A Lagrange–Newton algorithm for tensor sparse principal component analysis

Shuai Li, Ziyan Luo & Yang Chen

To cite this article: Shuai Li, Ziyan Luo & Yang Chen (2024) A Lagrange–Newton algorithm for tensor sparse principal component analysis, Optimization, 73:9, 2933-2951, DOI: 10.1080/02331934.2023.2231482

To link to this article: https://doi.org/10.1080/02331934.2023.2231482

Published online: 12 Jul 2023.

OPTIMIZATION
2024, VOL. 73, NO. 9, 2933–2951
https://doi.org/10.1080/02331934.2023.2231482

A Lagrange–Newton algorithm for tensor sparse principal component analysis

Shuai Li, Ziyan Luo and Yang Chen

School of Mathematics and Statistics, Beijing Jiaotong University, Beijing, People's Republic of China

ABSTRACT
This paper is concerned with the tensor sparse principal component analysis (TSPCA) by obtaining principal components which are linear combinations of a small subset of the original features for tensorial data. The core mathematical model can be formulated as a nonsmooth nonconvex optimization problem with a polynomial objective function, and with the sparsity constraint and the unit Euclidean spherical constraint. By employing the tools in tensor analysis, along with the variational properties for the involved ℓ_0-norm, the optimality condition of TSPCA is analysed in terms of stationary points. To well resolve the problem, we reformulate the stationary conditions into the Lagrange stationary equation system via the property of the projection operator onto the sparsity constraint set. With special emphasis on the Jacobian nonsingularity of the corresponding nonlinear system, we propose the Lagrange–Newton algorithm for pursuing the stationary point, which serves as a promising approximation of the optimal solution to TSPCA. The locally quadratic convergence rate is also established under mild conditions. Numerical experiments illustrate the effectiveness of our proposed TSPCA approach in terms of the solution accuracy as well as the computation time.

ARTICLE HISTORY
Received 6 March 2023
Accepted 6 May 2023

KEYWORDS
Tensor principal component analysis; sparse nonlinear programming; ℓ_0-norm; Lagrange–Newton algorithm; locally quadratic convergence

1. Introduction

The tensor Principal Component Analysis (PCA), as a higher-order generalization of the traditional PCA, is an important and prevalent approach for modern data analysis, with wide applications in computer vision, diffusion magnetic resonance imaging, signal processing, spectral hypergraph theory and higher-order statistics [1–5]. To solve principal components (PCs, i.e. linear combinations of original variables) of tensors, Qi et al. [6] have proposed a class of Z-eigenvalue methods for four tensor cases with different orders and dimensions, and showed the effectiveness and prospect of methods through numerical experiments. Under the assumption of rank-one tensor data, Jiang et al. [7] have

CONTACT Ziyan Luo zyluo@bjtu.edu.cn

equivalently reformulated the tensor PCA problem as the matrix optimization with a rank-one constraint, and proposed two solution methods for solving the new matricization model. Huang et al. [8] have studied the convergence and statistical inference aspects of the power iteration algorithm for solving the tensor PCA model.

However, one drawback of the existing work on tensor PCA is that the obtained solution of PCs is usually dense, that is, PCs are linear combinations of all original variables. This makes tensor PCA difficult to interpret when the data dimension increases in practical problems. To deal with this problem, [9–11] have imposed a sparsity constraint on PCs of traditional PCA frameworks, which yields the so-called Sparse Principal Component Analysis (SPCA). The existing research on SPCA mainly focuses on matrix data, while the discussion on tensor data is relatively lacking. Considering this, as well as the inherited characteristics of the correlation structure of tensor data, we establish the following tensor SPCA (TSPCA) problem:

$$\min_{x\in\mathbb{R}^n} -\langle \mathcal{A}, x\circ x\circ\cdots\circ x\rangle \quad \text{s.t. } x^{\mathrm{T}}x = 1,\ \|x\|_0 \le s, \tag{1}$$

where $\mathcal{A} = (a_{i_1\ldots i_m}) \in S_{m,n}$ is an m-th order n-dimensional real supersymmetric tensor, i.e. the entry $a_{i_1\ldots i_m}$ of $\mathcal{A}$ remains unchanged under any permutation $\pi(i_1,\ldots,i_m)$ of index $(i_1,\ldots,i_m)$, $x\circ x \in \mathbb{R}^{n\times n}$ is the outer product of vectors. This optimization problem is nonconvex and noncontinuous due to the involved ℓ_0-norm.

The TSPCA model (1) is actually a special case of sparse nonlinear programming (SNP) problems:

$$\min_{x\in\mathbb{R}^n} f(x) \quad \text{s.t. } h(x) = 0,\ x \in S, \tag{2}$$

where $f:\mathbb{R}^n \to \mathbb{R}$, $h:\mathbb{R}^n \to \mathbb{R}^m$ are twice continuously differentiable functions, $S := \{x \in \mathbb{R}^n : \|x\|_0 \le s\}$ is the sparsity constraint set. For SNP problems resolution, existing numerical algorithms can be divided into two mainstream categories. The first category is the 'relaxation' method, which approximates the nonconvex and noncontinuous ℓ_0-norm as a continuous and/or convex surrogate function. Great progress on such relaxation methods has been made in [12–14]. The second one is the 'greedy' algorithm, which deals with the ℓ_0-norm constraint directly. This type of methods mainly includes first-order algorithms [15–17] and second-order algorithms using Newton step interpolation [18–20]. Recently, Zhao et al.[21] have proposed a locally quadratic convergent Lagrange–Newton algorithm (LNA) under mild conditions and shown its efficiency and superiority numerically. This inspires us to apply LNA to solve the TSPCA problem (1).

In this paper, we first propose the TSPCA model, in which the original ℓ_0-norm constraint is directly used for pursuing sparsity PCs instead of the existing relaxation schemes. Furthermore, the optimality condition of the TSPCA model,

Table 1. A list of notation.

Notation	Description
$[n]$	$:= \{1, 2, \dots, n\}$.
$\lvert t\rvert$	the absolute value of a scalar t.
$T_{m,n}$	the linear space of all real m-th order n-dimensional tensors.
$S_{m,n}$	the set of m-th order n-dimensional real supersymmetric tensors.
$x_{(i)}$	the i-th largest element of a vector x.
$\lVert x\rVert_\infty$	$:= \max\{\lvert x_i\rvert, i \in [n]\}$, the ℓ_∞-norm of a vector x.
Γ^*	$:= \{i \in [n] : x_i^* \neq 0\}$, the support set of the vector x^*.
T	index set from $[n]$.
$\lvert T\rvert$	cardinality of the set T.
T^c	the complementary set of T.
$\mathbb{J}_s$	$:= \{J \subseteq [n] : \lvert J\rvert = s\}$.
$\mathbb{J}_s(x)$	$:= \{J \in \mathbb{J}_s : \{i \in [n] : x_i \neq 0\} \subseteq J\}$.
$\mathbb{N}(x)$	$:= \{d \in \mathbb{R}^n : d^\top x = 0\}$ the subspace that orthogonal to the vector x.
$\boldsymbol{I}$	the identity matrix.
x_T	the subvector of x containing elements indexed by T.
$\boldsymbol{A}_{T,J}$	the submatrix of $\boldsymbol{A}$ whose rows and columns are respectively indexed by T and J.
$\boldsymbol{A}_{T,:}$	the submatrix of $\boldsymbol{A}$ containing rows indexed by T.
$\mathrm{rank}(\boldsymbol{A})$	the rank of the matrix $\boldsymbol{A}$.
$\mathrm{Tr}(\boldsymbol{A})$	the trace of the matrix $\boldsymbol{A}$.
$\mathcal{A}_{T^m}$	the subtensor of $\mathcal{A}$ indexed by T at each order.
$\mathcal{A}_{T,J^{(m-1)}}$	the subtensor of $\mathcal{A}$ whose first order is indexed by T and the other $m-1$ orders are indexed by J.
$\lVert x\rVert$	the Euclidean norm of the vector x.
$\lVert \boldsymbol{A}\rVert$	the spectral norm of the matrix $\boldsymbol{A}$.
$\lVert \boldsymbol{A}\rVert_*$	the nuclear norm of the matrix $\boldsymbol{A}$.
$\lVert \mathcal{A}\rVert_F$	the Frobenius norm of the tensor $\mathcal{A}$. If $\mathcal{A} \in T_{m,n}$, then $\lVert \mathcal{A}\rVert_F := \left(\sum_{i_1,i_2,\dots,i_m=1}^{n} \lvert a_{i_1\cdots i_m}\rvert^2\right)^{\frac{1}{2}}$.

and the nonsingularity of the Jacobian matrix of the Lagrangian stationary equation system are discussed under a mild condition. This can provide theoretical guarantees for the design of Newton-type algorithm. Finally, we develop LNA for solving TSPCA and establish its locally quadratic convergence.

