Scientific Research Publishing

Applied Mathematics, 2017, 8, 77-86
http://www.scirp.org/journal/am
ISSN Online: 2152-7393
ISSN Print: 2152-7385

Fast Tensor Principal Component Analysis via Proximal Alternating Direction Method with Vectorized Technique

Haiyan Fan[1], Gangyao Kuang[1], Linbo Qiao[2]

[1]Department of Electronic Science and Engineering, National University of Defense Technology, Changsha, China
[2]Department of Computer, National University of Defense Technology, Changsha, China
Email: hy_fan@yeah.net, linboqiao@foxmail.com

How to cite this paper: Fan, H.Y., Kuang, G.Y. and Qiao, L.B. (2017) Fast Tensor Principal Component Analysis via Proximal Alternating Direction Method with Vectorized Technique. *Applied Mathematics*, **8**, 77-86.
http://dx.doi.org/10.4236/am.2017.81007

Received: November 12, 2016
Accepted: January 21, 2017
Published: January 24, 2017

Abstract

This paper studies the problem of tensor principal component analysis (PCA). Usually the tensor PCA is viewed as a low-rank matrix completion problem via matrix factorization technique, and nuclear norm is used as a convex approximation of the rank operator under mild condition. However, most nuclear norm minimization approaches are based on SVD operations. Given a matrix $\boldsymbol{X} \in \mathbb{R}^{m \times n}$, the time complexity of SVD operation is $O\left(mn^2\right)$, which brings prohibitive computational complexity in large-scale problems. In this paper, an efficient and scalable algorithm for tensor principal component analysis is proposed which is called Linearized Alternating Direction Method with Vectorized technique for Tensor Principal Component Analysis (LADMVTPCA). Different from traditional matrix factorization methods, LADMVTPCA utilizes the vectorized technique to formulate the tensor as an outer product of vectors, which greatly improves the computational efficacy compared to matrix factorization method. In the experiment part, synthetic tensor data with different orders are used to empirically evaluate the proposed algorithm LADMVTPCA. Results have shown that LADMVTPCA outperforms matrix factorization based method.

Keywords

Tensor Principal Component Analysis, Proximal Alternating Direction Method, Vectorized Technique

1. Introduction

A tensor is a multidimensional array. For example, a first-order tensor is a vector, a second-order tensor is a matrix, and tensors with three or higher-order are

called higher-order tensors. Principal component analysis (PCA) finds a few linear combinations of the original variables. The PCA plays an important role in dimension reduction and data analysis related research areas [1]. Although the PCA and eigenvalue problem for the matrix has been well studied in the literature, little work has been done on the study of tensor PCA analysis.

The tensor PCA is of great importance in practice and has many applications, such as computer vision [2], social network analysis [3], diffusion Magnetic Resonance Imaging (MRI) [4] and so on. Similar to its matrix form, the problem of finding the PCs is related to the most variance of a tensor $\mathcal{F}$, which can be specifically formulated as [5]:

$$\begin{aligned} &\min_{x^1,x^2,\cdots,x^m} \left\| \mathcal{F} - \lambda x^1 \otimes x^2 \otimes \cdots \otimes x^m \right\|_F^2, \\ &st \lambda \in \mathbb{R}, \ \left\| x^i \right\| = 1, i = 1, 2, \cdots, m, \end{aligned} \tag{1}$$

which is equivalent to

$$\begin{aligned} &\min_{x^1,x^2,\cdots,x^m} -\mathcal{F} \cdot \left(x^1 \otimes x^2 \otimes \cdots \otimes x^m \right), \\ &st \left\| x^i \right\| = 1, i = 1, 2, \cdots, m, \end{aligned} \tag{2}$$

where $\cdot$ denotes the inner product between two tensors, and the inner product of two tensors $\mathcal{F}_1, \mathcal{F}_2 \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_m}$ is denoted as

$$\mathcal{F}_1 \cdot \mathcal{F}_2 = \sum_{i_1, i_2, \cdots, i_m} \left(\mathcal{F}_1 \right)_{i_1 i_2 \cdots i_m} \left(\mathcal{F}_2 \right)_{i_1 i_2 \cdots i_m}. \tag{3}$$

