Journal of Optimization Theory and Applications (2022) 194:821–851
https://doi.org/10.1007/s10957-022-02050-x

On Approximation Algorithm for Orthogonal Low-Rank Tensor Approximation

Yuning Yang[1]

Received: 8 May 2021 / Accepted: 12 May 2022 / Published online: 28 June 2022

Abstract
This work studies solution methods for approximating a given tensor by a sum of R rank-1 tensors with one or more of the latent factors being orthonormal. Such a problem arises from applications such as image processing, joint singular value decomposition, and independent component analysis. Most existing algorithms are of the iterative type, while algorithms of the approximation type are limited. By exploring the multilinearity and orthogonality of the problem, we introduce an approximation algorithm in this work. Depending on the computation of several key subproblems, the proposed approximation algorithm can be either deterministic or randomized. The approximation lower bound is established, both in the deterministic and the expected senses. The approximation ratio depends on the size of the tensor, the number of rank-1 terms, and is independent of the problem data. When reduced to the rank-1 approximation case, the approximation bound coincides with those in the literature. Moreover, the presented results fill a gap left in Yang (SIAM J Matrix Anal Appl 41:1797–1825, 2020), where the approximation bound of that approximation algorithm was established when there is only one orthonormal factor. Numerical studies show the usefulness of the proposed algorithm.

Keywords Tensor · Orthogonality · Approximation algorithm · Approximation bound · Polar decomposition

Mathematics Subject Classification 90C26 · 15A18 · 15A69 · 41A50

Communicated by Guoyin Li.

✉ Yuning Yang
yyang@gxu.edu.cn

[1] College of Mathematics and Information Science, Guangxi University, Nanning 530004, China

1 Introduction

Tensors (hypermatrices) play an important role in signal processing, data analysis, statistics, and machine learning nowadays, key to which are tensor decompositions/approximations [4, 6, 20, 37]. Canonical polyadic (CP) decomposition factorizes a data tensor into some rank-1 components. Such a decomposition can be unique under mild assumptions [21, 34, 36]. However, the degeneracy of the associated optimization problem requires one to impose additional constraints, such as nonnegativity, angularity, and orthogonality constraints [19, 26, 27]. Orthogonal CP approximation requires that one or more of the latent factors be orthonormal [11, 19], modeling applications arising from image processing, joint SVD, independent component analysis, DS-CDMA systems, and so on [5, 7, 32, 35, 38, 41, 42].

Most existing algorithms for orthogonal CP approximation are iterative ones [3, 11, 16, 23, 24, 28, 31, 41, 47, 48]; just to name a few. Specifically, alternating least squares (ALS) algorithms were developed in [3, 41, 47]: Chen and Saad [3] considered the model that all the factors are orthonormal and established the convergence of ALS under certain local assumptions; Sørensen et al. [41] considered the case that one of the factors is orthonormal, while the convergence was not provided. Such a gap was filled in [47] for generic tensors. Guan and Chu [11] considered the case that one or more of the factors are orthonormal, proposed a method that simultaneously updates two vectors corresponding to two factors, and proved the global convergence for generic tensors. Shifted type ALS algorithms were developed in [16, 31, 48]: Pan and Ng [31] considered symmetric tensors and established global convergence for generic tensors; Hu and Ye [16] studied the case where all factors are orthonormal and established the global and linear convergence of the algorithm; Yang [48] established the global convergence when the orthonormal factors are one or more than one. Local linear convergence of ALS was proved in [44] under certain positive definiteness assumptions. In the rank-1 approximation case, global linear convergence was shown to be valid for generic tensors [15]. Martin and Van Loan [28] and Li et al. [23] proposed Jacobi-type algorithms, where Li et al. [23] considered symmetric tensors and established the global convergence of the developed algorithms. Besides, Sørensen et al. [41] proposed a simultaneous matrix diagonalization approach for the problem with one orthonormal factor.

As a nonconvex problem, the solution quality of iterative algorithms depends on the initializer. To find a good initializer for iterative algorithms, Yang [48, Procedure 3.1] developed an approximation algorithm that first finds the orthonormal factors by using the celebrated higher-order SVD (HOSVD) [8] and then computes the non-orthonormal ones by solving rank-1 approximation problems. However, the approximation bound of Yang [48, Procedure 3.1] was only established when the number of orthonormal latent factors is one.

To fill this gap, by further exploring the multilinearity and orthogonality of the problem, we modify [48, Procedure 3.1] to devise a new approximation algorithm. A main feature of the proposed algorithm is that it can be either deterministic or randomized, depending on how to deal with a set of subproblems. Specifically, two deterministic and one randomized procedures are considered. The approximation lower bound is derived, both in the deterministic and in the expected senses, regardless of how many latent orthonormal factors there are. The approximation ratio depends on the size of the tensor, the number of rank-1 terms, and is independent of the problem data. Reducing to the rank-1 approximation case, the approximation ratio is of the same order as in [13, 51]. We provide numerical studies to show the efficiency of the introduced algorithm and its usefulness in tensor recovery and clustering.

In fact, approximation algorithms have already been widely studied for tensor approximations. For example, HOSVD, sequentially truncated HOSVD [45], and hierarchical HOSVD [10] are also approximation algorithms for the Tucker decomposition. On the other hand, several randomized approximation algorithms have been proposed in recent years for Tucker decomposition or t-SVD [1, 2, 29, 50]. The solution quality of the aforementioned algorithms was measured via error bounds. In the context of tensor best rank-1 approximation and related polynomial optimization problems, deterministic or randomized approximation algorithms were developed [9, 12, 13, 18, 39, 40], where the solution quality was measured by approximation bounds; our algorithm extends these to orthogonal rank-R approximations.

The rest of this work is organized as follows. Preliminaries are given in Sect. 2. The new approximation algorithm is introduced in Sect. 3, while the approximation bound is derived in Sect. 4. Section 5 provides numerical results, and Sect. 6 draws some conclusions.

2 Preliminaries

Throughout this work, vectors are written as boldface lowercase letters $(\mathbf{x}, \mathbf{y}, \ldots)$, matrices correspond to italic capitals $(A, B, \ldots)$, and tensors are written as calligraphic capitals $(\mathcal{A}, \mathcal{B}, \cdots)$. $\mathbb{R}^{n_1\times\cdots\times n_d}$ denotes the space of $n_1 \times \cdots \times n_d$ real tensors. For two tensors $\mathcal{A}, \mathcal{B}$ of the same size, their inner product $\langle \mathcal{A}, \mathcal{B}\rangle$ is given by the sum of entry-wise products. The Frobenius (or Hilbert–Schmidt) norm of $\mathcal{A}$ is defined by $\|\mathcal{A}\|_F = \langle \mathcal{A}, \mathcal{A}\rangle^{1/2}$. $\otimes$ denotes the outer product. For $\mathbf{u}_j \in \mathbb{R}^{n_j}$, $j = 1, \ldots, d$, $\mathbf{u}_1 \otimes \cdots \otimes \mathbf{u}_d$ denotes a rank-1 tensor in $\mathbb{R}^{n_1\times\cdots\times n_d}$, which is written as $\bigotimes_{j=1}^d \mathbf{u}_j$ for short. For $j = 1, \ldots, d$ and for each j, if we are given R vectors of size n_j: $\mathbf{u}_{j,1}, \ldots, \mathbf{u}_{j,R}$, we usually collect them into a matrix $U_j = [\mathbf{u}_{j,1}, \ldots, \mathbf{u}_{j,R}] \in \mathbb{R}^{n_j\times R}$.

The mode-j unfolding of $\mathcal{A} \in \mathbb{R}^{n_1\times\cdots\times n_d}$, denoted as $A_{(j)}$, is a matrix in $\mathbb{R}^{n_j\times\prod_{k\neq j}^d n_k}$. The product between a tensor $\mathcal{A} \in \mathbb{R}^{n_1\times\cdots\times n_d}$ and a vector $\mathbf{u}_j \in \mathbb{R}^{n_j}$ with respect to the j-th mode, written as $\mathcal{A} \times_j \mathbf{u}_j^\top$, is a tensor in $\mathbb{R}^{n_1\times\cdots\times n_{j-1}\times n_{j+1}\times\cdots\times n_d}$.