The remainder of this paper is organized as follows. Section 2 analyses the optimality condition of the proposed TSPCA problem based on a strong β-Lagrangian stationary point. In Section 3, the nonsingularity of the Jacobian matrix of the Lagrangian stationary equation system is discussed under a mild condition. In Section 4, we design LNA for solving TSPCA and establish its locally quadratic convergence. Numerical results on synthetic data sets and real data sets are given in Section 5. Conclusions are drawn in Section 6. For convenience, notation that will be used throughout the paper is listed in Table 1.

2. Optimality analysis

In this section, we aim to analyse the optimality condition of problem (1) by introducing a strong β-Lagrangian stationary point, and establish the equivalent Lagrange stationary equation system based on the structural characteristic of the sparse projection operator. This provides a theoretical basis for the design of Lagrange–Newton algorithm for problem (1).

Considering the TSPCA problem (1), we denote

$$f(x) = -\langle \mathcal{A}, x \circ x \circ \cdots \circ x \rangle = -\mathcal{A}x^m$$
$$= -\sum_{i_1=1}^{n} \cdots \sum_{i_m=1}^{n} a_{i_1 \dots i_m} x_{i_1} x_{i_2} \cdots x_{i_m},$$

and

$$h(x) = x^{\mathrm{T}}x - 1.$$

The Lagrangian function of problem (1) is

$$L(x, y) = f(x) - y \cdot h(x), \quad \forall\, x \in \mathbb{R}^n,\ y \in \mathbb{R}.$$

Then, we have

$$\begin{cases} \nabla f(x) = -m\mathcal{A}x^{m-1} = \left(-m \sum_{i_2=1}^{n} \cdots \sum_{i_m=1}^{n} a_{ii_2 \cdots i_m} x_{i_2} \cdots x_{i_m} \right) \in \mathbb{R}^n, \\ \nabla^2 f(x) = -m(m-1)\mathcal{A}x^{m-2} \\ \quad = \left(-m(m-1) \sum_{i_3=1}^{n} \cdots \sum_{i_m=1}^{n} a_{iji_3 \cdots i_m} x_{i_3} \cdots x_{i_m} \right) \in \mathbb{R}^{n \times n}, \\ \nabla h(x) = 2x,\ \nabla^2 h(x) = 2\boldsymbol{I}, \\ \nabla_x L(x, y) = -m\mathcal{A}x^{m-1} - 2yx, \\ \nabla^2_{xx} L(x, y) = -m(m-1)\mathcal{A}x^{m-2} - 2y\boldsymbol{I}, \\ \nabla^2 L(x, y) = \begin{bmatrix} -m(m-1)\mathcal{A}x^{m-2} - 2y\boldsymbol{I} & 2x, \\ 2x^{\mathrm{T}} & 0 \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}. \end{cases} \tag{3}$$

By invoking the strong β-Lagrangian stationary point that introduced in [21], we can also get the specific definition of the same type of stationary point for problem (1) as below.

Definition 2.1: Given $x^* \in \mathbb{R}^n$, $\beta > 0$, if there exists a Lagrangian multiplier $y^* \in \mathbb{R}$ such that

$$\begin{cases} x^* = \Pi_S\left((1 + 2y^*)x^* + \beta m\mathcal{A}(x^*)^{m-1}\right), \\ (x^*)^{\mathrm{T}}x^* = 1, \end{cases} \tag{4}$$

where $\Pi_S(\cdot)$ is the projection operator on the sparsity set S, then x^* is called a strong β-Lagrangian stationary point of problem (1).

Using the above strong β-Lagrangian stationary point, we establish the following optimality conditions.

Theorem 2.2: *Let x^* be a local minimizer of problem (1). Then there exists a unique $y^* \in \mathbb{R}$ such that x^* is a strong β-Lagrangian stationary point of problem (1) for any $\beta \in (0, \hat{\beta})$, where*

$$\hat{\beta} = \begin{cases} \dfrac{|x^*|_{(s)}}{m\left\| \mathcal{A}_{(\Gamma^*)^c,(\Gamma^*)^{(m-1)}}(x^*_{\Gamma^*})^{(m-1)} \right\|_\infty}, & \text{if } \|x^*\|_0 = s \quad \text{and } (\mathcal{A}(x^*)^{m-1})_{(\Gamma^*)^c} \neq 0, \\ +\infty, & \textit{otherwise.} \end{cases} \tag{5}$$

Proof: Since x^* is a local minimizer of problem (1), we know that

$$\text{rank}\,(\nabla_{\Gamma^*} h(x)) = \text{rank}\left(2x^*_{\Gamma^*}\right) = 1,$$

and

$$\begin{aligned} \nabla_x L(x^*, y^*)_{(\Gamma^*)^c} =& (-m\mathcal{A}(x^*)^{m-1})_{(\Gamma^*)^c} - 2y^* x^*_{(\Gamma^*)^c} \\ =& \left(-m \sum_{i_2=1}^{n} \cdots \sum_{i_m=1}^{n} a_{ii_2\cdots i_m} x^*_{i_2} \cdots x^*_{i_m} \right)_{i\in(\Gamma^*)^c} \\ =& \left(-m \sum_{i_2\in\Gamma^*} \cdots \sum_{i_m\in\Gamma^*} a_{ii_2\cdots i_m} x^*_{i_2} \cdots x^*_{i_m} \right)_{i\in(\Gamma^*)^c} \\ =& \mathcal{A}_{(\Gamma^*)^c(\Gamma^*)^{m-1}}(x^*_{\Gamma^*})^{m-1}. \end{aligned}$$

Thus, according to [21, Theorem 1], there exists a unique $y^* \in \mathbb{R}$ such that x^* satisfies formula (4) provided that $\beta \in (0, \hat{\beta})$ with $\hat{\beta}$ given in (5). ■

To deal with the non-differentiable projection operator $\Pi_S(\cdot)$, we reformulate the optimality condition (4) by the following collection of sparse projection index sets.

Definition 2.3 ([21, Definition 2]): Given $x \in S$, $y \in \mathbb{R}$ and $\beta > 0$, denote $u = x - \beta\nabla_x L(x, y)$. Then the collection of sparse projection index sets of u is denoted by

$$\mathbb{T}(x, y; \beta) = \{T \in \mathbb{J}_s : |u_i| \geq |u_j|,\ \forall\, i \in T,\ \forall j \in T^c\}. \tag{6}$$

Theorem 2.4: *Given $x^* \in S, y^* \in \mathbb{R}, \beta > 0$, x^* is a strong β-Lagrangian stationary point of (1) with the Lagrangian multiplier y^* if and only if for any $T \in \mathbb{T}(x^*, y^*; \beta)$,*

$$F(x^*, y^*; T) := \begin{bmatrix} (\nabla_x L(x^*, y^*))_T \\ x^*_{T^c} \\ -h(x^*) \end{bmatrix} = \begin{bmatrix} (-m\mathcal{A}(x^*)^{m-1} - 2y^* x^*)_T \\ x^*_{T^c} \\ 1 - (x^*)^{\mathrm{T}} x^* \end{bmatrix} = 0. \tag{7}$$

In addition, $\mathbb{T}(x^*, y^*; \beta) = \mathbb{J}_s(x^*)$.