And $\otimes$ denotes the outer product between vectors, *i.e.* [6] [7]:

$$\left(x^1 \otimes x^2 \otimes \cdots \otimes x^m \right)_{i_1 i_2 \cdots i_m} = \prod_{k=1}^{m} \left(x^k \right)_{i_k}. \tag{4}$$

The above solution is called the leading PC. Once the leading PC is found, the other PCs can be computed sequentially via the so-called deflation technique. For example, the second PC could be gotten in the following ways: 1) Generate the first leading PC of the tensor, 2) Subtract the first leading PC of the tensor from the original tensor, 3) Generate the leading PC of the rest tensor. This leading PC is noted as the second PC of the original Tensor. And the rest PCs could be obtained in a similar way [8] [9]. The deflation procedure is presented in Algorithm 1. Although theoretical analysis of deflation procedure for matrix is well established (see [10] and the references therein for more details), the tensor counterpart has not been completely studied. However, the deflation process does provide an efficient heuristic way to compute multiple principal components of a tensor. The time consumption of different scaled between full SVD and leading PC is shown in **Table 1**. When the size of matrix increases, the computational cost for leading PC is far less than that of full SVD. Therefore,

Table 1. The comparison of running time (seconds) between full SVD and leading PC.

Scale	100 × 100	1000 × 1000	1000 × 5000	5000 × 5000	5000 × 10,000	10,000 × 10,000
Full SVD	1.70e−3	0.13	1.26	22.38	55.64	263.03
Leading PC	4.13e−4	0.01	0.09	0.52	1.13	2.27

although more iterations are needed for greedy atom decomposition methods to reach convergence, their total computational costs are much less compared with SVD based matrix completion methods. Thus, in the rest of this paper, we focus on finding the leading PC of a tensor.

If $\mathcal{F}$ is a super-symmetrical tensor, problem (2) can be reduced to [9]

$$\begin{aligned} &\min_{x} -\mathcal{F}\cdot\left(x\otimes x\otimes\cdots\otimes x\right), \\ &st\left\|x\right\|=1. \end{aligned} \tag{5}$$

In fact, the algorithm for supersymmetric tensor PCA problem can be extended to tensors that are not super-symmetric [5]. Therefore, this paper focuses on the PCA analysis of super-symmetric tensors. Problem (5) is NP-hard and is known as the maximum Z-eigenvalue problem. Note that a variety of eigenvalues and eigenvectors of a real symmetric tensor were introduced by Lim [11] and Qi [9] independently in 2005. Since then, various methods have been proposed to find the Z-eigenvalues, which however may correspond only to local optimums.

Another research line, like CANDECOMP (canonical decomposition) and PARAFAC (parallel factors) propose imposing rank-one constraint of tensor to realize the tensor decomposition:

$$\begin{aligned} &\min -\mathcal{F}\cdot\mathcal{X}, \\ &st\sum\nolimits_{k\in\mathbb{K}(n,m)}\frac{m!}{\prod_{j=1}^{n}k_j!}\mathcal{X}_{1^{k_1}2^{k_2}\cdots n^{k_n}}=1, \\ &\operatorname{rank}\left(\mathcal{X}\right)=1,\ \mathcal{X}\in\mathcal{S}^{n^m} \end{aligned} \tag{6}$$

where $\mathcal{X}=x\otimes x\otimes\cdots\otimes x$ and $\mathbb{K}\left(n,m\right)=\left\{k=\left(k_1,\cdots,k_n\right)\in\mathbb{Z}_+^m\middle|\sum_{j=1}^{n}k_j=m\right\}$. $\mathcal{X}\right)=1,\ \mathcal{X}\in\mathcal{S}^{n^m}$ indicates that $\mathcal{X}$ should be super-symmetrical. The first equality constraint is due to the fact that $\sum_{k\in\mathbb{K}(n,m)}\frac{m!}{\prod_{j=1}^{n}k_j!}\mathcal{X}_{1^{k_1}2^{k_2}\cdots n^{k_n}}=\left\|x\right\|^d=1$.