Given a d-th order tensor $\mathcal{A} \in \mathbb{R}^{n_1\times\cdots\times n_d}$, a positive integer R, decision matrices (factors) $U_j = [\mathbf{u}_{j,1}, \ldots, \mathbf{u}_{j,R}] \in \mathbb{R}^{n_j\times R}$, $j = 1, \ldots, d$, and coefficients $\sigma_i \in \mathbb{R}$, $i = 1, \ldots, R$, the orthogonal CP approximation under consideration can be written as [11]:

$$
\begin{aligned}
\min_{U_1,\ldots,U_d,\sigma_1,\ldots,\sigma_R} \quad & \left\| \mathcal{A} - \sum\nolimits_{i=1}^R \sigma_i \bigotimes\nolimits_{j=1}^d \mathbf{u}_{j,i} \right\|_F^2 \\
\text{s.t.} \quad & \mathbf{u}_{j,i}^\top \mathbf{u}_{j,i} = 1,\ j = 1, \ldots, d-t,\ 1 \le i \le R, \\
& U_j^\top U_j = I,\ j = d-t+1, \ldots, d,\ \sigma_i \in \mathbb{R},
\end{aligned}
\tag{2.1}
$$

where $1 \le t \le d$ denotes the number of latent orthonormal factors. The constraints mean that the last t latent factors are required to be orthonormal, while the first $d-t$ ones are not. However, due to the presence of the scalars σ_i, each column $\mathbf{u}_{j,i}$ of the first $d-t$ factors can be normalized. Using orthogonality, (2.1) can be equivalently formulated as a maximization problem [11]:

$$\begin{array}{ll} \max_{U_1,\ldots,U_d} & G(U_1,\ldots,U_d) := \sum_{i=1}^R \left\langle \mathcal{A}, \bigotimes_{j=1}^d \mathbf{u}_{j,i} \right\rangle^2 \\ \text{s.t.} & \mathbf{u}_{j,i}^\top \mathbf{u}_{j,i} = 1,\, j = 1,\ldots,d-t,\, 1 \le i \le R, \\ & U_j^\top U_j = I,\, j = d-t+1,\ldots,d, \end{array} \tag{2.2}$$

where the variables σ_i's have been eliminated. The approximation algorithms and the corresponding analysis studied here are mainly focused on (2.2) .

Let $V \in \mathbb{R}^{m\times n}$, $m \ge n$. The polar decomposition of V is to decompose it into two matrices $U \in \mathbb{R}^{m\times n}$ and $H \in \mathbb{R}^{n\times n}$ such that $V = UH$, where U is columnwisely orthonormal, and H is a symmetric positive semidefinite matrix. Its relation with SVD is given below.

Theorem 2.1 (c.f. [14]) *Let $V \in \mathbb{R}^{m\times n}$, $m \ge n$. Then, there exist $U \in \mathbb{R}^{m\times n}$ and a unique symmetric positive semidefinite matrix $H \in \mathbb{R}^{n\times n}$ such that*

$$V = UH,\ U^\top U = I \in \mathbb{R}^{n\times n}.$$

(U, H) is the polar decomposition of V. If $\operatorname{rank}(V) = n$*, then H is symmetric positive definite and U is uniquely determined.*

Furthermore, let $H = Q\Lambda Q^\top$, $Q, \Lambda \in \mathbb{R}^{n\times n}$ be the eigenvalue decomposition of H, namely, $Q^\top Q = QQ^\top = I$, $\Lambda = \operatorname{diag}(\lambda_1,\ldots,\lambda_n)$ be a diagonal matrix where $\lambda_1 \ge \cdots \ge \lambda_n \ge 0$. Then, $U = PQ^\top$, and $V = P\Lambda Q^\top$ is a reduced SVD of V.

Let $V = [\mathbf{v}_1,\ldots,\mathbf{v}_n]$ and $U = [\mathbf{u}_1,\ldots,\mathbf{u}_n]$ where $\mathbf{v}_i, \mathbf{u}_i \in \mathbb{R}^m$, $i = 1,\ldots,n$. The following two lemmas are useful.

Lemma 2.1 *Let $U \in \mathbb{R}^{m\times n}$ and $H \in \mathbb{R}^{n\times n}$ be two matrices generated by the polar decomposition of $V \in \mathbb{R}^{m\times n}$ such that $V = UH$, where U is columnwisely orthonormal and H is symmetric positive semidefinite. Let $\mathbf{v}_i$ and $\mathbf{u}_i$ be defined as above. Then, it holds that*

$$\langle \mathbf{u}_i, \mathbf{v}_i\rangle \ge 0,\ i = 1,\ldots,m.$$

Proof Since $V = UH$ and $U^\top U = I$, we have $U^\top V = H$, which means that $\langle \mathbf{u}_i, \mathbf{v}_i\rangle$ is exactly the i-th diagonal entry of H. As H is positive semidefinite, its diagonal entries are nonnegative. The desired result follows. □

Denote $\|\cdot\|_*$ as the nuclear norm of a matrix, i.e., the sum of its singular values. From Theorem 2.1, we easily see that:

Lemma 2.2 *Let U and V be defined as in Theorem 2.1. Then, $\langle U, V\rangle = \|V\|_*$.*

Denote $G_{\max}$ as the maximal value of (2.2) and $\lambda_i(A_{(d)})$ the i-th largest singular value of $A_{(d)}$. We have the following result, whose proof is in Appendix A.

Lemma 2.3 *It holds that* $\sum_{i=1}^R \lambda_i(A_{(d)})^2 \geq G_{\max}$ *in* (2.2).

3 Approximation Algorithm

The approximation algorithm proposed in this work inherits certain ideas from Yang [48, Procedure 3.1], and so we first briefly recall its strategy. [48, Procedure 3.1] obtained an approximation solution to (2.2) as follows: One first applies the truncated HOSVD [8] to get $U_d, \ldots, U_{d-t+1}$. That is to say, for $j = d, \ldots, d-t+1$, one unfolds $\mathcal{A}$ to $A_{(j)} \in \mathbb{R}^{n_j \times \prod_{l\neq j}^d n_l}$ and performs the truncated SVD to get its left leading R singular vectors $\mathbf{u}_{j,1}, \ldots, \mathbf{u}_{j,R}$, forming the j-th factor $U_j = [\mathbf{u}_{j,1}, \ldots, \mathbf{u}_{j,R}]$. Once $U_d, \ldots, U_{d-t+1}$ are obtained, one then approximately solves R tensor best rank-1 approximation problems:

$$\begin{array}{rl} \max_{\mathbf{u}_{1,1},\ldots,\mathbf{u}_{d-t,R}} & \left\langle \mathcal{A} \bigotimes_{j=d-t+1}^d \mathbf{u}_{j,i}, \bigotimes_{j=1}^{d-t} \mathbf{u}_{j,i} \right\rangle \\ \text{s.t.} & \mathbf{u}_{j,i}^\top \mathbf{u}_{j,i} = 1, \, j = 1, \ldots, d-t, \end{array} \tag{3.3}$$

where the data tensors $\mathcal{A} \bigotimes_{j=d-t+1}^d \mathbf{u}_{j,i} := \mathcal{A} \times_{d-t+1} \mathbf{u}_{d-t+1,i}^\top \times \cdots \times_d \mathbf{u}_{d,i}^\top \in \mathbb{R}^{n_1 \times \cdots \times n_{d-t}}$, $i = 1, \ldots, R$.

The approximation bound of the above strategy was established when $t = 1$ [48, Proposition 3.2]. However, when $t \geq 2$, it is difficult to build connections between orthonormal factors theoretically, making it hard to derive the approximation bound. Our new algorithm differs from [48] in the computation of the orthonormal factors $U_{d-1}, \ldots, U_{d-t+1}$. By taking (2.2) with $d = t = 3$ as an example, we state our idea below and depict the workflow in Fig. 1.

The computation of U_3 is the same as [48], namely, we let $U_3 := [\mathbf{u}_{3,1}, \ldots, \mathbf{u}_{3,R}]$ where $\mathbf{u}_{3,1}, \ldots, \mathbf{u}_{3,R}$ are the left R leading singular vectors of the unfolding matrix $A_{(d)} \in \mathbb{R}^{n_3 \times n_1 n_2}$. Then, finding U_2 can be divided into a *splitting* step and a *gathering* step. In the splitting step, with $\mathbf{u}_{3,1}, \ldots, \mathbf{u}_{3,R}$ at hand, we first compute R matrices $M_{2,1}, \ldots, M_{2,R}$, with

$$M_{2,i} := \mathcal{A} \times_3 \mathbf{u}_{3,i}^\top \in \mathbb{R}^{n_2 \times n_1}, \; i = 1, \ldots, R.$$

Then, using a procedure called `get_v_from_M` that will be specified later, we compute R vectors $\mathbf{v}_{2,i}$ of size n_2 from $M_{2,i}$ for each i. The principle of `get_v_from_M` is to retain as much of the information from $M_{2,i}$ as possible, and to satisfy some theoretical bounds. Three versions of such a procedure will be detailed later. In the gathering step, we first denote $V_2 := [\mathbf{v}_{2,1}, \ldots, \mathbf{v}_{2,R}] \in \mathbb{R}^{n_2 \times R}$, which may not be orthonormal and hence not feasible. To orthogonalize it, a straightforward idea is to find the nearest orthonormal matrix to V_2. This is equivalent to applying the polar decomposition to V_2, to obtain the orthonormal matrix $U_2 := [\mathbf{u}_{2,1}, \ldots, \mathbf{u}_{2,R}]$.