Proof: For the TSPCA problem (1), it follows from [21, Theorem 3] that the optimality condition (7) is satisfied and $\mathbb{T}(x^*, y^*; \beta) = \mathbb{J}_s(x^*)$. ■

Let $\boldsymbol{B}_T = m\mathcal{A}_{T^m}(x_T^*)^{m-2}$. By invoking Theorem 2.4, we have the following properties of $\boldsymbol{B}_T$.

Theorem 2.5: *Let x^* be a strong β-Lagrangian stationary point of problem* (1) *with the Lagrangian multiplier $y^* \in \mathbb{R}$. Then for any $T \in \mathbb{J}_s(x^*)$, x_T^* is an eigenvector of $\boldsymbol{B}_T$ with the corresponding eigenvalue $-2y^*$.*

Denote $z^* = (x^*, y^*)$. To solve the smooth Lagrangian stationary Equation (7) for a given $T \in \mathbb{T}(x^*, y^*; \beta)$, we next focus on the Jacobian matrix $\nabla_{(x,y)}F(x, y; T)$ and analyse its nonsingularity in a neighbourhood of z^*, where

$$\nabla_{(x,y)}F(x, y; T) := \begin{bmatrix} (-m(m-1)\mathcal{A}x^{m-2} - 2y\boldsymbol{I})_{T,:} & -2x_T \\ \boldsymbol{I}_{T^c} & 0 \\ -2x_T^{\mathrm{T}} & 0 \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}. \tag{8}$$

Performing elementary operations, we can get the following reduced Jacobian matrix

$$G(x, y; T) := \begin{bmatrix} (-m(m-1)\mathcal{A}x^{m-2} - 2y\boldsymbol{I})_{T,T} & -2x_T \\ -2x_T^{\mathrm{T}} & 0 \end{bmatrix} \in \mathbb{R}^{(s+1)\times(s+1)}, \tag{9}$$

which has the same nonsingularity as $\nabla_{(x,y)}F(x, y; T)$. Thus, it suffices to show that the reduced Jacobian matrix $G(x, y; T)$ is nonsingular in a neighbourhood of z^*.

3. Jacobian nonsingularity

In this section, we focus on the nonsingularity of the Jacobian matrix of Lagrangian stationary Equation (7). The following two main theorems give some useful properties, as well as the nonsingularity of Jacobian matrix under mild conditions. These crucial results provide theoretical guarantees for establishing the convergence of Lagrange–Newton algorithm.

Theorem 3.1: *Let x^* be a strong β-Lagrangian stationary point of problem* (1) *with the Lagrangian multiplier $y^* \in \mathbb{R}$. We have the following properties.*

(i) $\forall\, T \in \mathbb{J}_s(x^*)$, $\operatorname{rank}(\nabla_T h(x^*)) = 1$.
(ii) *$\nabla^2 f$ and $\nabla^2 h$ are Lipschitz continuous near x^*.*

Proof: For (i), we know from (4) that $(x^*)^{\mathrm{T}}x^* = 1$, which yields

$$\operatorname{rank}\left(\nabla_T h(x^*)\right) = \operatorname{rank}\left(2x^*_T\right) = 1, \quad \forall\, T \in \mathbb{J}_s(x^*).$$

For any $x_1, x_2 \in N(x^*, \delta_0) := \{x \in \mathbb{R}^n : \|x - x^*\| < \delta_0\}$, one has $\|x_1\| \le 1 + \delta_0$, and $\|x_2\| \le 1 + \delta_0$. Then

$$\begin{aligned}
&\|\nabla^2 f(x_1) - \nabla^2 f(x_2)\| \\
&\quad = m(m-1)\|-(\mathcal{A}{x_1}^{m-2} - \mathcal{A}{x_2}^{m-2})\| \\
&\quad \le m(m-1)\|\mathcal{A}x_1 \cdot x_1 \cdots x_1 - \mathcal{A}x_1 \cdot x_1 \cdots x_2 + \cdots + \mathcal{A}x_1 \cdot x_2 \cdots x_2 \\
&\qquad - \mathcal{A}x_2 \cdot x_2 \cdots x_2\| \\
&\quad \le m(m-1)(\|\mathcal{A}x_1 \cdots (x_1 - x_2)\| + \|\mathcal{A}x_1 \cdots (x_1 - x_2)x_2\| + \cdots \\
&\qquad + \|\mathcal{A}(x_1 - x_2)x_2 \cdots x_2\|) \\
&\quad \le m(m-1)\left(\|\mathcal{A}\|_F\|x_1\|^{m-3} + \|\mathcal{A}\|_F\|x_1\|^{m-4}\|x_2\| + \cdots + \|\mathcal{A}\|_F\|x_2\|^{m-3}\right) \\
&\qquad \times \|x_1 - x_2\| \\
&\quad \le m(m-1)\left(\|\mathcal{A}\|_F(1+\delta_0)^{m-3} + \cdots + \|\mathcal{A}\|_F(1+\delta_0)^{m-3}\right)\|x_1 - x_2\| \\
&\quad = \left(m(m-1)(m-2)\left(\|\mathcal{A}\|_F(1+\delta_0)^{m-3}\right)\right)\|x_1 - x_2\|.
\end{aligned}$$

Therefore, $\nabla^2 f$ is Lipschitz continuous near x^*. The Lipschitz continuity of $\nabla^2 h$ is trivial since $\nabla^2 h(x) = 2I$ for all x. This completes the proof. ■

The Lipschitz continuity of $\nabla^2 f$ and $\nabla^2 h$ allows us to find positive constants δ_0^*, L_1 and L_2, such that for any $z := (x; y) \in N(z^*, \delta_0^*)$,

$$\begin{aligned}
\|\nabla_x L(x,y) - \nabla_x L(x^*, y^*)\| &\le L_1\|z - z^*\|, \\
\|\nabla^2 L(x,y) - \nabla^2 L(x^*, y^*)\| &\le L_2\|z - z^*\|.
\end{aligned} \tag{10}$$

Specifically, note that

$$\begin{aligned}
&\|\nabla_x L(x,y) - \nabla_x L(x^*, y^*)\| \\
&\quad = \|-(m\mathcal{A}x^{m-1} - m\mathcal{A}(x^*)^{m-1}) - (2yx - 2y^*x^*)\| \\
&\quad \le \|-(m\mathcal{A}x^{m-1} - m\mathcal{A}(x^*)^{m-1})\| + 2\|yx - yx^* + yx^* - y^*x^*\| \\
&\quad \le \left(\sum_{i=0}^{m-2} m\,\|\mathcal{A}\|_F\left(1+\delta_0^*\right)^i\right)\|x - x^*\| + 2(|y^*| + \delta_0^*)\|x - x^*\| + 2|y - y^*| \\
&\quad \le \left(\sum_{i=0}^{m-2} m\,\|\mathcal{A}\|_F\left(1+\delta_0^*\right)^i\right)\|x - x^*\| + (m\|\mathcal{A}(x^*)^{m-2}\| + 2\delta_0^*)\|x - x^*\| \\
&\qquad + 2|y - y^*|
\end{aligned}$$

$$\begin{aligned}
&\leq \left(\sum_{i=0}^{m-2}(m\,\|\mathcal{A}\|_F\left(1+\delta_0^*\right)^i) + m\|\mathcal{A}\|_F\|x^*\|^{m-2} + 2\delta_0^*\right)\|x-x^*\| \\
&\quad + 2|y-y^*| \\
&= \left(\sum_{i=0}^{m-2}(m\,\|\mathcal{A}\|_F\left(1+\delta_0^*\right)^i) + m\|\mathcal{A}\|_F + 2\delta_0^*\right)\|x-x^*\| \\
&\quad + 2|y-y^*| \\
&\leq \left(\sum_{i=0}^{m-2}(m\,\|\mathcal{A}\|_F\left(1+\delta_0^*\right)^i) + m\|\mathcal{A}\|_F + 2\delta_0^* + 2\right)\|z-z^*\|,
\end{aligned}$$

where the second inequality can be obtained by using the similar proof skills of Theorem 3.1, and the third inequality follows from

$$\begin{aligned}
|-2y^*| &\leq \left\|m\mathcal{A}_{T^m}\left(x_T^*\right)^{m-2}\right\| = \left\|m\mathcal{A}_{T,T,\cdot,\ldots,\cdot}\left(x^*\right)^{m-2}\right\| \\
&\leq \left\|m\mathcal{A}\left(x^*\right)^{m-2}\right\|, \quad \forall\, T \in \mathbb{J}_s(x^*),
\end{aligned}$$

owing to Theorem 2.5 and the fact $x_{T^c}^* = 0$ for any $T \in \mathbb{J}_s(x^*)$. Thus, we can take

$$L_1 = \sum_{i=0}^{m-2}(m\,\|\mathcal{A}\|_F\left(1+\delta_0^*\right)^i) + m\|\mathcal{A}\|_F + 2\delta_0^* + 2. \tag{11}$$