The difficulty of problem (6) lies in the dealing of the rank constraint $\operatorname{rank}\left(\mathcal{X}\right)=1$. Not only the rank function itself is difficult to deal with, but also determining the rank of a specific given tensor is already a difficult task, which is NP-hard in general. One way to deal with the difficulty is to convert the tensor optimization problem into a matrix optimization problem. [5] proved that if the tensor is rank-one, then the embedded matrix must be rank-one too, and vice versa. The tensor PCA problem can thus be solved by means of matrix optimization under a rank-one constraint. For low-rank matrix optimization problem, a nuclear norm penalty is often adopted to enforce a low-rank solution. However, most nuclear norm minimization approaches are based on SVD operations. Given a matrix $\boldsymbol{X}\in\mathbb{R}^{m\times n}$, the time complexity of SVD operation is $O\left(mn^2\right)$, which will bring prohibitive computational complexity in large problems (refer to **Table 1**).

To avoid the matrix SVD operation, we reformulate the problem (5) with vectorized technique, and consider the following optimization problem:

$$\begin{aligned} &\min_{x^1,x^2,\cdots,x^m} -\mathrm{Vec}(\mathcal{F})\cdot\mathrm{Vec}\left(x^1\otimes x^2\otimes\cdots\otimes x^m\right),\\ &st\left\|x^i\right\|=1, i=1,2,\cdots,m,\\ &x^1=x^2=\cdots=x^m \end{aligned} \tag{7}$$

where $\mathrm{Vec}(\mathcal{F})$ is the vectorized form of tensor $\mathcal{F}$. Related stream of algorithms to solve problem (7) are the ADM-type algorithms [12] [13]. Such a kind of algorithms has recently been shown effective to handle some non-convex optimization problems [14] [15]. However, the results of [14] require a lot well-justified assumption. Besides, the subproblems in ADM are easily solvable only when the linear mappings in the constraints are identities. To address this issue, [16] proposed a linearized ADM (LADM) method by linearizing the quadratic penalty term and adding a proximal term when solving the subproblems. In this paper, we adopt LADM algorithm for solving the optimization problem (7).

The rest of this paper is organized as follows. In Section 2, a brief review of LADM algorithm is firstly given. And then, the detailed description of using LADM with vectorized technique to solve tensor principal component problem is presented. Section 3 is the experiment part, in which synthetic tensor data with different orders are used to empirically evaluate the proposed algorithm LADMVTPCA. The last section gives concluding remarks.

Algorithm 1 Deflation Procedure

Input: $\Sigma^0=\varnothing$, $V^0=\varnothing$, $\mathcal{F}^0=\mathcal{F}$
Output: Σ, V
Initialization: Setup parameters
while (True) **do**
  Update λ^{k+1} and x^{k+1} by solving (7)
  Update $\Sigma^{k+1}=\Sigma^k\cup x^{k+1}$,
  Update $V^{k+1}=V^k\cup\lambda^{k+1}$,
  Update $\mathcal{F}^{k+1}=\mathcal{F}^k-\lambda^{k+1}x^{k+1}\otimes x^{k+1}$,
  $k=k+1$,
  if (stopping criteria satisfied) **then**
    Break.
  end if
end while

2. Linearized Alternating Direction Method (LADM)

In this section, we first review the Linearized Alternating Direction Method of Multipliers (LADM) [16], and then we present the linearized ADM for solving tensor principal component problem.

