2018 IEEE Confs on Internet of Things, Green Computing and Communications, Cyber, Physical and Social Computing, Smart Data, Blockchain, Computer and Information Technology, Congress on Cybermatics

An Implementable Accelerated Alternating Direction Method of Multipliers for Low-Rank Tensor Completion

Duo Qiu*, Hu Zhu† and Xiongjun Zhang‡
*School of Economics and Trade
Hunan University, Changsha, China
Email: qiuduoduo13@hnu.edu.cn
†College of Telecommunication and Information Engineering
Nanjing University of Posts and Telecommunications, Nanjing, China
Email: peter.hu.zhu@gmail.com
‡School of Mathematics and Statistics
Central China Normal University, Wuhan, China
Email: xjzhang@mail.ccnu.edu.cn

***Abstract*—Low-rank tensor completion has attracted much interest in many applications such as image processing, data mining and machine learning. A widely used method is to minimize the sum of nuclear norms of unfolding matrices of a tensor. In this paper, we study a accelerated alternating direction method of multipliers (ADMM) for solving the sum of nuclear norms of unfolding matrices of a tensor. The basic idea of accelerated ADMM is to incorporate a multistep acceleration scheme into the classical ADMM. We design efficient implementations of the algorithm and present its convergence result. Extensive numerical examples on both random and real world data are presented to validate the superiority of our proposed algorithm over class ADMM.**

***Keywords*-Accelerated alternating direction method of multipliers, low-rank tensor completion, nuclear norm**

I. Introduction

Recovering an unknown low-rank tensor from partial observations plays an important role in many fields such as image processing, data mining, cloud computing [1], [2], [3], [4], multienergy computed tomography and machine learning [5], to name only a few. For low-rank tensor completion, a general model for low-rank tensor completion is to minimize the following rank minimization problem:

$$\begin{aligned} &\min \ \text{rank}(\mathcal{X}), \\ &\text{s.t. } \mathcal{P}_\Omega(\mathcal{X}) = \mathcal{P}_\Omega(\overline{\mathcal{X}}), \end{aligned} \tag{1}$$

where $\mathcal{X}, \overline{\mathcal{X}} \in \mathbb{R}^{n_1 \times \cdots \times n_N}$, $\text{rank}(\mathcal{X})$ denotes the rank of $\mathcal{X}$, $\mathcal{P}_\Omega(\overline{\mathcal{X}})$ denotes that the entries of $\overline{\mathcal{X}}$ are given in Ω while the remaining parts are missing, and Ω is an index set.

Tensors are high-order extensions of vectors and matrices, which are called as the first-order and second-order tensors, respectively. In particular, (1) reduces to the matrix completion problem when the order of $\mathcal{X}$ satisfies $N = 2$ [6]. The rank minimization of matrix completion is NP-hard in general [7]. A widely used convex relaxation is the nuclear norm, which is the convex hull of rank minimization under the unit ball [8]. Many efficient algorithms are also proposed to solve the nuclear norm minimization problems recently, see, e.g., [9] and reference therein.

In contrast, the rank of a tensor is much more difficult than that of a tensor. Two popular kinds of ranks of tensors are CP rank [10] and Tucker rank [11], respectively. However, the computation of CP rank is generally NP-hard. Tucker rank can be computed by matrix singular value decomposition efficiently. Similar to matrix completion, the Tucker rank minimization is written as follows.

$$\begin{aligned} &\min \ \sum_{i=1}^{N} \alpha_i \text{rank}(\mathcal{X}_{(i)}), \\ &\text{s.t. } \mathcal{P}_\Omega(\mathcal{X}) = \mathcal{P}_\Omega(\overline{\mathcal{X}}), \end{aligned} \tag{2}$$

where where α_i are weight coefficients satisfying $\alpha_i \geq 0$ and $\sum_{i=1}^{N} \alpha_i = 1$ and $\mathcal{X}_{(i)}$ is the mode-i unfolding of $\mathcal{X}$. Then a convex relaxation of (2) is to minimize the nuclear norms of unfolding matrices [5]:

$$\begin{aligned} &\min \ \sum_{i=1}^{N} \alpha_i \|(\mathcal{X}_{(i)})\|_*, \\ &\text{s.t. } \mathcal{P}_\Omega(\mathcal{X}) = \mathcal{P}_\Omega(\overline{\mathcal{X}}). \end{aligned} \tag{3}$$