Next, by direct manipulations, one has

$$\begin{aligned}
&\|\nabla^2 L(x,y) - \nabla^2 L(x^*,y^*)\| \\
&\quad = \left\|\begin{bmatrix} -m(m-1)(\mathcal{A}x^{m-2} - \mathcal{A}(x^*)^{m-2}) - 2(y-y^*)\boldsymbol{I} & -2(x-x^*) \\ -2(x-x^*)^{\mathrm{T}} & 0 \end{bmatrix}\right\| \\
&\quad \leq \left\|\begin{bmatrix} -m(m-1)(\mathcal{A}x^{m-2} - \mathcal{A}(x^*)^{m-2}) & 0 \\ 0 & 0 \end{bmatrix}\right\| \\
&\qquad + \left\|\begin{bmatrix} -2(y-y^*)\boldsymbol{I} & -2(x-x^*) \\ -2(x-x^*)^{\mathrm{T}} & 0 \end{bmatrix}\right\| \\
&\quad \leq m(m-1)\|\mathcal{A}x^{m-2} - \mathcal{A}(x^*)^{m-2}\| + \left\|\begin{bmatrix} -2(y-y^*)\boldsymbol{I} & 0 \\ 0 & 0 \end{bmatrix}\right\|
\end{aligned}$$

$$
\begin{aligned}
&+ \left\| \begin{bmatrix} 0 & -2(x-x^*) \\ -2(x-x^*)^{\mathrm{T}} & 0 \end{bmatrix} \right\| \\
&= m(m-1)\|\mathcal{A}x^{m-2} - \mathcal{A}(x^*)^{m-2}\| + 2|y-y^*| + 2\|x-x^*\| \\
&\leq \left(\sum_{i=0}^{m-3} m\,(m-1)\,\|\mathcal{A}\|_F \left(1+\delta_0^*\right)^i \right) \|x-x^*\| + 2\sqrt{2}\|z-z^*\| \\
&\leq \left(\sum_{i=0}^{m-3} (m\,(m-1)\,\|\mathcal{A}\|_F \left(1+\delta_0^*\right)^i) + 2\sqrt{2} \right) \|z-z^*\|.
\end{aligned}
$$

Therefore, one can take

$$
L_2 = \sum_{i=0}^{m-3} (m\,(m-1)\,\|\mathcal{A}\|_F \left(1+\delta_0^*\right)^i) + 2\sqrt{2}. \tag{12}
$$

We next discuss the nonsingularity of the reduced Jacobian matrix (9) under the following assumption.

Assumption 3.1: Let x^* be a strong β-Lagrangian stationary point of problem (1) with the Lagrangian multiplier $y^* \in \mathbb{R}$. For any $T \in \mathbb{J}_s(x^*)$, $-2y^*$ is the largest eigenvalue of $\boldsymbol{B}_T = m\mathcal{A}_{T^m}(x_T^*)^{m-2}$, and $-2y^*$ is of multiplicity 1. Moreover, $\forall\ T \in \mathbb{J}_s(x^*)$, $-2y^* > (m-1)\lambda_T$, where λ_T is any other eigenvalue of $\boldsymbol{B}_T$ distinct from $-2y^*$.

For illustration, we give a simple example in which Assumption 3.1 holds.

Example 3.2: Let $\mathcal{A} = \sum_{i=1}^{R} \alpha_i (u_i)^m \in S_{m,n}$ with $\alpha_1 > \alpha_2 \geq \alpha_3 \geq \cdots \geq \alpha_R \geq 0$, and let $\boldsymbol{U} = [u_1, \ldots, u_R] \in \mathbb{R}^{n\times R}$ be column orthogonal with $\|u_1\|_0 \leq s$ (For example, all diagonal tensors with the largest diagonal entry nonzero and of multiplicity 1). It is easy to verify that $x^* = u_1$ is the unique optimal solution of problem (1), and

$$
m\mathcal{A}(x^*)^{m-2} = m\sum_{i=1}^{R} \alpha_i u_i^m (x^*)^{m-2} = m\alpha_1 u_1 {u_1}^{\mathrm{T}} =: \boldsymbol{B}.
$$

Note that

$$
\left(m\mathcal{A}(x^*)^{m-1}\right)_{(\Gamma^*)^c} = (m\alpha_1 u_1)_{(\Gamma^*)^c} = 0.
$$

Thus, by applying Theorem 2.2, x^* is a strong β-Lagrangian stationary point of problem (1) for any $\beta \in (0, +\infty)$. The corresponding multiplier $y^* = -\frac{m\alpha_1}{2}$, due to Theorem 2.5. For any $T \in \mathbb{J}_s(x^*)$, $\boldsymbol{B}_T$ is a rank-one matrix. Thus

$$
-2y^* > (m-1)\lambda_T = 0,
$$

for any other eigenvalue λ_T of $\boldsymbol{B}_T$. This indicates that Assumption 3.1 holds.

Theorem 3.3: *Let x^* be a strong β-Lagrangian stationary point of problem* (1) *with the Lagrangian multiplier $y^* \in \mathbb{R}$. We define that $\delta^* := \min\{\delta_0^*, \delta_1^*\}$ with*

$$\delta_1^* = \frac{\min_{i\in\Gamma^*}|x_i^*| - \beta \max_{i\in(\Gamma^*)^c}|\nabla_x L(x^*,y^*)_i|}{\sqrt{2}(1+\beta L_1)}.$$

If Assumption 3.1 *holds, then there exists constants $\tilde{\delta}^* \in (0,\delta^*]$ and $M^* \in (0,+\infty)$ such that for any $z := (x;y) \in N_S(z^*;\tilde{\delta}^*)$ with $N_S(z^*;\tilde{\delta}^*) := \{z \in \mathbb{R}^{n+1} : x \in S, \|z - z^*\| < \tilde{\delta}^*\}$, the reduced Jacobian matrix $G(x,y;T)$ is nonsingular and*

$$\|G^{-1}(x,y;T)\| \le M^*, \quad \forall\, T \in \mathbb{T}(x,y;\beta).$$

Proof: We first prove that Assumption 2 of [21] holds when Assumption 3.1 is satisfied. For any $T \in \mathbb{J}_s(x^*)$,

$$\begin{aligned}(\nabla_{xx}^2 L(x^*,y^*))_{T,T} &= (-m(m-1)\mathcal{A}(x^*)^{m-2} - 2y^*\boldsymbol{I}_n)_{T,T}\\ &= \Big(-m(m-1)\sum_{i_3\in T}\cdots\sum_{i_m\in T} a_{iji_3\cdots i_m}x_{i_3}\cdots x_{i_m}\Big)_{i\in T,j\in T} - 2y^*\boldsymbol{I}_s \\ &= -m(m-1)\mathcal{A}_{T^m}(x_T^*)^{m-2} - 2y^*\boldsymbol{I}_s.\end{aligned} \tag{13}$$