2.1. Algorithm

Considering the convex optimization problem,

$$\begin{aligned} &\min_{x,y} l(x)+r(y),\\ &s.t.\ Ax-By=0. \end{aligned} \tag{8}$$

where $x\in\mathbb{R}^d$, $y\in\mathbb{R}^l$, $A\in\mathbb{R}^{p\times d}$, $B\in\mathbb{R}^{p\times l}$, and $b\in\mathbb{R}^p$. It is well known

that problem (8) can be solved by the standard ADMM with typical iteration written as

$$x^{k+1} := \arg\min_{x} \mathcal{L}_{\beta}\left(x, y^{k}, \lambda^{k}\right), \tag{9}$$

$$\lambda^{k+1} := \lambda^{k} - \beta\left(Ax^{k+1} - By^{k}\right), \tag{10}$$

$$y^{k+1} := \arg\min_{y} \mathcal{L}_{\beta}\left(x^{k+1}, y, \lambda^{k+1}\right), \tag{11}$$

where the augmented Lagrangian function $\mathcal{L}_{\beta}(x, y, \lambda)$ is defined as

$$\mathcal{L}_{\beta}(x, y, \lambda) = l(x) + r(y) - \langle \lambda, Ax - By \rangle + \frac{\beta}{2}\|Ax - By\|^{2}. \tag{12}$$

The penalty parameter $\beta > 0$ is a constant dual step-size. The inefficient of ADMM inspires a linearized ADMM algorithm [16] by linearizing $l(x)$ in the x-subproblem. Specifically, it considers a modified augmented Lagrangian function:

$$\begin{aligned}\bar{\mathcal{L}}_{\beta}(x, \hat{x}, y, \lambda) = {} & l(\hat{x}) + \langle \nabla l(\hat{x}), x - \hat{x} \rangle + r(y) \\ & - \langle \lambda, Ax - By \rangle + \frac{\beta}{2}\|Ax - By\|^{2}.\end{aligned} \tag{13}$$

Then the LADM algorithm solves problem (8) by generating a sequence $\left\{x^{k+1}, \lambda^{k+1}, z^{k+1}\right\}$ as follows:

$$x^{k+1} := \arg\min_{x} \bar{\mathcal{L}}_{\beta}\left(x, x^{k}, y^{k}, \lambda^{k}\right), \tag{14}$$

$$\lambda^{k+1} := \lambda^{k} - \beta\left(Ax^{k+1} - By^{k}\right), \tag{15}$$

$$y^{k+1} := \arg\min_{y} \bar{\mathcal{L}}_{\beta}\left(x^{k+1}, x^{k}, y, \lambda^{k+1}\right). \tag{16}$$

The framework of linearized ADMM is given in Algorithm 2.

Algorithm 2 LADM

Choose the parameter β such that Equation (9) is satisfied;

Initialize an iteration counter $k \leftarrow 0$ and a bounded starting point $\left(x^{0}, \lambda^{0}, y^{0}\right)$;

repeat

Update x^{k+1} according to Equation (14);

$\lambda^{k+1} \leftarrow \lambda^{k} - \beta\left(Ax^{k+1} - By^{k}\right)$;

Update y^{k+1} according to Equation (16);

if some stopping criterion is satisfied; **then**

Break;

else

$k \leftarrow k + 1$;

end if

until exceed the maximum number of outer loop.

2.2. Linearized ADM with Vectorized Technique

In the following of this section, we present the linearized ADM method to solve the leading principal component problem. Without loss of generality, we consider a 4-th order tensor in this section for the leading principal component problem with rank-one constraint, we formulated the original problem as prob-

lem (17).