Many efficient algorithms are presented to solve (3). For example, Liu et al. [5] proposed penalty methods by variable splitting techniques. Gandy et al. [12] proposed an alternating direction method of multipliers (ADMM) for solving this problem. Moreover, Ouyang et al. [13] studied the accelerated ADMM for convex optimization problem. They also showed that the rate of convergence of accelerated ADMM is better than that of classical ADMM. In this paper, we study an implementable accelerated ADMM for solving (3). The basic idea of accelerated ADMM is to incorporate a multistep acceleration scheme into classical ADMM. Numerical examples on both random and real world data show the efficiency of the accelerated ADMM.

978-1-5386-7975-3/18/$31.00 ©2018 IEEE
DOI 10.1109/Cybermatics_2018.2018.00113

The remaining parts of this paper are organized as follows. Some notation and notions are introduced in section II. The accelerated ADMM is presented to solve the sum of nuclear norms of unfolding matrices of a tensor minimization problem in section III. Numerical experiments including random data and real-world data show the efficiency of the proposed algorithm in section IV. We give the conclusions in section V.

II. Preliminaries

In this section, we give a brief review about some notation used throughout this paper. More details about tensors can be found in [14]. We use lowercase letters for scalars, e.g., x, bold lowercase letters for vectors, e.g., $\mathbf{x}$, uppercase letters for matrices, e.g., X, and calligraphy letters for tensors e.g., $\mathcal{X}$. For a vector $\mathbf{x} \in \mathbb{R}^n$, the Euclidean norm of $\mathbf{x}$ is denoted by $\|\mathbf{x}\|$. The nuclear norm of a matrix X is denoted by $\|X\|_*$, i.e., the sum of all singular values.

The inner product of two tensors $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{n_1 \times \cdots \times n_N}$ with the same size is defined as the sum of the products of their entries, i.e.,

$$\langle \mathcal{X}, \mathcal{Y} \rangle := \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_N=1}^{n_N} x_{i_1 i_2 \cdots i_N} y_{i_1 i_2 \cdots i_N}.$$

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

III. Accelerated ADMM for low-rank tensor completion

In this section, the ADMM and accelerated ADMM is presented to solve (3). Now we consider a general convex relaxation method for tensor rank minimization problem, which is reformulated as the following convex problem for tensor recovery.

$$\begin{aligned} \min_{\mathcal{X}} \ & \sum_{i=1}^{N} \alpha_i \|\mathcal{X}_{(i)}\|_*, \\ \text{s.t. } & \mathcal{A}\mathcal{X} = b. \end{aligned} \tag{4}$$

A special case of above problem is tensor completion, then (4) can become to (3)

In order to solve (4), we consider the following unconstrained problem:

$$\min_{\mathcal{X}} \ \sum_{i-1}^{N} \alpha_i \|\mathcal{X}_{(i)}\|_* + \frac{\lambda}{2} \|\mathcal{A}\mathcal{X} - b\|^2, \tag{5}$$

where λ is the penalty. In order to split the two terms of objective function, we introduce the new tensor variables $\mathcal{Y}_1, \cdots, \mathcal{Y}_N$, which denote the N different unfoldings matrices $\mathcal{X}_{(1)}, \ldots, \mathcal{X}_{(N)}$ of the tensor $\mathcal{X}$ alone different directions, i.e., introducing tensor variables: $\mathcal{Y}_i \in \mathbb{R}^{n_1 \times \cdots \times n_N}$ such that $\mathcal{Y}_{i,(i)} = \mathcal{X}_{(i)},\ i = 1, \ldots, N$. Thus, we can obtain the following equivalence formulation of (5),