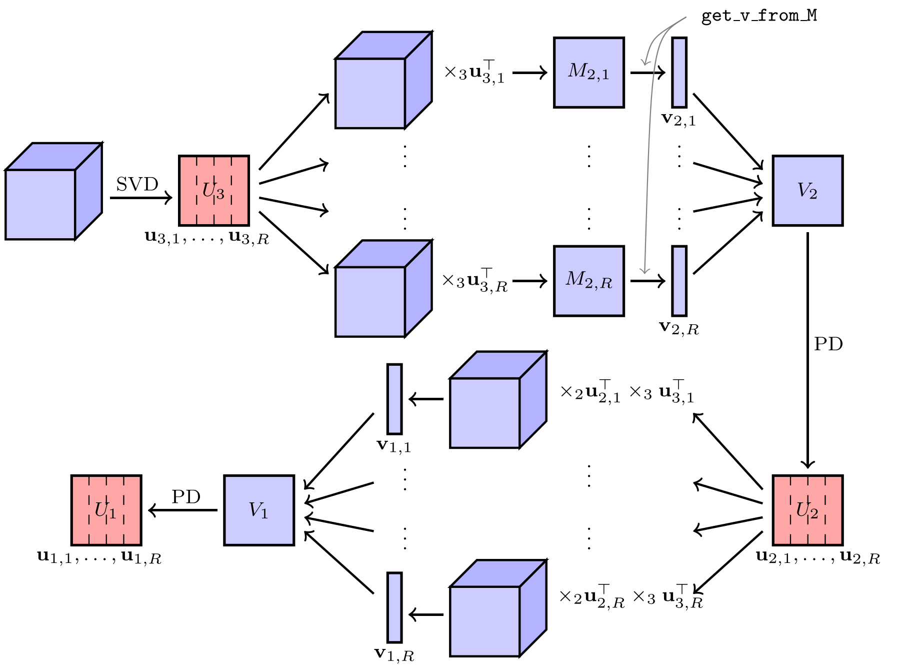

get_v_from_M
SVD
U_3
$\mathbf{u}_{3,1},\ldots,\mathbf{u}_{3,R}$
$\times_3\mathbf{u}_{3,1}^\top$
$\times_3\mathbf{u}_{3,R}^\top$
$M_{2,1}$
$M_{2,R}$
$\mathbf{v}_{2,1}$
$\mathbf{v}_{2,R}$
V_2
PD
U_2
$\mathbf{u}_{2,1},\ldots,\mathbf{u}_{2,R}$
$\times_2\mathbf{u}_{2,1}^\top\times_3\mathbf{u}_{3,1}^\top$
$\times_2\mathbf{u}_{2,R}^\top\times_3\mathbf{u}_{3,R}^\top$
$\mathbf{v}_{1,1}$
$\mathbf{v}_{1,R}$
V_1
PD
U_1
$\mathbf{u}_{1,1},\ldots,\mathbf{u}_{1,R}$

Fig. 1 Workflow of Algorithm 1 for approximately solving (2.2) when $d = 3$ and $t = 3$. PD is short for polar decomposition

Computing U_1 is similar. In the splitting step, with U_3 and U_2 at hand, we first compute $M_{1,i} := \mathcal{A} \times_2 \mathbf{u}_{2,i}^\top \times \mathbf{u}_{3,i}^\top \in \mathbb{R}^{n_1}$, $i = 1,\ldots,R$. Since $M_{1,i}$'s are already vectors, we immediately let $\mathbf{v}_{1,i} = M_{1,i}$. In the gathering step, we let $V_1 := [\mathbf{v}_{1,1},\ldots,\mathbf{v}_{1,R}]$ and then perform polar decomposition on V_1 to get U_1.

For general $d \geq 3$, we can similarly apply the above splitting step and gathering step to obtain $U_{d-1}, U_{d-2}, \ldots$, sequentially. In fact, the design of the splitting and gathering steps follows the principle that

$$\cdots \geq \sum\nolimits_{i=1}^{R} \left\langle \mathbf{u}_{j,i}, \mathbf{v}_{j,i} \right\rangle^2 \geq \alpha \sum\nolimits_{i=1}^{R} \left\langle \mathbf{u}_{j+1,i}, \mathbf{v}_{j+1,i} \right\rangle^2 \geq \cdots, \tag{3.4}$$

where $\alpha \in (0,1)$ and this will be studied in detail later. If $t < d$, i.e., there exist non-orthonormal constraints, then with $U_{d-t+1},\ldots,U_d$ at hand, we compute $U_1,\ldots,U_{d-t}$ via approximately solving R tensor rank-1 approximation problems (3.3). The whole algorithm is summarized in Algorithm 1.

Some remarks on Algorithm 1 are given as follows.

Remark 3.1 1. When $t = 1$, Algorithm 1 and [48, Procedure 3.1] are exactly the same if they take the same procedure `rank1approx`.

Algorithm 1 Approximation algorithm for (2.2)

```
Require: $\mathcal{A} \in \mathbb{R}^{n_1\times\cdots\times n_d}$, $d \geq 3$, $R \geq 1$, $t \geq 1$
1: Compute $\mathbf{u}_{d,1},\ldots,\mathbf{u}_{d,R}$ as the left leading $R$ unit singular vectors of the unfolding matrix $A_{(d)} \in \mathbb{R}^{n_d\times\prod_{k=1}^{d-1}n_k}$. Denote $U_d := [\mathbf{u}_{d,1},\ldots,\mathbf{u}_{d,R}]$
2: for $j = d-1 : -1 : d-t+1$ do
3:     % splitting step
4:     for $i = 1 : R$ do
5:         Compute the $j$-th order tensor
               $\mathcal{B}_{j,i} := \mathcal{A} \times_{j+1} \mathbf{u}_{j+1,i}^\top \times_{j+2} \cdots \times_d \mathbf{u}_{d,i}^\top \in \mathbb{R}^{n_1\times\cdots\times n_j}$
6:         if $j > 1$ then
7:             Unfold $\mathcal{B}_{j,i}$ to matrix $M_{j,i}$ as
                   $M_{j,i} := \texttt{reshape}\left(\mathcal{B}_{j,i}, n_j, \prod_{k=1}^{j-1} n_k\right) \in \mathbb{R}^{n_j\times\prod_{k=1}^{j-1}n_k}$
8:             Extract a vector $\mathbf{v}_{j,i}$ from $M_{j,i}$ as
                   $\mathbf{v}_{j,i} := \texttt{get\_v\_from\_M}(M_{j,i}) \in \mathbb{R}^{n_j}$   % to be introduced later
9:         else
10:                $\mathbf{v}_{j,i} := \mathcal{B}_{j,i} \in \mathbb{R}^{n_j}$
11:        end if
12:    end for
13:    % gathering step
14:    Denote $V_j := [\mathbf{v}_{j,1},\ldots,\mathbf{v}_{j,R}] \in \mathbb{R}^{n_j\times R}$
15:    Compute the polar decomposition of $V_j$ to obtain orthonormal $U_j$
           $U_j = [\mathbf{u}_{j,1},\ldots,\mathbf{u}_{j,R}] := \texttt{polar\_decomp}(V_j)$
16: end for                 % end of the computation of orthonormal factors
17: if $t < d$ then
18:    for $i = 1 : R$ do
19:        Compute a rank-1 approximation solution to the tensor $\mathcal{A} \times_{d-t+1} \mathbf{u}_{d-t+1,i}^\top \times \cdots \times_d \mathbf{u}_{d,i}^\top = \mathcal{B}_{d-t+1,i} \times_{d-t+1} \mathbf{u}_{d-t+1,i}^\top \in \mathbb{R}^{n_1\times\cdots\times n_{d-t}}$, with each $\mathbf{u}_{j,i}$ normalized:
               $(\mathbf{u}_{1,i},\ldots,\mathbf{u}_{d-t,i}) = \texttt{rank1approx}(\mathcal{A} \times_{d-t+1} \mathbf{u}_{d-t+1,i}^\top \times \cdots \times_d \mathbf{u}_{d,i}^\top)$
20:    end for
21:    Collect $U_j := [\mathbf{u}_{j,1},\ldots,\mathbf{u}_{j,R}]$,  $j = 1,\ldots,d-t$
22: end if                  % end of the computation of non-orthonormal factors
23: return $U_1,\ldots,U_d$
```

2. Since $\mathcal{B}_{j,i} = \mathcal{A} \times_{j+1} \mathbf{u}_{j+1,i}^\top \times_{j+2} \cdots \times_d \mathbf{u}_{d,i}^\top$ and $\mathcal{B}_{j-1,i} = \mathcal{A} \times_j \mathbf{u}_{j,i}^\top \times_{j+1} \cdots \times_d \mathbf{u}_{d,i}^\top$, we have the relation that

$$\mathcal{B}_{j-1,i} = \mathcal{B}_{j,i} \times_j \mathbf{u}_{j,i}^\top.$$

   This recursive definition on $\mathcal{B}_{j,i}$ reduces its computational complexity.
3. Computing $\mathbf{v}_{j,1},\ldots,\mathbf{v}_{j,R}$ can be done in parallel.
4. The algorithm does not guarantee that the generated objective value is larger than certain values, such as $\sum_{i=1}^R \mathcal{A}(i,\ldots,i)^2$ (here $A(i_1,\ldots,i_d)$ denotes the

$(i_1, \ldots, i_d)$-th entry of $\mathcal{A}$, and we assume that $\mathcal{A}(1, \ldots, 1)^2 \geq \mathcal{A}(2, \ldots, 2)^2 \geq \cdots$). This objective function corresponds to the feasible point U_j's with $\mathbf{u}_{j,i} = \mathbf{e}_i$, where $\mathbf{e}_i$ is a vector of the proper size such that its i-th entry is 1 and other entries are 0.