$$\begin{aligned} &\min_{x,y,z} -\mathrm{Vec}(\mathcal{F})\cdot\mathrm{Vec}(x\otimes y\otimes z\otimes w),\\ &s.t.\ x=y, y=z, z=w,\\ &\|x\|_2=1,\ \|y\|_2=1,\ \|z\|_2=1,\ \|w\|_2^2=1. \end{aligned} \tag{17}$$

where x, y, z, w are variables updated from iteration to iteration, the constraints $x=y, y=z, z=w$ limite the variables are close to each other and finally overlap, and the constraints $\|x\|_2=1,\ \|y\|_2=1,\ \|z\|_2=1,\ \|w\|_2^2=1$ make the solution robust to the scaling. After adding the constraints into the loss function, the problem (17) is equivalently reformed as

$$\begin{aligned} \min_{x,y,z} &-\mathrm{Vec}(\mathcal{F})\cdot\mathrm{Vec}(x\otimes y\otimes z\otimes w)+\langle\lambda_1, x-y\rangle\\ &+\langle\lambda_2, y-z\rangle+\langle\lambda_3, z-w\rangle+\langle\lambda_4, w-x\rangle\\ &+\frac{\rho}{2}*\left(\|x-y\|_2^2+\|y-z\|_2^2+\|z-w\|_2^2+\|w-x\|_2^2\right),\\ s.t.\ &x, y, z, w\in\mathcal{S}, \end{aligned} \tag{18}$$

where $\mathcal{S}=\left\{x\middle|\|x\|_2=1\right\}$ is a unit ball and ρ is a parameter to balance the object loss and the smooth terms. It should be noted that tensors with different orders could be deduced in a similarly way. The augmented Lagrangian function $\mathcal{L}(x, y, z, w, \lambda_1, \lambda_2, \lambda_3, \lambda_4, \rho)$ is defined as

$$\begin{aligned} &\mathcal{L}(x, y, z, w, \rho)\\ &:=-\mathrm{Vec}(\mathcal{F})\cdot\mathrm{Vec}(x\otimes y\otimes z\otimes w)+\langle\lambda_1, x-y\rangle\\ &\quad+\langle\lambda_2, y-z\rangle+\langle\lambda_3, z-w\rangle+\langle\lambda_4, w-x\rangle\\ &\quad+\frac{\rho}{2}*\left(\|x-y\|_2^2+\|y-z\|_2^2+\|z-w\|_2^2+\|w-x\|_2^2\right). \end{aligned} \tag{19}$$

where $x, y, z, w\in\mathcal{S}$. In the k-th iteration, we denote the variables by x^k, y^k, z^k, w^k, and ρ^k. Given $x^k, y^k, z^k, w^k, \rho^k$, iterates as follows

Algorithm 3 Algorithm for solving program (18)

Input: x^0, y^0, z^0, w^0, ρ^0
Output: x, y, z, w
Initialization: Setup parameters
while (Stop == False) **do**
  Update x^{k+1} by solving (20a)
  Update y^{k+1} by solving (20b),
  Update z^{k+1} by solving (20c),
  Update w^{k+1} by solving (20d),
  Update $\lambda_1, \lambda_2, \lambda_3, \lambda_4$,
  Update ρ^{k+1} by (20f),
  $k=k+1$,
  if Stopping condition satisfied
  then
    Stop = *True*
  end if
end while

$$x^{k+1}:=\arg\min_{x\in\mathcal{S}}\mathcal{L}(x, y^k, z^k, w^k, \rho^k), \tag{20a}$$

$$y^{k+1} := \arg\min_{y\in\mathcal{S}} \mathcal{L}\left(x^{k+1}, y, z^k, w^k, \rho^k\right), \tag{20b}$$

$$z^{k+1} := \arg\min_{z\in\mathcal{S}} \mathcal{L}\left(x^{k+1}, y^{k+1}, z, w^k, \rho^k\right), \tag{20c}$$

$$w^{k+1} := \arg\min_{w\in\mathcal{S}} \mathcal{L}\left(x^{k+1}, y^{k+1}, z^{k+1}, w, \rho^k\right), \tag{20d}$$

$$\text{Update } \lambda_1, \lambda_2, \lambda_3, \lambda_4 \tag{20e}$$

$$\rho^{k+1} := \begin{cases} \alpha\rho^k & \text{Balancedcases} \\ \rho^k/\alpha & \text{Others.} \end{cases} \tag{20f}$$

In the following part, we show how to solve these subproblems in the algorithm through a linearized way with vectorized technique. After all these subproblems solved, we will give the framework of the algorithm that summarize our algorithm for solving (18) in Algorithm 3. In order to achieve the saddle fast and improve the quality of the solution, we adjust the parameter ρ adaptively to balance the decrease speed of these two parts.