Since $-2y^*$ is the largest eigenvalue of $\boldsymbol{B}_T$, and $-2y^*$ is of multiplicity 1, we know that the eigenspace of $\boldsymbol{B}_T$ corresponding to $-2y^*$ is exactly Span(x_T^*). Let the eigenvectors of $\boldsymbol{B}_T$ corresponding to other eigenvalues λ_i be $\alpha_i, i = 2,\ldots,s$. Denote the orthogonal matrix $\boldsymbol{U} = [x_T^*, \alpha_2, \ldots, \alpha_s]$. Then we have

$$\begin{aligned}&-(m-1)\boldsymbol{B}_T - 2y^*\boldsymbol{I}_s\\ &= -(m-1)\boldsymbol{U}\begin{bmatrix}-2y^* & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_s\end{bmatrix}\boldsymbol{U}^{\mathrm{T}}\\ &\quad + \boldsymbol{U}\begin{bmatrix}-2y^* & & & \\ & -2y^* & & \\ & & \ddots & \\ & & & -2y^*\end{bmatrix}\boldsymbol{U}^{\mathrm{T}}\\ &= \boldsymbol{U}\begin{bmatrix}(2m-4)y^* & & & \\ & -2y^* - (m-1)\lambda_2 & & \\ & & \ddots & \\ & & & -2y^* - (m-1)\lambda_s\end{bmatrix}\boldsymbol{U}^{\mathrm{T}}.\end{aligned} \tag{14}$$

By virtue of the orthogonality of $\boldsymbol{U}$, we have that for any nonzero $d \in \mathbb{N}(x_T^*)$, there exist not all zero scalars $c_2, \dots, c_s \in \mathbb{R}$ such that $d = \sum_{i=2}^{s} c_i\alpha_i$. Thus,

$$\begin{aligned} &d^{\mathrm{T}}(-(m-1)\boldsymbol{B}_T - 2y^*\boldsymbol{I}_s)d \\ &\quad = (2m-4)y^*(d^{\mathrm{T}}x_T^*)^2 + (-2y^* - (m-1)\lambda_2)c_2^2\|\alpha_2\|^4 \\ &\qquad + \dots + (-2y^* - (m-1)\lambda_s)c_s^2\|\alpha_s\|^4 \\ &\quad = (-2y^* - (m-1)\lambda_2)c_2^2\|\alpha_2\|^4 + \dots + (-2y^* - (m-1)\lambda_s)c_s^2\|\alpha_s\|^4. \end{aligned} \tag{15}$$

It is known from Assumption 3.1 that $-2y^* > (m-1)\lambda_i$, $i = 2, \dots, s$. Together with the fact that c_i, $i = 2, \dots, s$ are not all 0 since $d \neq 0$, we have

$$\begin{aligned} d^{\mathrm{T}}(\nabla_{xx}^2 L(x^*, y^*))_{T,T}d &= -d^{\mathrm{T}}(m(m-1)\mathcal{A}_{T^m}(x_T^*)^{m-2} + 2y^*\boldsymbol{I}_s)d \\ &> 0, \quad \forall\, 0 \neq d \in \mathbb{N}(\nabla_T h(x^*)), \end{aligned}$$

which yields that Assumption 2 in [21] holds. By invoking Theorem 3.1 and [21, Theorem 5], we can find $\delta_T > 0$ and $M_T > 0$ for any $T \in \mathbb{T}(x, y; \beta)$, such that $\|G^{-1}(x, y; T)\| < M_T$. Set

$$\tilde{\delta}^* := \min\{\delta^*, \{\delta_T\}_{T \in \mathbb{J}_s(x^*)}\}, \; M^* := \max_{T \in \mathbb{J}_s(x^*)} \{M_T\}. \tag{16}$$

It follows readily that for any $z = (x, y) \in N_S(z^*; \tilde{\delta}^*)$, $G(x, y; T)$ is nonsingular and $\|G^{-1}(x, y; T)\| \le M^*$, for all $T \in \mathbb{T}(x, y; \beta)$. ■

4. Lagrange–Newton algorithm

In this section, we give the framework of Lagrange–Newton algorithm (LNA) for solving problem (1) and establish its locally quadratic convergence.

LNA is essentially a process of iteratively solving $F(x, y; T) = 0$ by using Newton method and updating the corresponding index set $T(T \in \mathbb{T}(x, y; \beta))$. Let $(x^k, y^k) \in S \times \mathbb{R}$ be the current iteration. Given $\beta > 0$, we work on the following Newton equation to get the next iterate (x^{k+1}, y^{k+1}):

$$\nabla_{(x,y)}F(x^k, y^k; T_k)(x^{k+1} - x^k; y^{k+1} - y^k) = -F(x^k, y^k; T_k). \tag{17}$$

After simple calculation, (17) can be reduced to

$$\begin{cases} x_{T_k^c}^{k+1} = 0, \\ G(x^k, y^k; T_k)\begin{bmatrix} x_{T_k}^{k+1} - x_{T_k}^k \\ y^{k+1} - y^k \end{bmatrix} = \begin{bmatrix} \left(m\mathcal{A}(x^k)^{m-1}\right)_{T_k} + 2y^k x_{T_k}^k \\ (x^k)^{\mathrm{T}}x^k - 1 \end{bmatrix}. \end{cases} \tag{18}$$

The above system admits a unique solution since $G(x^k, y^k; T_k)$ is nonsingular due to Theorem 3.3, and hence the new iteration is well-defined.

To measure the accuracy of the numerical solution, we adopt the following metric from [21],

$$\eta_\beta(x^k, y^k; T_k) = \|F(x^k, y^k; T)\| + \max_{i \in T_k^c}\{\max(|\nabla_x L(x^k, y^k)_i| - |x^k|_{(s)}/\beta, 0)\}, \tag{19}$$

where the first term is to characterize the solution accuracy of the Newton step, and the second term is for the correctness of the support set. The algorithmic framework of LNA is summarized in Algorithm 1.

Algorithm 1 LNA for solving problem (1)

```
Step 0 Choose $\beta > 0, \varepsilon > 0$. Set $k = 0$ and choose the initial point $(x^0, y^0) \in \mathbb{R}^n \times \mathbb{R}$.
Step 1 Choose $T_k \in \mathbb{T}(x^k, y^k; \beta)$ by (6).
Step 2 If $\eta_\beta(x^k, y^k; T_k) \leq \varepsilon$, then stop. Otherwise, go to Step 3.
Step 3 Update $(x^{k+1}, y^{k+1})$ by (18), set $k = k + 1$ and go to Step 1.
```

Theorem 4.1: *Let x^* be a strong β-Lagrangian stationary point of problem (1) with the Lagrangian multiplier $y^* \in \mathbb{R}$ and Assumption 3.1 hold. Suppose that the initial point z^0 of sequence $\{z^k\} := \{(x^k, y^k)\}$ generated by LNA satisfies $z^0 \in N_S(z^*, \delta)$, where $\delta = \min\{\tilde{\delta}^*, 1/M^* L_2\}$ with $\tilde{\delta}^*, M^*$ defined in (16). Then for any $k \geq 0$,*

(i) $\lim_{k\to\infty} z^k = z^*$ *with quadratic convergence rate, i.e.*

$$||z^{k+1} - z^*|| \leq \frac{M^* L_2}{2}\|z^k - z^*\|^2.$$

(ii) *LNA terminates with accuracy ε when $k \geq \lceil \frac{\log_2(4\delta\sqrt{L_1{}^2+1}/\varepsilon)}{2} \rceil$.*

Here L_1 and L_2 are defined as in (11) and (12) by replacing δ^* with δ.

Proof: By employing Theorem 3.3, we know that the sequence $\{z^k\}$ generated by LNA is well-defined from the nonsingularity of $G(x^k, y^k; T_k)$ for all $k \geq 0$. In addition, we know from [21, Theorem 6] that LNA enjoys the locally quadratic convergence if $G(x^k, y^k; T_k)$ is nonsingular for all $k \geq 0$. Thus, utilizing Theorem 3.3, the desired properties (i) and (ii) are consequences of [21, Theorem 6]. ■

5. Numerical experiments

This section reports some numerical experiments on synthetic data and the real Hypergraph data. All experiments are implemented in MATLAB R2020b, running on a laptop computer of 8GB memory and Inter(R) Core(TM) i7 2.8Ghz CPU. Our codes are available at https://github.com/BJTUShuaiLi/LNA-for-TSPCA-.