On the procedure `get_v_from_M`: How to obtain $\mathbf{v}_{j,i}$ from $M_{j,i}$ is important both in theory and practice. To retain more information in $\mathbf{v}_{j,i}$ from $M_{j,i}$, we expect that

$$\left\|\mathbf{v}_{j,i}\right\|^2 \geq \beta \left\|M_{j,i}\right\|_F^2, \tag{3.5}$$

where $\beta \in (0, 1)$. We provide three procedures to achieve this, two deterministic and one randomized. To simplify the notations in the procedures, we omit the footscripts j and i. In the procedures, $M \in \mathbb{R}^{n\times m}$, $\mathbf{v} \in \mathbb{R}^n$, and $\mathbf{y} \in \mathbb{R}^m$.

The first idea is straightforward: we let $\mathbf{v}$ be M times its leading right unit singular vector:

```
Procedure v = get_v_from_M(M)                                   (A)

1. Compute the leading right unit singular vector of $M$, denoted as $\mathbf{y}$;
2. Return $\mathbf{v} = M\mathbf{y}$.
```

The second one is to first pick the row of M with the largest magnitude, normalize it to yield a column vector $\mathbf{y}$, and then let $\mathbf{v} = M\mathbf{y}$:

```
Procedure v = get_v_from_M(M)                                   (B)

1. Denote $\mathbf{m}^k$ as the $k$-th row of $M$. Let $\mathbf{m}^{\bar{k}}$ be the row with the largest magnitude, i.e.,
       $\left\|\mathbf{m}^{\bar{k}}\right\| = \max_{1\leq k\leq n} \left\|\mathbf{m}^k\right\|$.
   If there exist multiple $\bar{k}$ satisfying the above condition, choose the first one;
2. Denote $\mathbf{y} := (\mathbf{m}^{\bar{k}})^\top / \left\|\mathbf{m}^{\bar{k}}\right\|$;
3. Return $\mathbf{v} = M\mathbf{y}$.
```

The third one is to first randomly and uniformly pick a row of M, normalize it to yield $\mathbf{y}$, and then let $\mathbf{v} = M\mathbf{y}$:

```
Procedure v = get_v_from_M(M)                                   (C)

1. Randomly and uniformly choose $\bar{k} \in \{1, \ldots, n\}$;
2. Denote $\mathbf{y} := (\mathbf{m}^{\bar{k}})^\top / \left\|\mathbf{m}^{\bar{k}}\right\|$;
3. Return $\mathbf{v} = M\mathbf{y}$.
```

The analysis of the procedures will be given in the next section.

On the procedure `rank1approx`*:* To be more general, `rank1approx` in Algorithm 1 can be any efficient approximation algorithm for solving tensor best rank-1 approximation problems, such as [13, Algorithm 1 with DR 2], [51, Sect. 5], [48, Procedure 3.3], or even Algorithm 1 itself (the case that $R = 1$ and $t = d$). For the purpose of approximation bound analysis, we require `rank1approx` to have an approximation bound characterization, i.e., for any m-th order data tensor $\mathcal{C} \in \mathbb{R}^{n_1\times\cdots\times n_m}$ (assuming that $n_1 \le \cdots \le n_m$), the normalized solution $(\mathbf{x}_1, \ldots, \mathbf{x}_m)$ returned by `rank1approx` admits an approximation bound of the form:

$$\left\langle \mathcal{C}, \bigotimes\nolimits_{j=1}^{m} \mathbf{x}_j \right\rangle \ge \frac{\|\mathcal{C}\|_F}{\zeta(m)}, \ \forall \mathcal{C} \in \mathbb{R}^{n_1\times\cdots\times n_m}, \ m \ge 1, \tag{3.6}$$

where $1/\zeta(m) \le 1$ represents the approximation factor. For He et al. [13, Approximation 1 with DR 2], it can be checked from He et al. [13, Sect. 3.1] that when $m \ge 3$,

$$\zeta(m) = \sqrt{\prod\nolimits_{j=1}^{m-2} n_j} \cdot \sqrt{n_1}.$$

For Yang [48, Procedure 3.3], it follows from Yang [48, Proposition 3.1] that

$$\zeta(m) = \begin{cases} \sqrt{\prod_{j=1}^{m-1} n_j \cdot \prod_{j=1}^{m/2-2} n_{2j+1} \cdot n_2^{-1}} \sqrt{n_{m-1} n_m}, & m \text{ even and } m \ge 4, \\ \sqrt{\prod_{j=2}^{m-1} n_j \cdot \prod_{j=1}^{(m+1)/2-2} n_{2j}} \sqrt{n_{m-1} n_m}, & m \text{ odd and } m \ge 3. \end{cases}$$

When $m = 1, 2$, namely $\mathcal{C}$ is a vector or a matrix, we have that $\zeta(1) = 1$ and $\zeta(2) = \sqrt{n_1}$ for both algorithms.

Computational complexity of Algorithm 1: to simplify the presentation, we consider $n_1 = \cdots = n_d = n$. Computing U_d is a truncated SVD, whose complexity is $O(n^2 \cdot n^{d-1})$. In the computation of U_{d-1}, computing $\mathcal{B}_{d-1,i}$ and $M_{d-1,i}$ requires $O(n^d)$ flops. As $M_{d-1,i} \in \mathbb{R}^{n\times n^{d-2}}$, computing $\mathbf{v}_{d-1,i}$ requires $O(n^2 \cdot n^{d-2})$ flops if Procedure A is used, while it takes $O(n \cdot n^{d-2})$ flops for Procedures B and C. In any case, this is dominated by $O(n^d)$. Thus, the complexity of the splitting step is $O(Rn^d)$. The main effort of the gathering step is the polar decomposition of $V_{d-1,i} \in \mathbb{R}^{n\times R}$, with complexity $O(R^2 n)$. Hence, computing U_{d-1} requires $O(Rn^d) + O(R^2 n)$ flops.

The complexity of computing U_{d-2} is similar: from Remark 3.1, computing $\mathcal{B}_{d-2,i}$ and $M_{d-2,i}$ needs $O(n^{d-1})$ flops. The total complexity of U_{d-2} is $O(Rn^{d-1}) + O(R^2 n)$.

For $U_{d-3}, \ldots, U_{d-t+1}$, the analysis is analogous. Denote $O(\texttt{rank1approx}(\cdot))$ as the complexity of the rank-1 approximation procedure, depending on which procedure is used. Then, the updating of $U_1, \ldots, U_{d-t}$ has complexity

$$O(Rn^{d-t+1}) + O(R \cdot \texttt{rank1approx}(\mathcal{B}_{d-t+1,i} \times_{d-t+1} \mathbf{u}_{d-t+1,i}^{\top})),$$

where the first term comes from computing $\mathcal{B}_{d-t+1,i}$. Summarizing the above discussions, we have:

Proposition 3.1 (Computational complexity of Algorithm 1) *Assume that* $n_1 = n_2 = \cdots = n_d = n$. *The computational complexity of Algorithm 1 is*

$$O(n^{d+1}) + \sum_{j=2}^{t+1} O(Rn^{d-j+2}) + O(tR^2n)$$
$$+O(R \cdot \texttt{rank1approx}(\mathcal{B}_{d-t+1,i} \times_{d-t+1} \mathbf{u}_{d-t+1,i}^\top)).$$

In particular, if $t = d$, *then it is*

$$O(n^{d+1}) + \sum\nolimits_{j=2}^{d+1} O(Rn^{d-j+2}) + O(dR^2n),$$

which is dominated by $O(n^{d+1})$.

Note that Yang [48, Procedure 3.1] requires t SVDs of size $n \times n^{d-1}$, while Algorithm 1 performs only one SVD of this size plus additional operations of smaller sizes. This makes Algorithm 1 more efficient, which will be confirmed by numerical observations.