$$\begin{aligned} \min_{\mathcal{X}, \mathcal{Y}_i} \ & \sum_{i=1}^{N} \alpha_i \|\mathcal{Y}_{i,(i)}\|_* + \frac{\lambda}{2} \|\mathcal{A}\mathcal{X} - b\|^2, \\ \text{s.t. } & \mathcal{Y}_i = \mathcal{X}, \quad i = 1, \ldots, N. \end{aligned} \tag{6}$$

Let $A = (\mathcal{I}, \ldots, \mathcal{I})^T$, where the number of $\mathcal{I}$ is N, $\mathcal{Y} = (\mathcal{Y}_1, \ldots, \mathcal{Y}_N)^T$. Then let

$$f(\mathcal{X}) := \frac{\lambda}{2} \|\mathcal{A}\mathcal{X} - b\|^2, g(\mathcal{Y}) := \sum_{i=1}^{N} \alpha_i \|\mathcal{Y}_{i,(i)}\|_*. \tag{7}$$

Since $\mathcal{Y} = (\mathcal{I}, \cdots, \mathcal{I})^T \mathcal{X} = A\mathcal{X}$, the augmented Lagrangian function of (6) is defined by

$$\begin{aligned} \mathcal{L}(\mathcal{X}, \mathcal{Y}, \mathcal{Z}) &:= f(\mathcal{X}) + g(\mathcal{Y}) - \langle \mathcal{Z}, G(\mathcal{X}) - \mathcal{Y} \rangle \\ &\quad + \frac{\beta}{2} \|G(\mathcal{X}) - \mathcal{Y}\|_F^2 \\ &= \frac{\lambda}{2} \|\mathcal{A}(\mathcal{X}) - b\|^2 + \sum_{i=1}^{N} (\alpha_i \|\mathcal{Y}_{i,(i)}\|_* \\ &\quad - \langle \mathcal{Z}_i, \mathcal{X} - \mathcal{Y}_i \rangle + \frac{\beta}{2} \|\mathcal{X} - \mathcal{Y}_i\|_F^2), \end{aligned} \tag{8}$$

where $\mathcal{Z} = (\mathcal{Z}_1, \cdots, \mathcal{Z}_N)^T$ is the Lagrangian multiplier. Note that (8) is convex, unconstrained problem, each subproblem can be solved independently.

Computing $\mathcal{Y}_i$:

We consider $\mathcal{Y}_{i,(i)}$ firstly. According to (8), $\mathcal{Y}_{i,(i)}$ can be computed as follows.

$$\begin{aligned} &\mathcal{Y}_{i,(i)} \\ &= \arg\min_{\mathcal{Y}_{i,(i)}} \Big\{ \frac{\alpha_i}{\beta} \|\mathcal{Y}_{i,(i)}\|_* + \frac{1}{2} \|\mathcal{Y}_{i,(i)} - (\mathcal{X}_{(i)} - \frac{1}{\beta} \mathcal{Z}_{i,(i)})\|_F^2 \Big\} \\ &= S_{\frac{\alpha_i}{\beta}} (\mathcal{X}_{(i)} - \frac{1}{\beta} \mathcal{Z}_{i,(i)}), \end{aligned} \tag{9}$$

where $\mathcal{S}_\tau(Y) := U \text{diag}(\{\sigma_i - \tau\}_+) V^*$ with $Y = U \text{diag}\{\sigma_i\} V^T$ and $(t)_+ = \max\{t, 0\}$. Then

$$\mathcal{Y}_i = \text{Fold}_i (S_{\frac{\alpha_i}{\beta}} (\mathcal{X}_{(i)} - \frac{1}{\beta} \mathcal{Z}_{i,(i)})).$$

Hence, $\mathcal{Y} = (\mathcal{Y}_1, \ldots, \mathcal{Y}_N)^T$.