We verify the effectiveness of the proposed TSPCA by comparing it with two approaches: alternating direction method of multipliers (ADMM) for nuclear norm penalty problem [7] and GloptiPoly3 (GLP) [22]. In [7], the even order tensor PCA problem without sparsity constraint is reformulated as the following optimization

$$\begin{array}{ll} \min_{X\in\mathbb{R}^{n^d\times n^d}} & -\mathrm{Tr}(\boldsymbol{A}_m\boldsymbol{X}) + \rho\|\boldsymbol{X}\|_* \\ \text{s.t.} & \boldsymbol{X} \in C, \end{array} \tag{20}$$

where $\boldsymbol{A}_m := M(\mathcal{A}) \in T_{2,n^d}$ and $\boldsymbol{X} := M(x^{2d}) \in T_{2,n^d}$ are the mode-$(1,\dots,d)$ unfolding matrices of $\mathcal{A}$ and x^{2d}, respectively. $C := \{\boldsymbol{X} \in S_{2,n^d} | \mathrm{Tr}(\boldsymbol{X}) = 1, M^{-1}(\boldsymbol{X}) \in S_{2d,n}\}$, $\rho > 0$ is a regularization parameter. By introducing the auxiliary variable $\boldsymbol{Y} = \boldsymbol{X}$, the Lagrange multiplier $\boldsymbol{\Lambda}$ and the penalty parameter $\mu > 0$, the iterative scheme of ADMM for solving (20) is given by

$$\begin{cases} \boldsymbol{X}^{k+1} := \arg\min\limits_{X\in C} -\mathrm{Tr}\left(\boldsymbol{A}_m\boldsymbol{Y}^k\right) + \rho\left\|\boldsymbol{Y}^k\right\|_* - \left\langle \boldsymbol{\Lambda}^k, \boldsymbol{X} - \boldsymbol{Y}^k\right\rangle + \dfrac{1}{2\mu}\left\|\boldsymbol{X} - \boldsymbol{Y}^k\right\|_F^2, \\ \boldsymbol{Y}^{k+1} := \arg\min\limits_{Y} -\mathrm{Tr}\left(\boldsymbol{A}_m Y\right) + \rho\|\boldsymbol{Y}\|_* - \left\langle \boldsymbol{\Lambda}^k, \boldsymbol{X}^{k+1} - \boldsymbol{Y}\right\rangle \\ \quad + \dfrac{1}{2\mu}\left\|\boldsymbol{X}^{k+1} - \boldsymbol{Y}\right\|_F^2, \\ \boldsymbol{\Lambda}^{k+1} := \boldsymbol{\Lambda}^k - (\boldsymbol{X}^{k+1} - \boldsymbol{Y}^{k+1})/\mu. \end{cases}$$

Throughout our experiments, the parameters used in LNA are chosen as $\beta = 0.01$, $y^0 = 1$. We terminate LNA whenever $\eta_\beta(x^k, y^k; T_k) \le 10^{-6}$ or iteration k reaches 1000. For ADMM, the parameters are chosen as $\mu = 0.5$ and $\rho = 10$, and we terminate it whenever

$$\frac{\|\boldsymbol{X}^k - \boldsymbol{X}^{k-1}\|_F}{\|\boldsymbol{X}^{k-1}\|_F} + \left\|\boldsymbol{X}^k - \boldsymbol{Y}^k\right\|_F \le 10^{-6}.$$

In the following testing examples, we record the comparisons between LNA and other algorithms in the average relative error (Re), CPU time (Time) and the number of iterations (Iter). Here, Re is defined as

$$\mathrm{Re} = \left\|\hat{x} - x^*\right\| / \left\|x^*\right\|,$$

where $\hat{x}$ is the optimal solution of problem (1) produced by one algorithm. The symbol '-' is used to represent the cases that Time reaches 1 hour.

5.1. Synthetic data

In this subsection, we utilize the synthetic data in Example 3.2 to test the effectiveness of our approach.

Table 2. Numerical comparison of Example 5.1.

n	s	Re(LNA|GLP|ADMM)	Time(s)(LNA|GLP|ADMM)	Iter(LNA|GLP|ADMM)
5	1	**1.94e-12**|6.39e-06|8.30e-07	0.142|1.185|4.342	4|8|409
10	1	**1.16e-11**|7.89e-06|1.43e-06	0.145|3.455|59.632	5|8|1452
15	1	**3.34e-11**|4.82e-05|3.89e-06	0.164|108.183|305.023	5|8|3194
	2	**1.62e-10**|5.88e-05|2.33e-06	0.175|114.653|313.056	5|9|3212
20	1	**1.19e-11**|3.64e-06|3.01e-06	0.186|2056.337|1121.042	4|8|5416
	2	**1.62e-10**|5.32e-05|3.00e-06	0.180|2367.227|1138.193	5|9|5523
30	1	1.93e-10|-|-	0.184|-|-	4|-|-
	2	1.43e-11|-|-	0.164|-|-	7|-|-
	3	9.09e-11|-|-	0.193|-|-	5|-|-
50	1	4.33e-10|-|-	0.168|-|-	5|-|-
	3	7.72e-11|-|-	0.170|-|-	5|-|-
	5	3.35e-09|-|-	0.188|-|-	4|-|-
100	1	3.59e-11|-|-	0.187|-|-	5|-|-
	5	8.32e-11|-|-	0.204|-|-	6|-|-
	10	8.87e-11|-|-	0.193|-|-	5|-|-

Example 5.1 (Third-order random tensors): We use the following pseudo MATLAB codes to generate third-order random tensors $\mathcal{A} = \sum_{i=1}^{R} \alpha_i(u_i)^3$ claimed in Example 3.2, the ground truth x^* and the initial point x^0 adopted in LNA.

$$u_1 = u_2 = u_3 = \texttt{zeros(n,1)}, \alpha = [3,2,1], \Gamma = \texttt{randperm(n,s)},$$
$$u_{1\Gamma(1,s)} = \texttt{randn(s,1)},$$

$u_{2\Gamma(s+1,n-1)} = \texttt{randn(n-s-1,1)}$, $u_{3\Gamma(n)} = \texttt{randn(1)}$, $\mathcal{A} = \sum_{i=1}^{3} \alpha_i(u_i/||u_i||)^3, x^* = u_1, x^0 = x^* - 0.1 \cdot \texttt{rand(n,1)}$.

Example 5.2 (Fourth-order random tensors): Let $\mathcal{A} = \sum_{i=1}^{R} \alpha_i(u_i)^4$ be a fourth-order random tensor. We generate the tensors $\mathcal{A}$, the ground truth x^* and the initial point x^0 adopted in LNA in the same fashion as Example 5.1, in which $u_i, i \in [3]$ are generated from the uniform distribution $U(0,1)$.

Set the sparsity $s = \lceil 0.01n \rceil$, $\lceil 0.05n \rceil$, $\lceil 0.1n \rceil$ in the above two examples. Numerical results are collected in Tables 2 and 3. As shown in Table 2, when $n = 5, 10, 15, 20$, LNA has the significant superiority than GLP and ADMM in terms of Re and Time. For the testing instances of $n \geq 30$, GLP and ADMM report more than an hour of Time, while LNA remains stable in Re, Time and Iter. Moreover, we can see from Table 3 that LNA gives the highest solution accuracy and the least CPU time in the instances of $n = 5, 10, 15, 20$. See, e.g. for the case of $n = 20$ and $s = 2$, Re and Time reported by LNA are no more than $1/(2.5 \times 10^6)$ and 1/5000 of those by GLP and ADMM, respectively. With the increase of n, Time reported by GLP and ADMM reaches an hour, while LNA maintains excellent performances with the high solution accuracy and short CPU time under different n and s.