4 Approximation Bound

Approximation bound results for general tensors with $1 \le t \le d$ are presented in Sect. 4.1, and then, we give the results for nearly orthogonal tensors in Sect. 4.2.

To begin with, we need some preparations. Recall that there is an intermediate variable $\mathbf{y}$ in Procedures A–C. This variable is important in the analysis. To distinguish $\mathbf{y}$ with respect to each i and j, when obtaining $\mathbf{v}_{j,i}$, we denote the associated $\mathbf{y}$ as $\mathbf{y}_{j,i}$, which is of size $\prod_{k=1}^{j-1} n_k$. The procedures show that $\mathbf{v}_{j,i} = M_{j,i}\mathbf{y}_{j,i}$. Since $M_{j,i} = \texttt{reshape}\left(\mathcal{B}_{j,i}, n_j, \prod_{k=1}^{j-1} n_k\right)$ and $\mathcal{B}_{j,i} = \mathcal{A} \times_{j+1} \mathbf{u}_{j+1,i}^\top \times_{j+2} \cdots \times_d \mathbf{u}_{d,i}^\top$, we have the following expression of $\langle \mathbf{u}_{j,i}, \mathbf{v}_{j,i} \rangle$:

$$\begin{aligned}\langle \mathbf{u}_{j,i}, \mathbf{v}_{j,i} \rangle &= \langle \mathbf{u}_{j,i}, M_{j,i}\mathbf{y}_{j,i} \rangle \\ &= \left\langle M_{j,i}^\top \mathbf{u}_{j,i}, \mathbf{y}_{j,i} \right\rangle \\ &= \left\langle \mathcal{A} \times_j \mathbf{u}_{j,i}^\top \times_{j+1} \cdots \times_d \mathbf{u}_{d,i}^\top, \mathcal{Y}_{j,i} \right\rangle, \qquad (4.7)\end{aligned}$$

where we denote

$$\mathcal{Y}_{j,i} := \texttt{reshape}(\mathbf{y}_{j,i}, n_1, \ldots, n_{j-1}) \in \mathbb{R}^{n_1 \times \cdots \times n_{j-1}}. \qquad (4.8)$$

Note that $\|\mathcal{Y}_{j,i}\|_F = 1$ according to Procedures A–C.

The above expression is only well defined when $j \ge 2$ (see Algorithm 1). To make it consistent when $j = 1$ (which happens when $t = d$), we set $\mathcal{Y}_{1,i} = \mathbf{y}_{1,i} = 1$. We also denote $M_{1,i} := \mathcal{B}_{1,i} = \mathbf{v}_{1,i}$ accordingly. It is clear that (4.7) still makes sense when $j = 1$.

On the other hand, V_j in the algorithm is only defined when $j = d-t+1, \ldots, d-1$. For convenience, we denote

$$V_d = [\mathbf{v}_{d,1}, \ldots, \mathbf{v}_{d,R}] \in \mathbb{R}^{n_d \times R} \text{ with } \mathbf{v}_{d,i} := \lambda_i(A_{(d)})\mathbf{u}_{d,i},\ i = 1, \ldots, R. \quad (4.9)$$

We then define $\mathcal{B}_{d-t,i}$. Note that in the algorithm, $\mathcal{B}_{j,i}$ is only defined when $j \geq d-t+1$. We thus similarly define $\mathcal{B}_{d-t,i} := \mathcal{A} \times_{d-t+1} \mathbf{u}_{d-t+1,i}^\top \times_{d-t+2} \cdots \times_d \mathbf{u}_{d,i}^\top \in \mathbb{R}^{n_1 \times \cdots \times n_{d-t}}$, $i = 1, \ldots, R$. When $t = d$, $\mathcal{B}_{0,i}$'s are scalars.

4.1 Approximation Bound for General Tensors

When $t = 1$, Algorithm 1 coincides with Yang [48, Procedure 3.1] if they take the same best rank-1 approximation procedure. Thus, similar to Yang [48, Proposition 3.2], we have:

$$\begin{aligned} G(U_1, \ldots, U_d) &= \sum_{i=1}^R \left\langle \mathcal{A} \times_d \mathbf{u}_{d,i}^\top, \bigotimes_{j=1}^{d-1} \mathbf{u}_{j,i} \right\rangle^2 \geq \sum_{i=1}^R \frac{\left\| \mathcal{A} \times_d \mathbf{u}_{d,i}^\top \right\|_F^2}{\zeta(d-1)^2} \\ &= \sum_{i=1}^R \frac{\lambda_i(A_{(d)})^2}{\zeta(d-1)^2}, \end{aligned} \quad (4.10)$$

where the inequality comes from (3.6), and the last equality is due to that the unfolding of $\mathcal{A} \times_d \mathbf{u}_{d,i}^\top$ to a vector is exactly $\mathbf{v}_{d,i}$ defined in (4.9).

The remaining part is focused on the $t \geq 2$ cases. We first present an overview and informal analysis of the approximation bound, under the setting that (3.4) holds with a factor α_j, and then we detail α_j. The formal approximation bound results are stated in the last of this subsection.

Lemma 4.4 (Informal approximation bound of Algorithm 1) *Let $2 \leq t \leq d$ and let $U_1, \ldots, U_d$ be generated by Algorithm 1. If for each $d-t+1 \leq j \leq d-1$, there holds*

$$\sum\nolimits_{i=1}^R \langle \mathbf{u}_{j,i}, \mathbf{v}_{j,i} \rangle^2 \geq \alpha_j \sum\nolimits_{i=1}^R \langle \mathbf{u}_{j+1,i}, \mathbf{v}_{j+1,i} \rangle^2, \quad (4.11)$$

where $\alpha_j \in (0, 1]$, then we have the following approximation bound:

$$\begin{aligned} G(U_1, \ldots, U_d) &\geq \frac{1}{\zeta(d-t)^2} \prod\nolimits_{j=d-t+1}^{d-1} \alpha_j \cdot \sum\nolimits_{i=1}^R \lambda_i(A_{(d)})^2 \\ &\geq \frac{1}{\zeta(d-t)^2} \prod\nolimits_{j=d-t+1}^{d-1} \alpha_j \cdot G_{\max}, \end{aligned}$$

where $1/\zeta(d-t)$ is the rank-1 approximation ratio of $n_1 \times \cdots \times n_{d-t}$ tensors, as noted in (3.6).

Proof To make the analysis below consistent with the case $t = d$ that does not require perform rank-1 approximation, we denote $\zeta(0) = 1$ and $\bigotimes_{j=1}^{0} \mathbf{u}_{j,i} = 1$. It holds that

$$\begin{aligned} G(U_1, \ldots, U_d) &= \sum_{i=1}^{R} \left\langle \mathcal{A}, \bigotimes_{j=1}^{d} \mathbf{u}_{j,i} \right\rangle^2 \\ &= \sum_{i=1}^{R} \left\langle \mathcal{B}_{d-t,i}, \bigotimes_{j=1}^{d-t} \mathbf{u}_{j,i} \right\rangle^2 \\ &\geq \frac{1}{\zeta(d-t)^2} \sum_{i=1}^{R} \left\| \mathcal{B}_{d-t,i} \right\|_F^2 = \frac{1}{\zeta(d-t)^2} \max_{\|\mathcal{Y}\|_F=1} \left\langle \mathcal{B}_{d-t,i}, \mathcal{Y} \right\rangle \\ &\geq \frac{1}{\zeta(d-t)^2} \sum_{i=1}^{R} \left\langle \mathcal{B}_{d-t,i}, \mathcal{Y}_{d-t+1,i} \right\rangle^2; \end{aligned}$$

here, the first inequality follows from the setting (3.6), and the second one is due to that $\mathcal{Y}_{d-t+1,i} \in \mathbb{R}^{n_1 \times \cdots \times n_{d-t}}$ defined in (4.8) satisfies $\left\| \mathcal{Y}_{d-t+1,i} \right\|_F = 1$. From the definition of $\mathcal{B}_{d-t,i}$ and (4.7), we have

$$\begin{aligned} \left\langle \mathcal{B}_{d-t,i}, \mathcal{Y}_{d-t+1,i} \right\rangle &= \left\langle \mathcal{A} \times_{d-t+1} \mathbf{u}_{d-t+1,i}^{\top} \times_{d-t+2} \cdots \times_d \mathbf{u}_{d,i}^{\top}, \mathcal{Y}_{d-t+1,i} \right\rangle \\ &= \left\langle \mathbf{u}_{d-t+1,i}, \mathbf{v}_{d-t+1,i} \right\rangle. \end{aligned}$$