In a traditional way, x^{k+1} is obtained by minimizing $\mathcal{L}$ with respect to variable x while y, z, w, ρ are fixed with the value y^k, y^k, w^k, ρ^k respectively, and the Lagrange function is put forward as follow:

$$\begin{aligned} x^{k+1} := \arg\min_{x\in\mathcal{S}} & -\mathrm{Vec}(\mathcal{F})\cdot\mathrm{Vec}\left(x\otimes y^k\otimes z^k\otimes w^k\right) + \left\langle \lambda_1, x - y^k\right\rangle \\ & + \left\langle \lambda_2, y^k - z^k\right\rangle + \left\langle \lambda_3, z^k - w^k\right\rangle + \left\langle \lambda_4, w^k - x\right\rangle \\ & + \frac{\rho}{2} * \left(\left\|x - y^k\right\|_2^2 + \left\|y^k - z^k\right\|_2^2 + \left\|z^k - w^k\right\|_2^2 + \left\|w^k - x\right\|_2^2\right). \end{aligned} \tag{21}$$

We slightly modify the above LADM algorithm by imposing a proximal term $\frac{\delta}{2}\left\|x - x^k\right\|^2$ on the subproblem of x and update x^{k+1} via

$$\begin{aligned} x^{k+1} := \arg\min_{x\in\mathcal{S}} & -\mathrm{Vec}(\mathcal{F})\cdot\mathrm{Vec}\left(x\otimes y^k\otimes z^k\otimes w^k\right) + \left\langle \lambda_1, x - y^k\right\rangle \\ & + \left\langle \lambda_2, y^k - z^k\right\rangle + \left\langle \lambda_3, z^k - w^k\right\rangle + \left\langle \lambda_4, w^k - x\right\rangle \\ & + \frac{\rho}{2} * \left(\left\|x - y^k\right\|_2^2 + \left\|y^k - z^k\right\|_2^2 + \left\|z^k - w^k\right\|_2^2 + \left\|w^k - x\right\|_2^2\right) \\ & + \frac{\delta}{2}\left\|x - x^k\right\|^2. \end{aligned} \tag{22}$$

The minimum value will be obtained while the derivative of $\mathcal{L}$ with respect to x setting to zero, and utilize this condition, we obtain an equation which implies the solution of the subproblem

$$\begin{aligned} 0 \in & \mathrm{Vec}(\mathcal{F})\cdot\mathrm{Vec}\left(y^k\otimes z^k\otimes w^k\right) + \left\langle \lambda_1 - \lambda_4, x\right\rangle \\ & + \delta\left(x - x^k\right) + \rho^k\left(2x - y^k - w^k\right). \end{aligned} \tag{23}$$

where $\mathrm{Vec}(\mathcal{F})$ is the vectorized form of tensor $\mathcal{F}$, after solving the above equation, we get the x^{k+1}. For parsimony purpose, we just present the solving of the first subproblem here.

3. Numerical Experiments

In this subsection, we report the numerical experiments and results on Algorithm

3 to solve the tensor leading PC problem (18). As the ADMPCA [5] is one of these fastest methods based on matrix factorization and outperforms the SVD based methods, we will mainly focus on the comparison of our approach with ADMPCA proposed in [5]. The MATLAB codes of ADMPCA were downloaded from Professor Shiqian Ma's [5] homepage.