Table 3. Numerical comparison of Example 5.2.

n	s	Re(LNA|GLP|ADMM)	Time(s)(LNA|GLP|ADMM)	Iter(LNA|GLP|ADMM)
5	1	**1.87e-11**|1.56e-05|8.38e-07	0.151|1.228|4.239	5|8|372
10	1	**3.56e-12**|6.77e-06|1.41e-06	0.202|3.778|54.329	6|7|1235
15	1	**1.88e-11**|2.03e-05|3.76e-06	0.174|101.625|243.852	6|9|2548
	2	**6.55e-12**|1.12e-05|6.33e-06	0.188|107.332|262.714	5|9|2699
20	1	**6.99e-12**|4.80e-06|2.43e-06	0.211|2270.975|1019.762	6|7|4123
	2	**1.51e-11**|4.42e-05|7.32e-05	0.193|2525.496|998.904	5|7|4289
30	1	4.01e-10|-|-	0.194|-|-	5|-|-
	2	6.09e-10|-|-	0.197|-|-	5|-|-
	3	6.19e-11|-|-	0.218|-|-	5|-|-
50	1	6.76e-10|-|-	0.315|-|-	5|-|-
	3	3.78e-10|-|-	0.323|-|-	6|-|-
	5	1.48e-11|-|-	0.324|-|-	5|-|-
100	1	3.78e-11|-|-	3.173|-|-	6|-|-
	5	5.92e-11|-|-	3.604|-|-	6|-|-
	10	1.96e-11|-|-	4.143|-|-	7|-|-

5.2. Hypergraph data

In this subsection, we test the effectiveness of our method on two types of hypergraph data sets, including the structured hypergraphs in Example 5.4 and the real social network data in Example 5.5.

For the following testing examples, the objective function value (Obj) is also adopted to measure the performances of three competitors. One aims to calculate the sparse principal components of the adjacency tensors of hypergraphs, where the adjacency tensor of a k-uniform hypergraph is defined as follows.

Definition 5.3: Let $G = (V, E)$ be a k-uniform hypergraph (each edge connects exactly k vertices) with the vertex set $V = [n]$ and the edge set $E = \{e_1, \ldots, e_m\}$. The adjacency tensor of G is defined as the k-th order n-dimensional tensor $\mathcal{A}$ whose $(i_1, \ldots, i_k)$-entry is:

$$a_{i_1\ldots i_k} := \begin{cases} \frac{1}{(k-1)!}, & \text{if } \{i_1, \ldots, i_k\} \in E, \\ 0, & \text{otherwise.} \end{cases} \tag{21}$$

Example 5.4 (Structured hypergraphs): This example considers three important structured hypergraphs: sunflower, hypercycle and squid. A k-uniform hypergraph $G = (V, E)$ is called

- a sunflower if $e_i = \{(i-1)(k-1)+1, (i-1)(k-1)+2, \ldots, i(k-1), n\}$ for all $i \in [m]$ and $n = m(k-1)+1$;
- a hypercycle if $e_i = \{(i-1)(k-1)+1, (i-1)(k-1)+2, \ldots, i(k-1)+1\}$ for all $i \in [m-1], e_m = \{(m-1)(k-1)+ 1, \ldots, m(k-1), 1\}$ and $n = m(k-1)$;
- a squid if $e_i = \{(i-1)k+1, (i-1)k+2, \ldots, ik\}$ for all $i \in [m-1], e_m = \{(i-1)k+1 \mid i \in [m]\}, m = k$ and $n = (m-1)k+1$.

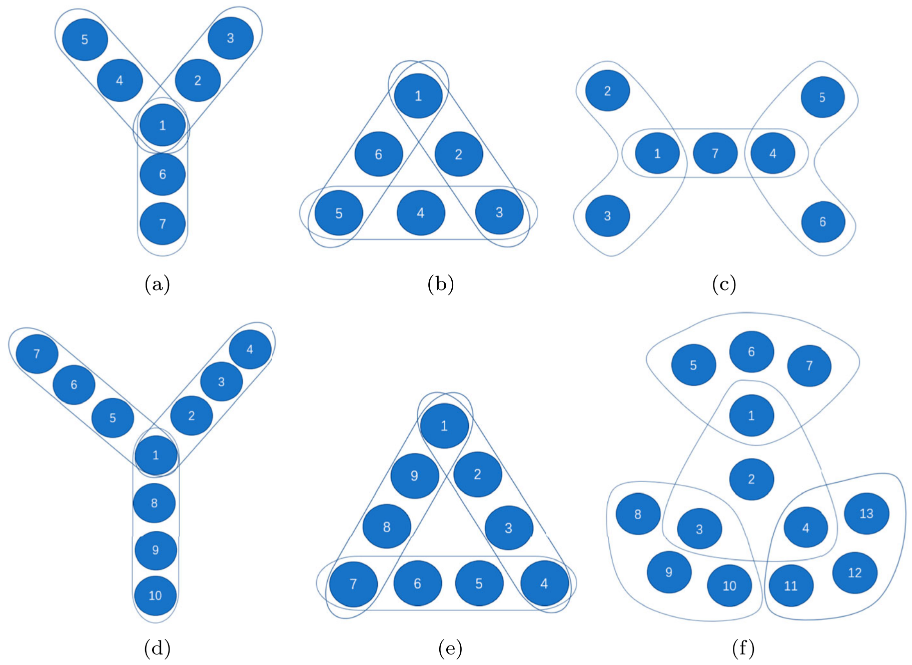

Figure 1. Illustration of structured hypergraphs. (a) and (d): sunflower; (b) and (e): hypercycle; (c) and (f): squid. (a) $n = 7, m = 3, k = 3$, (b) $n = 6, m = 3, k = 3$, (d) $n = 10, m = 3, k = 4$, (e) $n = 9, m = 3, k = 4$ and (f) $n = 13, m = 3, k = 4$.

Figure 1 gives the illustration of the above three structured hypergraphs. In this example, the corresponding supersymmetric tensor $\mathcal{A} \in S_{k,n}$ in the TSPCA problem (1) is constructed by (21) for the given n, m, k and edge set $E = \{e_1, \dots, e_m\}$.

Example 5.5 (Social networks): We use three real data sets about social networks (email-Enron, contact-primary-school and contact-high-school)[1]. In the email-Enron data set, each vertex corresponds to an email address, and each hyperedge consists of the sender address and all recipient addresses on this email. In the contact-primary-school and contact-high-school data sets, each node is a person and each hyperedge is a set of persons in close proximity to each other.

Figure 2 shows the 3-uniform hypergraph generated from the above real datasets. Taking email-Enron as an example, we pick out the hyperedges with three vertices to form a 3-uniform hypergraph $G = (V, E)$, whose vertex set V consists of 148 email addresses, and edge set E consists of all 324 hyperedges with three email addresses. The corresponding supersymmetric tensor $\mathcal{A} \in S_{3,148}$ of the 3-uniform hypergraph G can be generated by (21).

Table 4 records the numerical results of Example 5.4. As shown in Table 4, LNA reports the comparable Obj with GLP and ADMM, while the time of LNA

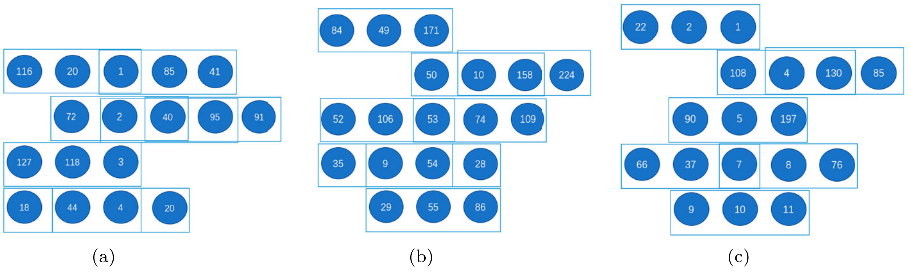

Figure 2. The hypergraphs constructed by (a) email-Enron ($n = 148$, $m = 324$, $k = 3$); (b) contact-primary-school ($n = 242$, $m = 4622$, $k = 3$); (c) contact-high-school ($n = 327, m = 2126, k = 3$).