Computing $\mathcal{X}$:

The minimization with respect to $\mathcal{X}$ is to solve the following quadratic function:

$$\begin{aligned} \min_{\mathcal{X}} \quad \mathcal{L}(\mathcal{X}) = \frac{\lambda}{2} \|\mathcal{A}\mathcal{X} - b\|^2 - \sum_{i=1}^{N} \langle \mathcal{Z}_i, \mathcal{X} - \mathcal{Y}_i \rangle \\ + \frac{\beta}{2} \sum_{i=1}^{N} \|\mathcal{X} - \mathcal{Y}_i\|_F^2), \end{aligned} \tag{10}$$

which is continuously differentiable. Thus, the minimizer $\widetilde{\mathcal{X}}$ of (10) satisfies $\nabla\mathcal{L}(\widetilde{\mathcal{X}}) = 0$. Thus, we obtain

$$\mathcal{X} = (\lambda\mathcal{A}^*\mathcal{A} + N\beta\mathcal{I})^{-1}\Big(\sum_{i=1}^{N}\mathcal{Z}_i + \sum_{i=1}^{N}\beta\mathcal{Y}_i + \lambda\mathcal{A}^*b\Big). \quad (11)$$

In tensor completion, $\mathcal{A} = \Omega$, we can calculate the inverse easily:

$$[(\lambda\mathcal{A}^*\mathcal{A} + N\beta\mathcal{I})^{-1}(\mathcal{Z})]_\Omega = (\lambda + N\beta)^{-1}(\mathcal{Z})_\Omega,$$
$$[(\lambda\mathcal{A}^*\mathcal{A} + N\beta\mathcal{I})^{-1}(\mathcal{Z})]_{\bar{\Omega}} = (N\beta)^{-1}(\mathcal{Z})_{\bar{\Omega}}.$$

Thus,

$$\mathcal{X} = \begin{cases} (\lambda + N\beta)^{-1}\Big(\sum_{i=1}^{N}\mathcal{Z}_i + \sum_{i=1}^{N}\beta\mathcal{Y}_i + \lambda\mathcal{A}^*b\Big)_\Omega, \\ (N\beta)^{-1}\Big(\sum_{i=1}^{N}\mathcal{Z}_i + \sum_{i=1}^{N}\beta\mathcal{Y}_i + \lambda\mathcal{A}^*b\Big)_{\bar{\Omega}}, \end{cases} \quad (12)$$

where $\mathcal{A}^*b$ is a tensor, and $(\mathcal{A}^*b)_{\bar{\Omega}} = 0$, the vectorization of $(\mathcal{A}^*b)_\Omega$ is b.

Now we are ready to present the accelerated ADMM for solving for solving (4). The accelerated scheme used in [13] can improve the performance of ADMM. The accelerated ADMM can be described as follows.

Acelerated ADMM for tensor completion for solving (4).

Step 0. let $\beta, \tau, \lambda, \eta > 0$ be given parameters. For $k = 0, 1, \ldots,$ perform the following steps.

Step 1. $\mathcal{X}^{k+1} = (\lambda\mathcal{A}^*\mathcal{A} + N\beta\mathcal{I})^{-1}\Big(\sum_{i=1}^{N}\widetilde{\mathcal{Z}}_i + \sum_{i=1}^{N}\beta\widetilde{\mathcal{Y}}_i + \lambda\mathcal{A}^*b\Big)$.

Step 2. $\mathcal{Y}_i^{k+1} = \text{refold}(S_{\frac{\alpha_i}{\beta}}(\mathcal{X}_{(i)}^{k+1} - \frac{1}{\beta}\widetilde{\mathcal{Z}}_{i,(i)}^{k}))$.