It thus follows from (4.11) that

$$\begin{aligned} G(U_1, \ldots, U_d) &\geq \frac{1}{\zeta(d-t)^2} \sum_{i=1}^{R} \left\langle \mathbf{u}_{d-t+1,i}, \mathbf{v}_{d-t+1,i} \right\rangle^2 \\ &\geq \cdots \\ &\geq \frac{1}{\zeta(d-1)^2} \prod\nolimits_{j=d-t+1}^{d-t} \alpha_j \cdot \sum\nolimits_{i=1}^{R} \left\langle \mathbf{u}_{d,i}, \mathbf{v}_{d,i} \right\rangle^2. \end{aligned}$$

From (4.9) and that $\left\| \mathbf{u}_{j,i} \right\| = 1$, we see that $\sum_{i=1}^{R} \left\langle \mathbf{u}_{j,i}, \mathbf{v}_{j,i} \right\rangle^2 = \sum_{i=1}^{R} \lambda_i (A_{(d)})^2 \geq G_{\max}$, where Lemma 2.3 gives the inequality. The result follows. □

4.1.1 On Chain Inequality (4.11)

To establish the connection between $\left\langle \mathbf{u}_{j,i}, \mathbf{v}_{j,i} \right\rangle$ and $\left\langle \mathbf{u}_{j+1,i}, \mathbf{v}_{j+1,i} \right\rangle$, we need to detail (3.5). We first simply assume that (3.5) holds with a factor β_j, and later we present β_j in Lemma 4.6. The proofs of Lemmas 4.5 and 4.6 are left to Appendix A.

Lemma 4.5 *Let $2 \leq t \leq d$ and let $U_1, \ldots, U_d$ be generated by Algorithm 1. If for each $d-t+1 \leq j \leq d-1$, there holds*

$$\left\| \mathbf{v}_{j,i} \right\|^2 \geq \beta_j \left\| M_{j,i} \right\|_F^2, \ i = 1, \ldots, R, \tag{4.12}$$

where $\beta_j \in (0, 1]$ (if $j = 1$ then (4.12) holds with $\beta_j = 1$), then we have for $j = d - t + 1, \ldots, d - 1$, and $i = 1, \ldots, R$,

$$\sum_{i=1}^{R} \langle \mathbf{u}_{j,i}, \mathbf{v}_{j,i} \rangle^2 \geq \frac{\beta_j}{R} \sum_{i=1}^{R} \langle \mathbf{u}_{j+1,i}, \mathbf{v}_{j+1,i} \rangle^2 .$$

Remark 4.2 When $j = 1$ (this happens when $t = d$), according to Algorithm 1, $\mathbf{v}_{1,i} = \mathcal{B}_{1,i} = M_{1,i}$ for each i, and so (4.12) holds with $\beta_1 = 1$.

We then specify (4.12). Denote E as the expectation operator.

Lemma 4.6 *For $2 \leq j \leq d - 1$, if $\mathbf{v}_{j,i}$ is generated from $M_{j,i}$ by Procedures A or B, then it holds that*

$$\|\mathbf{v}_{j,i}\|^2 \geq \frac{1}{n_j} \|M_{j,i}\|_F^2 .$$

If $\mathbf{v}_{j,i}$ is generated by Procedure C, then it holds that

$$\mathsf{E}\|\mathbf{v}_{j,i}\|^2 \geq \frac{1}{n_j} \|M_{j,i}\|_F^2 .$$

The above two lemmas show that (4.12) is crucial in deriving the approximation bound. (4.12) also implies that it is possible to devise `get_v_from_M` without using the form $M_{j,i}\mathbf{y}_{j,i}$, which can be studied in the future.

4.1.2 Putting the Pieces Together

Let β_j be such that

$$\beta_j = n_j^{-1} \text{ if } 2 \leq j \leq d - 1, \quad \text{and} \quad \beta_j = 1 \text{ if } j = 1.$$

Based on (4.10) and Lemmas 4.4–4.6, the main results are stated as follows.

Theorem 4.2 (Formal approximation bound) *Let $1 \leq t \leq d$ and let $U_1, \ldots, U_d$ be generated by Algorithm 1. For $j = d - t + 1, \ldots, d - 1$, $i = 1, \ldots, R$, if $\mathbf{v}_{j,i}$'s are generated deterministically by Procedures A or B, then*

$$\begin{aligned} G(U_1, \ldots, U_d) &\geq \frac{1}{\zeta(d-t)^2} \prod\nolimits_{j=d-t+1}^{d-1} \frac{\beta_j}{R} \cdot \sum\nolimits_{i=1}^{R} \lambda_i(A_{(d)})^2 \\ &\geq \frac{1}{\zeta(d-t)^2} \prod\nolimits_{j=d-t+1}^{d-1} \frac{\beta_j}{R} \cdot G_{\max}; \end{aligned}$$

if $\mathbf{v}_{j,i}$*'s are generated randomly by Procedure C, then*

$$\begin{aligned} EG(U_1,\ldots,U_d) &\geq \frac{1}{\zeta(d-t)^2}\prod\nolimits_{j=d-t+1}^{d-1}\frac{\beta_j}{R}\cdot\sum\nolimits_{i=1}^{R}\lambda_i(A_{(d)})^2 \\ &\geq \frac{1}{\zeta(d-t)^2}\prod\nolimits_{j=d-t+1}^{d-1}\frac{\beta_j}{R}\cdot G_{\max}. \end{aligned}$$

The ratio $\frac{1}{\zeta(d-t)^2}\prod_{j=d-t+1}^{d-1}\frac{\beta_j}{R}$ *is*

$$\begin{cases} \frac{1}{R^{d-1}\prod_{j=2}^{d-1}n_j}, & t=d, \\ \frac{1}{R^{t-1}\zeta(d-t)^2\prod_{j=d-t+1}^{d-1}n_j}, & 1\leq t<d. \end{cases}$$

Discussions on the approximation bound are left to Appendix B. In particular, the empirical study indicates that the approximation bound might be independent of R. However, how to improve it still needs further study.

Reducing to the best rank-1 approximation problem $\max_{\|\mathbf{u}_j\|=1,1\leq j\leq d}\left\langle \mathcal{A},\bigotimes_{j=1}^d\mathbf{u}_j\right\rangle$ i.e., $R=1, t=d$, from Theorem 4.2 we have the following corollary.

Corollary 4.1 *Let* $\mathbf{u}_j$, $1\leq j\leq d$ *be generated by Algorithm 1 with* $R=1$ *and* $t=d$. *Then, the approximation ratio is* $\frac{1}{\sqrt{\prod_{j=2}^{d-1}n_j}}$ *(either in the deterministic or the expected sense).*

The above ratio is of the same order as [13, 51], which is not the best to date [12, 40] (it lacks a factor $\sqrt{\prod_{j=2}^{d-1}\log n_j}$). However, our aim is not to compare them in the context of rank-1 approximation. Moreover, even in this case, Algorithm 1 with Procedures B and C is new (in fact, after this work is almost finished, we find that in the rank-1 setting, Algorithm 1 with Procedure A is essentially the same as [13, Algorithm 1 with DR 2].).

4.2 Approximation Results for Nearly Orthogonal Tensors

We consider the following type of tensors in this subsection:

Assumption 4.1

$$\mathcal{A}=\sum\nolimits_{i=1}^{R}\sigma_i\bigotimes\nolimits_{j=1}^{d}\mathbf{u}^*_{j,i}+\mathcal{E}, \tag{4.13}$$

where the last t U_j^*'s are orthonormal, and the first $(d-t)$ U_j^*'s are columnwisely normalized. We assume that $\sigma_1>\cdots>\sigma_R>0$. $\mathcal{E}$ denotes the noisy tensor.

The cases that $\sigma_{i_1}=\sigma_{i_2}$ for some $1\leq i_1<i_2\leq R$ need more assumptions and analysis, which is out of the current scope.

Running Algorithm 1 on $\mathcal{A}$ with $t\geq 2$, it is easy to obtain the following results.

Proposition 4.2 *Let $\mathcal{A}$ satisfy Assumption 4.1 with $t \geq 2$ and $\mathcal{E} = 0$. Let $U_1, \ldots, U_d$ be generated by Algorithm 1, where t in the algorithm takes the same value as that in the assumption. Then,*

$$G(U_1, \ldots, U_d) = \sum\nolimits_{i=1}^{R} \sigma_i^2 = G_{\max}, \quad \text{and} \quad U_j = U_j^*, \; j = 1, \ldots, d,$$

where for $U_j = U_j^*$, we ignore the sign, i.e., it in fact holds $\mathbf{u}_{j,i} = \pm\mathbf{u}_{j,i}^*$ for each j and i.

Remark 4.3 This proposition also shows that Algorithm 1 is well designed for orthogonal tensors.