We apply our approach to synthetic datasets. The data is generated with uniform distributed eigen vectors x_i and eigen value λ_i, and the tensor is generated through the summation $\mathcal{F} = \sum_{i=1}^{n} \lambda_i * v_i \otimes v_i \cdots \otimes v_i$, in which the rank of tensor is controlled by the number of eigen vectors n, the order of tensor is controlled by number of v_i appear in the outer product, and the dimension of tensor is controlled by the dimension of the vector v_i.

We compare LADMVTPCA with ADMPCA for solving problem (18). In **Table 2**, we report the results on randomly created tensor with $\text{order} = 4$ and $\text{dimension} = 4, 8, 16, 32$. "objDiff" is used to denote the relative difference of the solutions obtained by ADMPCA and LADMVTPCA, $\text{objDiff} = \frac{\left\|\mathcal{F}_{\text{LADMVTPCA}} - \mathcal{F}_{\text{ADMPCA}}\right\|_F}{\max\left\{1, \left\|\mathcal{F}_{\text{ADMPCA}}\right\|_F\right\}}$. "objVal" is used to denote the relative difference of the objective eigen vectors, $\text{objVal} = \frac{\left\|v_{\text{LADMVTPCA}} - v_{\text{ADMPCA}}\right\|_F}{\max\left\{1, \left\|v_{\text{ADMPCA}}\right\|_F\right\}}$. "Time"

Table 2. Results of different algorithms for solving randomly generated tensor's leading principal component.

		ADMPCA		LADMVTPCA	
Inst.#	objDiff.	objVal	Time	objVal	Time
Dimension n = 4					
1	2.92e−06	1.07e+02	6.62e-01	1.07e+02	2.00e−03
2	7.29e−04	1.00e+02	6.57e-01	1.00e+02	3.01e−03
3	3.73e−04	1.00e+02	7.02e-01	1.00e+02	1.01e−03
4	4.57e−05	1.00e+02	6.85e-01	1.00e+02	1.92e−03
Dimension n = 8					
1	2.64e−06	1.00e+02	4.55e+00	1.00e+02	3.01e−03
2	1.66e−08	1.00e+02	4.48e+00	1.00e+02	3.00e−03
3	1.58e−05	1.00e+02	4.16e+00	1.00e+02	3.00e−03
4	1.32e−07	1.00e+02	4.42e+00	1.00e+02	3.00e−03
Dimension n = 16					
1	2.89e−04	1.04e+02	4.45e+01	1.04e+02	1.10e−02
2	2.13e−07	1.00e+02	4.65e+01	1.00e+02	7.00e−03
3	1.29e−05	1.00e+02	4.61e+01	1.00e+02	1.30e−02
4	2.63e−07	1.00e+02	4.64e+01	1.00e+02	1.18e−02
Dimension n = 32					
1	3.40e−09	1.00e+02	2.48e+03	1.00e+02	6.77e−01
2	8.57e−09	1.00e+02	2.40e+03	1.00e+02	7.19e−01
3	4.69e−06	1.00e+02	2.38e+03	1.00e+02	6.26e−01
3	2.85e−08	1.00e+02	2.43e+03	1.00e+02	6.80e−01

denote the CPU times (in seconds) of ADMPCA and LADMVTPCA, respectively. From Table 2 we can see that, LADMVTPCA produced comparable solutions compared to ADMPCA; however, LADMVTPCA was much faster than ADMPCA, especially for large-scale problem, *i.e.* $n = 16, 32$. Note that when $n = 16$, LADMVTPCA was about 4000 times faster than ADMPCA.

4. Conclusion

Tensor PCA is an emerging area of research with many important applications in image processing, data analysis, statistical learning, and bio-informatics. In this paper, we propose a new efficient and scalable algorithm for tensor principal component analysis called LADMVTPCA. A vectorized technique is introduced in the processing procedure and linear alternating direction method is used to solve the optimization problem. LADMVTPCA provides an efficient way to compute the leading PC. We empirically evaluate the proposed algorithm on synthetic tensor data with different orders. Results have shown that LADMVTPCA has much better computational cost beyond matrix factorization based method. Especially for large-scale problems, matrix factorization based method is much more time-consuming than our method.