Step 3. $\mathcal{Z}_i^{k+1} = \widetilde{\mathcal{Z}}_i^k - \beta(\mathcal{X}^{k+1} - \mathcal{Y}_i^{k+1})$.

Step 4. $c_k = \tau^{-1}\|\mathcal{Z}_i^{k+1} - \widetilde{\mathcal{Z}}_i^k\|_F^2 + \tau\|\mathcal{Y}_i^{k+1} - \widetilde{\mathcal{Y}}_i^k\|_F^2$.

Step 5. if $c_k < \eta c_{k-1}$, then $\alpha_{k+1} = \frac{1+\sqrt{1+4\alpha_k^2}}{2}$, and

$$\widetilde{\mathcal{Y}}_i^{k+1} = \mathcal{Y}_i^{k+1} + \frac{\alpha_k - 1}{\alpha_{k+1}}(\mathcal{Y}_i^{k+1} - \mathcal{Y}_i^k),$$

$$\widetilde{\mathcal{Z}}_i^{k+1} = \mathcal{Z}_i^{k+1} + \frac{\alpha_k - 1}{\alpha_{k+1}}(\mathcal{Z}_i^{k+1} - \mathcal{Z}_i^k).$$

else

$$\alpha_{k+1} = 1, \widetilde{\mathcal{Y}}_i^{k+1} = \mathcal{Y}_i^{k-1},$$
$$\widetilde{\mathcal{Z}}_i^{k+1} = \mathcal{Z}_i^{k-1},\ c_k = \eta^{-1}c_{k-1}.$$

end

Step 6. If a stopping criterion is not met, go to Step 1.

Similar to [13], we summarize the convergence of accelerated ADMM as follows.

Theorem 3.1: Algorithm accelerated ADMM converges in the sense that $\lim_{k\to\infty} c_k = 0$.

IV. Numerical experiments

In this section, numerical results are presented to validate the efficiency of accelerated ADMM (Acc ADMM). We mainly compare with ADMM to show the superiority of accelerated scheme. We test the experiments on both random and real-world data with the missing values. All the experiments were performed under Windows 8 operating system and MATLAB R2013a running on an ASUS laptop equipped with AMD Core 4 Quad CPU at 2.5 GHz and 8 GB of memory.

Table I
Performance of different algorithms for tensor completion.

Algorithm	RelEr	Time(s)	Ite.	RelEr	Time(s)	Ite.
	20×30×40, SR=0.3, r=2			50×50×50, SR=0.6, r=5		
Acc ADMM	9.84e-8	27.89	415	8.96e-8	64.88	234
ADMM	9.67e-8	37.00	601	9.29e-8	181.12	908
	20×20×20×20, SR=0.3, r=2			20×20×20×20, SR=0.6, r=2		
Acc ADMM	9.28e-8	120.14	346	8.80e-8	71.86	206
ADMM	8.77e-8	368.20	1630	9.12e-8	124.10	547
	20×30×40×50, SR=0.3, r=2			20×30×40×50, SR=0.6, r=2		
Acc ADMM	8.44e-8	1050.4	352	5.20e-8	816.60	275
ADMM	7.41e-5	3812.9	2000	9.63e-8	1727.4	875

A. Random data

For the random tensor, the testing tensor are the following form with Tucker decomposition:

$$\mathcal{X} = \mathcal{C} \times_1 U_1 \times_2 \cdots \times_N U_N,$$

where $\mathcal{C} \in \mathbb{R}^{r_1\times r_2\times\cdots\times r_N}$ is the core tensor generated by Matlab command **randn**$(r_1, \cdots, r_N)$ and $U_i \in \mathbb{R}^{n_i\times r_i}$ are the left singular vectors from the singular value decomposition (SVD) of $\mathcal{X}_{(i)}$. We consider the form of U_i later. Thus, the Tuncker rank of tensor $\mathcal{X}$ equals $(r_1, \cdots, r_N)$ almost surely, which is similar to [15].