The proof is left to Appendix A. From its proof and by matrix perturbation theory, it is not hard to obtain the perturbed version of Proposition 4.2:

Proposition 4.3 *Under the setting of Proposition 4.2, if $\mathcal{E} \neq 0$ is small enough, then we have $G(U_1, \ldots, U_d) = G_{\max} - O(\|\mathcal{E}\|)$, where the constant behind the big O is nonnegative, and*

$$\min\{\left\|\mathbf{u}_{j,i} + \mathbf{u}_{j,i}^*\right\|, \left\|\mathbf{u}_{j,i} - \mathbf{u}_{j,i}^*\right\|\} = O(\|\mathcal{E}\|), \quad j = 1, \ldots, d, \quad i = 1, \ldots, R.$$

It remains to consider the case that $t = 1$ in Assumption 4.1, i.e., only U_d^* is orthonormal. In general, if there are no additional assumptions on $U_j^*, j = 1, \ldots, d-1$, then it is hard to obtain the approximation results. For instance, if $A = PQ^\top$ where P is orthonormal but Q is not, then it is hard to recover P and Q. We thus make the following incoherence assumption:

Assumption 4.2 There exists at least one U_j^*, $1 \leq j \leq d-1$ that is incoherent with modules $0 \leq \delta < 1$, i.e.,

$$\exists 1 \leq j \leq d-1, \text{ with } \left|\left\langle \mathbf{u}_{j,i_1}^*, \mathbf{u}_{j,i_2}^* \right\rangle\right| \leq \delta, \; \forall i_1 \neq i_2.$$

It is clear that U_j^* is a nearly orthonormal matrix if δ is small enough. In what follows, we assume without loss of generality that U_{d-1}^* satisfies Assumption 4.2. We immediately have:

Proposition 4.4 *If U_{d-1}^* is incoherent with modules δ, then U_{-d}^* is also incoherent with modules δ.*

We consider the expression of $A_{(d)}A_{(d)}^\top$. Let $\Sigma := \mathrm{diag}(\sigma_1, \ldots, \sigma_R)$ and $U_{-d}^* \in \mathbb{R}^{\prod_{j=1}^{d-1} n_j \times R}$ be the matrix whose i-th column is the vectorization of $\bigotimes_{j=1}^{d-1} \mathbf{u}_{j,i}^*$. We have

$$\begin{aligned}
A_{(d)}A_{(d)}^\top &= U_d^*\Sigma(U_{-d}^*)^\top U_{-d}^*\Sigma(U_d^*)^\top + O(\mathcal{E})\\
&= U_d^*\Sigma^2(U_d^*)^\top + U_d^*\Sigma\\
&\times \begin{bmatrix} 0 & (\mathbf{u}_{-d,1}^*)^\top\mathbf{u}_{-d,2}^* & \cdots & (\mathbf{u}_{-d,1}^*)^\top\mathbf{u}_{-d,R}^* \\ \vdots & 0 & \ddots & \vdots \\ \vdots & \ddots & 0 & \vdots \\ (\mathbf{u}_{-d,R}^*)^\top\mathbf{u}_{-d,1}^* & \cdots & (\mathbf{u}_{-d,R}^*)^\top\mathbf{u}_{-d,R-1}^* & 0 \end{bmatrix}\Sigma(U_d^*)^\top\\
&+ O(\mathcal{E})\\
&= U_d^*\Sigma^2(U_d^*)^\top + O(\delta) + O(\mathcal{E}),
\end{aligned}$$

namely, $A_{(d)}A_{(d)}^\top$ is a perturbation of the eigen-decomposition $U_d^*\Sigma^2(U_d^*)^\top$, given that δ and $\mathcal{E}$ are small enough. If Assumption 4.1 holds, by matrix perturbation theory, the above discussion implies that

$$\min\{\|\mathbf{u}_{d,i}+\mathbf{u}_{d,i}^*\|,\|\mathbf{u}_{d,i}-\mathbf{u}_{d,i}^*\|\} = O(\delta)+O(\|\mathcal{E}\|),\ i=1,\ldots,R,$$

where U_d is generated by the algorithm. It then follows that $\mathcal{A}\times_d \mathbf{u}_{d,i}^\top$ is a nearly rank-1 tensor with perturbation $O(\delta)+O(\mathcal{E})$, and so

$$\min\{\left\|\mathbf{u}_{j,i}+\mathbf{u}_{j,i}^*\right\|,\left\|\mathbf{u}_{j,i}-\mathbf{u}_{j,i}^*\right\|\} = O(\delta)+O(\|\mathcal{E}\|),\ j=1,\ldots,d-1.$$

We thus have a similar result as Proposition 4.3:

Proposition 4.5 *Denote $U_1,\ldots,U_d$ as the factors generated by Algorithm 1, where Assumption 1 holds with $t=1$; furthermore, Assumption 4.2 holds. If $\mathcal{E}$ and δ are small enough, and we also set $t=1$ in the algorithm, then $G(U_1,\ldots,U_d)=G_{\max}-O(\delta)-O(\|\mathcal{E}\|)$, where the constants behind the big O are nonnegative, and*

$$\begin{aligned}&\min\left\{\left\|\mathbf{u}_{j,i}+\mathbf{u}_{j,i}^*\right\|,\left\|\mathbf{u}_{j,i}-\mathbf{u}_{j,i}^*\right\|\right\}\\ &= O(\delta)+O(\|\mathcal{E}\|),\quad j=1,\ldots,d,\quad i=1,\ldots,R.\end{aligned}$$

5 Numerical Study

We evaluate the performance of Algorithm 1 with Procedures A–C, and compare them with Yang [48, Procedure 3.1]. All the computations are conducted on an Intel i7-7770 CPU desktop computer with 32 GB of RAM. The supporting software is MATLAB R2019b. The MATLAB package Tensorlab [46] is employed for tensor operations. The MATLAB code is available in the folder "alg" of https://github.com/yuningyang19/epsilon-ALS with the name "approx_alg_new".

Performance of approximation algorithms: we compare Algorithm 1 with Procedures A–C with [48, Procedure 3.1] in this part. They are, respectively, marked as Algorithm 1 (A), Algorithm 1 (B), Algorithm 1 (C), and Yang2020. All the algorithms employ the same `rank1approx` procedure [48, Procedure 3.2] for solving the rank-1 approximation subproblems. The tensors $\mathcal{A} \in \mathbb{R}^{n\times n\times n\times n}$ are randomly generated where every entry obeys the standard normal distribution. The presented results are averaged over 50 instances for each case.

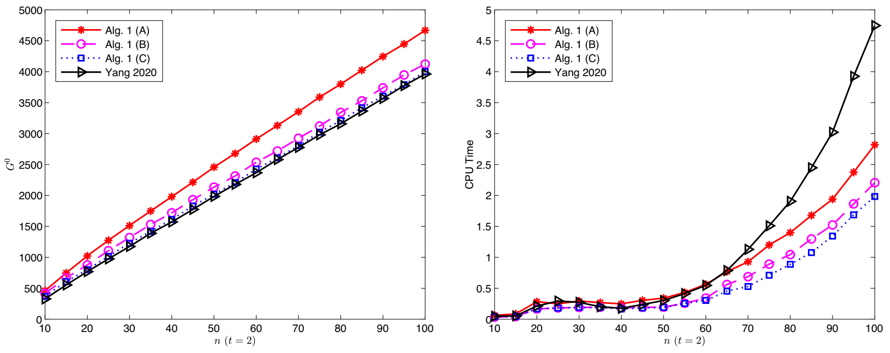
We first fix $R = 10$, and let n vary from 10 to 100. The results are depicted in Fig. 2. Algorithm 1 (A) is in red with star markers, Algorithm 1 (B) is in magenta with circle markers, Algorithm 1 (C) is in blue with square markers, and Yang2020 is in black with right arrow markers. The left panels show the curves of the objective values $G(U_1,\ldots,U_d)$ versus n, from which we observe that Algorithm 1 (A) performs the best, followed by Algorithm 1 (B). This is because by using SVDs, Procedure A retains more information in $\mathbf{v}$ than other procedures. Yang2020 performs not as well as Algorithm 1, and its performance gets worse when t increases. This also has been reported in [48, Table 3.1], while the performance of Algorithm 1 is more stable. Therefore, we may conclude that Algorithm 1 exploits the structure of the problem better than Yang2020. The right panels show the CPU time versus n, from which we see that Yang2020 needs more time than Algorithm 1. This is because Yang2020 requires to perform t SVDs of size $n \times n^{d-1}$, while the most expensive step of Algorithm 1 is only one SVD of this size. Considering Procedures A–C themselves, Algorithm 1 (A) is the most expensive one, followed by Algorithm 1 (B). The randomized version, i.e., Algorithm 1 (C), is the cheapest one.

We then fix $n = 80$, $t = 3$, and vary R from 5 to 80. The results are plotted in Fig. 3, from which we still observe that Algorithm 1 (A) performs the best in terms of the objective value. Considering the CPU time, Algorithm 1 (B) and 1 (C) seem to be far better. This shows the superiority of SVD-free procedures for computing the vector $\mathbf{v}$ concerning the CPU time.

