM J. Matrix Anal. Appl., 2014, **35**(1): 225–253.

[9] Weisheng DONG, Guangyu LI, Guangming SHI, et al. *Low-Rank Tensor approximation with Laplacian scale mixture modeling for multiframe image denoising*. Proceedings of the IEEE International Conference on Computer Vision, 2015, 442–449.

[10] Bo HUANG, Cun MU, D. GOLDFARB, et al. *Provable low-rank Tensor recovery*. Optimization-Online, 2014, 4252.

[11] Ji LIU, P. MUSIALSKI, P. WONKA, et al. *Tensor completion for estimating missing values in visual data*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013, **35**(1): 208–220.

[12] B. ROMERAPAREDES, M. PONTIL. *A new convex relaxation for tensor completion*. Advances in Neural Information Processing Systems (NIPS), 2013, 2967–2975.

[13] Zemin ZHANG, G. ELY, S. AERON, et al. *Novel methods for multilinear data completion and denoising based on Tensor-SVD*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, 3842–3849.

[14] Xiaoqin ZHANG, Zhengyuan ZHOU, Di WANG, et al. *Hybrid singular value thresholding for Tensor completion*. The AAAI Conference on Artificial Intelligence (AAAI), 2014, 1362–1368.

[15] Yangyang XU, Ruru HAO, Wotao YIN, et al. *Parallel matrix factorization for low-rank tensor completion*. Inverse Probl. Imaging, 2015, **9**(2): 601–624.

[16] A. RAJWADE, A. RANGARAJAN, A. BANERJEE. *Image denoising using the higher order singular value decomposition*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013, **35**(4): 849–862.

[17] Yi PENG, Deyu MENG, Zongben XU, et al. *Decomposable nonlocal Tensor dictionary learning for multispectral image denoising*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, 2949–2956.

[18] Qi XIE, Qian ZHAO, Deyu MENG, et al. *Multispectral images denoising by intrinsic Tensor sparsity regularization*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, 1692–1700.

[19] Ruru HAO, Zhixun SU. *A Patch-based Low-rank Tensor Approximation Model for Multiframe Image Denoising*. Journal of Computational and Applied Mathematics, 2017, DOI:10.1016/j.cam.2017.01.022.

[20] Canyi LU, Jiashi FENG, Yudong CHEN, et al. *Tensor Robust principal component analysis: exact recovery of corrupted low-rank Tensors via convex optimization*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, 5249–5257.

[21] Wenfei CAO, Yao WANG, Jian SUN, et al. *Total variation regularized tensor RPCA for background subtraction from compressive measurements*. IEEE Trans. Image Process, 2016, **25**(9): 4075–4090.

[22] Zaiwen WEN, Wotao YIN, Yin ZHANG. *Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm*. Math. Program. Comput., 2012, **4**(4): 333–361.

[23] Yuan SHEN, Zaiwen WEN, Yin ZHANG. *Augmented Lagrangian alternating direction method for matrix separation based on low-rank factorization*. Optim. Methods Softw., 2014, **29**(2): 239–263.