References

[1] Kolda, T.G. and Bader, B.W. (2009) Tensor Decompositions and Applications. *SIAM Review*, **51**, 455-500. https://doi.org/10.1137/07070111X

[2] Wang, H. and Ahuja, N. (2004) Compact Representation of Multidimensional Data Using Tensor Rank-One Decomposition. *Proceedings of the* 17*th International Conference on Pattern Recognition*, *ICPR* 2004, Cambridge, 26-26 August 2004, 104-107.

[3] Huang, F., Niranjan, U., Hakeem, M.U. and Anandkumar, A. (2013) Fast Detection of Overlapping Communities via Online Tensor Methods. arXiv:1309.0787

[4] Qi, L., Yu, G. and Wu, E.X. (2010) Higher Order Positive Semidefinite Diffusion Tensor Imaging. *SIAM Journal on Imaging Sciences*, **3**, 416-433. https://doi.org/10.1137/090755138

[5] Jiang, B., Ma, S. and Zhang, S. (2014) Tensor Principal Component Analysis via Convex Optimization. *Mathematical Programming*, 1-35.

[6] Hitchcock, F.L. (1927) The Expression of a Tensor or a Polyadic as a Sum of Products. *Journal of Mathematics and Physics*, **6**, 164-189.

[7] Kofidis, E. and Regalia, P.A. (2002) On the Best Rank-1 Approximation of Higher-Order Supersymmetric Tensors. *SIAM Journal on Matrix Analysis and Applications*, **23**, 863-884. https://doi.org/10.1137/S0895479801387413

[8] Kolda, T.G. and Mayo, J.R. (2011) Shifted Power Method for Computing Tensor Eigenpairs. *SIAM Journal on Matrix Analysis and Applications*, **32**, 1095-1124. https://doi.org/10.1137/100801482

[9] Qi, L. (2005) Eigenvalues of a Real Supersymmetric Tensor. *Journal of Symbolic Computation*, **40**, 1302-1324. https://doi.org/10.1016/j.jsc.2005.05.007

[10] Mackey, L.W. (2009) Deflation Methods for Sparse PCA. *Advances in Neural Information Processing Systems*, **21**, 1017-1024.

[11] Lim, L.H. (2005) Singular Values and Eigenvalues of Tensors: A Variational Ap-

proach. *IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, Puerto Vallarta, Mexico, 13-15 December 2005, 129-132.

[12] Zhong, L.W. and Kwok, J.T. (2013) Fast Stochastic Alternating Direction Method of Multipliers. *Proceedings of the* 30*th International Conference on Machine Learning*, Atlanta, Georgia, 2013.

[13] Zhao, P.l., Yang, J.W., Zhang, T. and Li, P. (2013) Adaptive Stochastic Alternating Direction Method of Multipliers. arXiv:1312.4564

[14] Magnússon, S., Weeraddana, P.C., Rabbat, M.G. and Fischione, C. (2014) On the Convergence of Alternating Direction Lagrangian Methods for Nonconvex Structured Optimization Problems. arXiv:1409.8033 [math.OC]

[15] Hong, M.Y., Luo, Z.-Q. and Razaviyayn, M. (2015) Convergence Analysis of Alternating Direction Method of Multipliers for a Family of Nonconvex Problems. 2015 *IEEE International Conference on Acoustics*, *Speech and Signal Processing* (*ICASSP*), Brisbane, 19-24 April 2015, 3836-3840. https://doi.org/10.1109/ICASSP.2015.7178689

[16] Yang, J. and Yuan, X. (2013) Linearized Augmented Lagrangian and Alternating Direction Methods for Nuclear Norm Minimization. *Mathematics of Computation*, **820**, 301-329.