In order to evaluate the performance of different algorithms, the relative error is used to measure the performance, which is defined as follows.

$$\text{RelEr} := \frac{\|\mathcal{X} - \mathcal{M}\|_F}{\|\mathcal{M}\|_F},$$

where $\mathcal{X}$ is the estimator tensor and $\mathcal{M}$ is the original tensor. The stopping criterion of the Acc ADMM is

$$\frac{\|\mathcal{X}^{k+1} - \mathcal{X}^k\|_F}{\|\mathcal{X}^k\|_F} \le 1 \times 10^{-6}.$$

For speeding up the convergence of the proposed algorithm, we compute a partial SVD of a matrix by using PROPACK package [16].

First, we consider the case that $U_1, U_2, \cdots, U_N$ are generated by Matlab command **randn**(n_i, r_i) with standard Gaussian distributed entries, i.e. $U_i \sim N(0, 1), i = 1, 2, \cdots, N$. The recovered results obtained by ADMM and Acc ADMM are shown for different tensors, sampling ratios and Tucker ranks in Table I. From Table I, we can see that Acc ADMM converges faster than ADMM for the same accurate solution. Acc ADMM needs less CPU time and fewer number of iterations than ADMM.

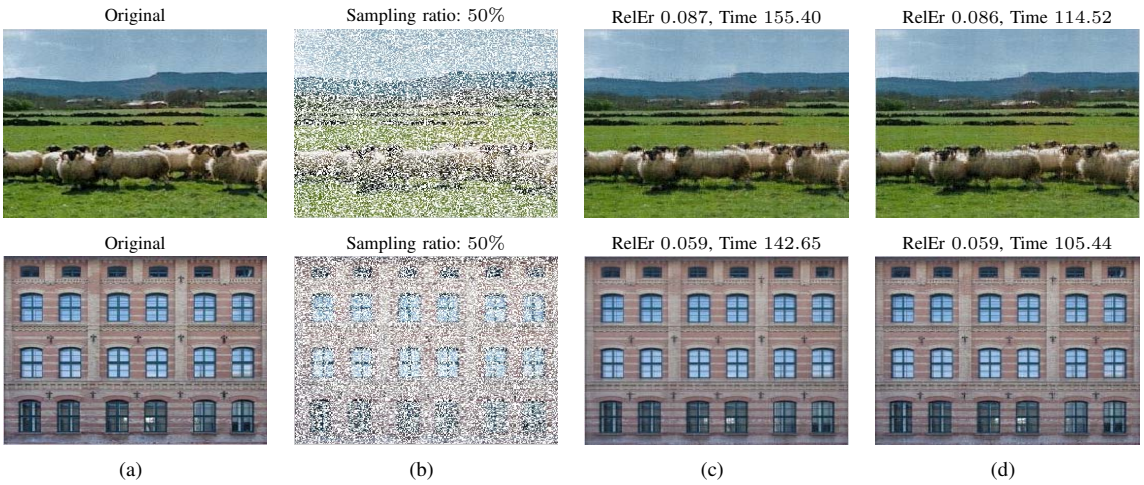

Figure 2. (a) Original images. (b) Observed images. (c) Recovery images by ADMM. (d) Recovery images by Acc ADMM.

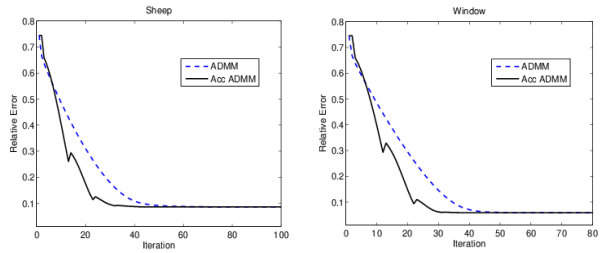

Figure 1. Results of relative error versus number of iterations for ADMM and Acc ADMM. (a) Sheep. (b) Window.