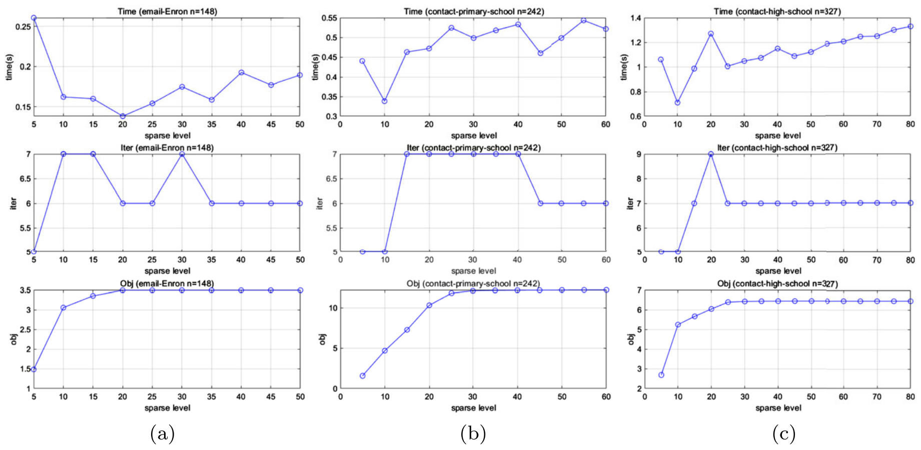

Figure 3. Numerical results for social network data. (a) email-Enron(n = 148), (b) contact-primary-school(n = 242) and (c) contact-high-school(n = 327).

is much less than the other two methods for all three cases of structured hypergraphs. Moreover, Figure 3 illustrates the results of Example 5.5. It can be seen from Figure 3 that LNA gives the superior performances on all three real social network data sets. With the increase of sparsity s, LNA performs very stable in terms of running time, which is basically less than 1.5 seconds for all testing instances. In addition, the time of GLP and ADMM is more than 1 hour when solving tensors with $n = 30$. Obviously, the time advantage of LNA is even more significant when dealing with large-scale data.

Overall, LNA is an efficient and stable approach in solving our proposed TSPCA problem (1), it can achieve the high solution accuracy with a low time cost.

6. Conclusions

In this paper, we established the optimization model for tensor SPCA, in which the original ℓ_0-norm constraint is used for solving sparse principal components.

Table 4. Numerical comparison of Example 5.4.

	m	n	Obj(LNA|GLP|ADMM)	Time(s)(LNA|GLP|ADMM)
sunflower	3	7	0.577|0.577|0.568	0.157|1.478|15.076
	4	10	0.250|0.250|0.250	0.180|17.732|26.203
hypercycle	3	6	0.577|0.667|0.582	0.160|1.327|8.382
	4	9	0.250|0.250|0.250	0.171|11.657|19.553
squid	3	7	0.577|0.577|0.568	0.152|1.373|15.072
	4	13	0.250|0.250|0.250	0.170|66.747|70.269

For the formulated TSPCA optimization problem, the first-order optimality condition was established by utilizing the sparse projection operator. Under some mild conditions, we showed the nonsingularity of Jacobian matrix for the corresponding Lagrangian stationary equation system. Based on this, the LNA algorithm was adopted to solve TSPCA. Meanwhile, we established its locally quadratic convergence. Numerical experiments showed the effectiveness of our proposed method comparing with two state-of-the-art approaches.

Note

1. The social network data are available in Temporal higher order networks (hypergraphs) at https://www.cs.cornell.edu/ ∼ arb/data/.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by the Beijing Natural Science Foundation (Grant No. Z190002) and the National Natural Science Foundation of China (Grant No.12271022).

References

[1] Wang H, Ahuja N. Compact representation of multidimensional data using tensor rank-one decomposition. In: Proceedings of the 17th International Conference on Pattern Recognition; 2004; Vol. 1(5). p. 44–47.

[2] Qi L, Yu G, Wu EX. Higher order positive semidefinite diffusion tensor imaging. SIAM J Imaging Sci. 2010;3(3):416–433. doi: 10.1137/090755138

[3] Cardoso JF. High-order contrasts for independent component analysis. Neural Comput. 1999;11(1):157–192. doi: 10.1162/089976699300016863

[4] Hu S, Qi L. Algebraic connectivity of an even uniform hypergraph. J Comb Optim. 2012;24(4):564–579. doi: 10.1007/s10878-011-9407-1

[5] Zhang T, Golub GH. Rank-one approximation to high order tensors. SIAM J Matrix Anal Appl. 2001;23(2):534–550. doi: 10.1137/S0895479899352045

[6] Qi L, Wang F, Wang Y. Z-eigenvalue methods for a global polynomial optimization problem. Math Program. 2009;118(2):301–316. doi: 10.1007/s10107-007-0193-6

[7] Jiang B, Ma S, Zhang S. Tensor principal component analysis via convex optimization. Math Program. 2015;150(2):423–457. doi: 10.1007/s10107-014-0774-0

[8] Huang J, Huang DZ, Yang Q, et al. Power iteration for tensor PCA. J Mach Learn Res. 2022;23(128):1–47. doi: 10.5555/3586589.3586717

[9] Beck A, Vaisbourd Y. The sparse principal component analysis problem: optimality conditions and algorithms. J Optim Theory Appl. 2016;170(1):119–143. doi: 10.1007/s10957-016-0934-x

[10] Bertsimas D, Cory-Wright R, Pauphilet J. Solving large-scale sparse PCA to certifiable (near) optimality. J Mach Learn Res. 2022;23(13):1–35. doi: 10.5555/3586589.3586602

[11] Lu Z, Zhang Y. An augmented Lagrangian approach for sparse principal component analysis. Math Program. 2012;135(1-2):149–193. doi: 10.1007/s10107-011-0452-4

[12] Fan J, Li R. Variable selection via nonconcave penalized likelihood and its oracle properties. J Am Stat Assoc. 2001;96(456):1348–1360. doi: 10.1198/016214501753382273

[13] Gotoh JY, Takeda A, Tono K. DC formulations and algorithms for sparse optimization problems. Math Program. 2018;169(1):141–176. doi: 10.1007/s10107-017-1181-0

[14] Chen X, Ge D, Wang Z, et al. Complexity of unconstrained $\ell_2 - \ell_p$ minimization. Math Program. 2014;143(1-2):371–383. doi: 10.1007/s10107-012-0613-0

[15] Blumensath T, Davies ME. Iterative hard thresholding for compressed sensing. Appl Comput Harmon Anal. 2009;27(3):265–274. doi: 10.1016/j.acha.2009.04.002

[16] Needell D, Tropp JA. CoSaMP: iterative signal recovery from incomplete and inaccurate samples. Appl Comput Harmon Anal. 2009;26(3):301–321. doi: 10.1016/j.acha.2008.07.002

[17] Pati YC, Rezaiifar R, Krishnaprasad PS. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In: Proceedings of 27th Asilomar Conference on Signals, Systems and Computers. IEEE; 1993. p. 40–44.

[18] Chen J, Gu Q. Fast newton hard thresholding pursuit for sparsity constrained nonconvex optimization. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2017; p. 757–766.

[19] Yuan X, Li P, Zhang T. Gradient hard thresholding pursuit. J Mach Learn Res. 2017;18(1):6027–6069. doi: 10.5555/3122009.3242023

[20] Zhou S, Xiu N, Qi H. Global and quadratic convergence of newton hard-thresholding pursuit. J Mach Learn Res. 2021;22(12):1–45. doi: 10.5555/3546258.3546270

[21] Zhao C, Xiu N, Qi H, et al. A Lagrange–Newton algorithm for sparse nonlinear programming. Math Program. 2022;195(1-2):903–928. doi: 10.1007/s10107-021-01719-x

[22] Henrion D, Lasserre JB, Löfberg J. Gloptipoly 3: moments, optimization and semidefinite programming. Optim Methods Softw. 2009;24(4-5):761–779. doi: 10.1080/10556780802699201