B. Real-world data

In this subsection, the missing pixels are generated uniformly at random. Figure 1 shows the relative error versus number of iterations of ADMM and Acc ADMM for Sheep, Window, Pepper and Boat, where the sampling ratio of observed images is 50%. We can observe that the relative errors of Acc ADMM decays faster than those of ADMM and Acc ADMM needs fewer number of iterations than those of ADMM.

Furthermore, we show the visual quality of different algorithms for Sheep, Window, pepper and Boat images in Figure 2, where the sampling ratio is 50%. From Figure 2, we can observe that the visual quality of ADMM and Acc ADMM is almost the same, but Acc ADMM needs less CPU time and fewer number of iterations than ADMM.

V. Conclusions

In this paper, we study an accelerated ADMM for low Tucker rank tensor completion. A convex relaxation method for tensor rank minimization is to minimize the sum of nuclear norms of unfolding matrices of a tensor. An accelerated ADMM is presented to solve the relaxation model. This algorithm can improve the performance of ADMM further, especially for large scale tensors. Extensive numerical examples including random and real world data are presented to demonstrate the superiority of the proposed algorithm.

Acknowledgments

The research of the third author was supported in part by the Fundamental Research Funds for the Central Universities under Grant CCNU17XJ031.

References

[1] X. Wang, L. T. Yang, X. Xie, J. Jin, and M. J. Deen, "A cloud-edge computing framework for cyber-physical-social services," *IEEE Commun. Magaz.*, vol. 55, no. 11, pp. 80–85, 2017.

[2] X. Wang, L. T. Yang, J. Feng, X. Chen, and M. J. Deen, "A tensor-based big service framework for enhanced living environments," *IEEE Cloud Comput.*, vol. 3, no. 6, pp. 36–43, 2016.

[3] L. T. Yang, X. Wang, X. Chen, J. Han, and J. Feng, "A tensor computation and optimization model for cyber-physical-social big data," *IEEE Trans. Sustainable Comput.*, 2017.

[4] L. T. Yang, X. Wang, X. Chen, L. Wang, R. Ranjan, X. Chen, and M. J. Deen, "A multi-order distributed hosvd with its incremental computing for big services in cyber-physical-social systems," *IEEE Trans. Big Data*, 2018.

[5] J. Liu, P. Musialski, P. Wonka, and J. Ye, "Tensor completion for estimating missing values in visual data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 208–220, Jan. 2013.

[6] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Found. Comput. Math.*, vol. 9, p. 717, Apr 2009.

[7] B. K. Natarajan, "Sparse approximate solutions to linear systems," *SIAM J. Comput.*, vol. 24, no. 2, pp. 227–234, 1995.

[8] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM Rev.*, vol. 52, no. 3, pp. 471–501, 2010.

[9] J. F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM J. Optim.*, vol. 20, no. 4, pp. 1956–1982, 2010.

[10] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *Studies Appl. Math.*, vol. 6, no. 1-4, pp. 164–189, 1927.

[11] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

[12] S. Gandy, B. Recht, and I. Yamada, "Tensor completion and low-n-rank tensor recovery via convex optimization," *Inverse Probl.*, vol. 27, no. 2, p. 025010, 2011.

[13] Y. Ouyang, Y. Chen, G. Lan, and E. P. Jr, "An accelerated linearized alternating direction method of multipliers," *SIAM J. Imaging Sci.*, vol. 8, no. 1, pp. 644–681, 2015.

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

[15] M. Bai, X. Zhang, G. Ni, and C. Cui, "An adaptive correction approach for tensor completion," *SIAM J. Imaging Sci.*, vol. 9, no. 3, pp. 1298–1323, 2016.

[16] R. Larsen, "PROPACK: A software package for the symmetric eigenvalue problem and singular value problems on lanczos and lanczos bidiagonalization with partial reorthogonalization, SCCM," *Available at http://soi. stanford. edu/rmunk/PROPACK*, 2004.