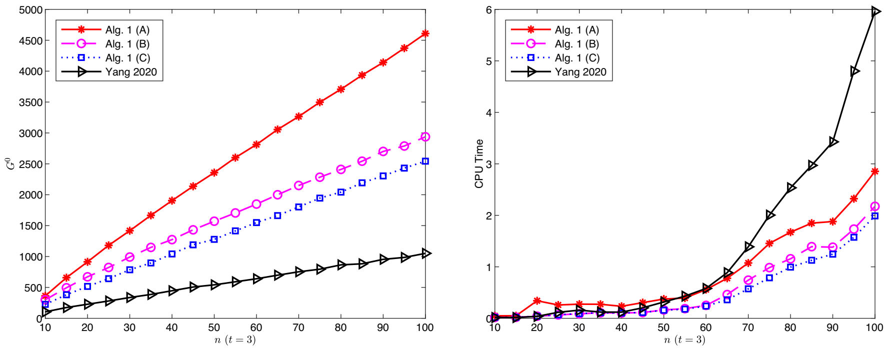
Performance of approximation algorithms plus ALS for factor recovery: the tensors are generated similarly as [41, 48]: $\mathcal{A} = \mathcal{B}/\|\mathcal{B}\| + \beta\cdot\mathcal{N}/\|\mathcal{N}\|$, where $\mathcal{B} = \sum_{i=1}^R \sigma_i \bigotimes_{j=1}^d \mathbf{u}^*_{j,i}$, $\mathcal{N}$ is an unstructured tensor, and β denotes the noise level. Here we set $\beta = 0.1$, $R = 10$, $d = 4$, and n varies from 60 to 90. σ_i, U^*_j, and $\mathcal{N}$ are randomly drawn from a uniform distribution in $[-1, 1]$. The last t U^*_j are then orthonormalized by using the `orth` function, while the first $(d-t)$ ones are columnwisely normalized. We only test $t = 3$ and $t = 4$ cases. We compare ϵ-ALS [48] with $\epsilon = 10^{-8}$ initialized by different initializers generated respectively by Yang2020, Algorithm 1 (A), Algorithm 1 (B), and Algorithm 1 (C). ϵ-ALS initialized by random initializers is used as a baseline. The stopping criterion for ϵ-ALS is $\sum_{j=1}^d \left\|U_j^{(k+1)} - U_j^{(k)}\right\|_F / \left\|U_j^{(k)}\right\|_F \leq 10^{-5}$ or $k \geq 2000$. The relative error is defined as follows [41, 48]:

$$\text{rel.err} = \sum_{j=1}^d \left\|U^*_j - U^{\text{out}}_j \cdot \Pi_j\right\|_F / \left\|U_j\right\|_F,$$

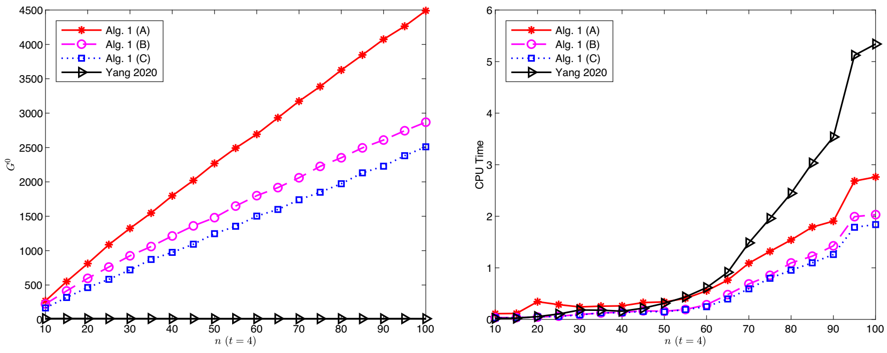
where $\Pi_j = \arg\min_{\Pi\in\boldsymbol{\Pi}} \left\|U^*_j - U^{\text{out}}_j \cdot \Pi\right\|_F$, $\boldsymbol{\Pi}$ denotes the set of permutation matrices, and U^{out}_j's are the factors output by the iterative algorithm. Finding the permutation can be efficiently done by the Hungarian algorithm [22]. The presented results are averaged over 50 instances for each case.



(a) $t = 2$. Left: $G(U_1, \ldots, U_d)$ versus n; right: CPU time versus n.



(b) $t = 3$. Left: $G(U_1, \ldots, U_d)$ versus n; right: CPU time versus n.



(c) $t = 4$. Left: $G(U_1, \ldots, U_d)$ versus n; right: CPU time versus n.

Fig. 2 Performance of Algorithm 1 (A) (red and star markers), Algorithm 1 (B) (magenta and circle markers), Algorithm 1 (C) (blue and square markers), and [48, Procedure 3.1] (black and right arrow markers) for approximately solving (2.2) where $\mathcal{A} \in \mathbb{R}^{n\times n\times n\times n}$ are randomly generated. n varies from 10 to 100

The results when $t = 3$ and $t = 4$ are reported in Tables 1 and 2, in which 'rel.err^0' and 'time^0, respectively, represent the relative error and CPU time evaluated at the initializers. From the tables, we first observe that ϵ-ALS initialized by Yang2020 and Algorithm 1 outperforms those with random initializations, considering both relative

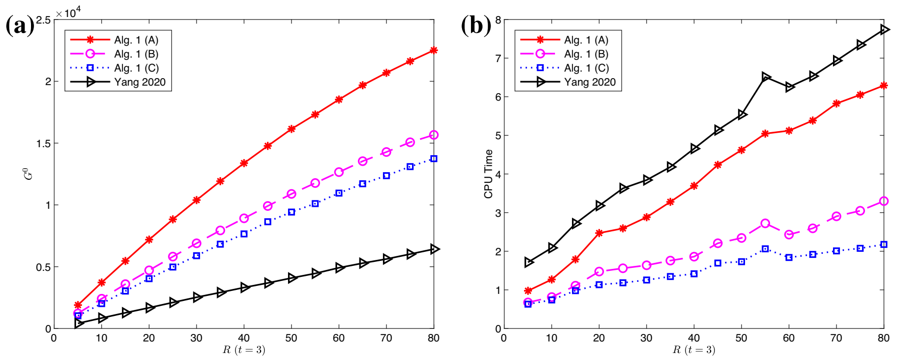


Fig. 3 Performances of Algorithm 1 (A) (red and star markers), Algorithm 1 (B) (magenta and circle markers), Algorithm 1 (C) (blue and square markers), and [48, Procedure 3.1] (black and right arrow markers) for approximately solving (2.2) where $\mathcal{A} \in \mathbb{R}^{80\times 80\times 80\times 80}$ are randomly generated. R varies from 5 to 80

error and computational time. Comparing Algorithm 1 with Yang2020, we can see that in most cases, ϵ-ALS initialized by Algorithm 1 performs comparable or slightly better. Comparing among ϵ-ALS initialized by the three versions of Algorithm 1, Algorithm 1 (C) is slightly worse in terms of time; this is because although Algorithm 1 (C) is the most efficient, ϵ-ALS initialized by it needs more iterations than the others. Therefore, it would be necessary to further improve the efficiency of the iterative algorithms.

Next, we fix $d = t = 4$, $n = 100$, $\beta = 0.1$, and vary R from 10 to 100. We also fix $n = 50$, $R = 10$, and vary β from 0.05 to 1. Plots of relative error of different methods are depicted in Fig. 4. From the table, we still observe that ϵ-ALS initialized by approximation algorithms performs better than that with random initializers, and the iterative algorithm initialized by Algorithm 1 is comparable with that initialized by Yang2020.

CP approximation for clustering: tensor CP approximation for clustering works as follows: Suppose that we have N samples $\mathcal{A}_1, \ldots, \mathcal{A}_n \in \mathbb{R}^{n_1\times\cdots\times n_d}$, $d \geq 2$. For a given parameter R, and for unknown variables $A \in \mathbb{R}^{N\times R}$, $U_j \in \mathbb{R}^{n_j\times R}$, $j = 1, \ldots, d$, one solves the following problem first:

$$\min_{A,U_1,\ldots,U_d} \sum_{k=1}^{N} \left\| \mathcal{A}_k - \sum\nolimits_{i=1}^{R} a_i^k \bigotimes\nolimits_{j=1}^{d} \mathbf{u}_{j,i} \right\|_F^2 \quad \text{s.t. } A = (a_i^k) \in \mathbb{R}^{N\times R}, U_j^\top U_j = I, 1 \leq j \leq d, \tag{5.14}$$

where in $a_i^k \in \mathbb{R}$, k represents the row index while i is the column index. Write $\mathbf{a}^k := [a_1^k, \ldots, a_R^k]^\top \in \mathbb{R}^R$, i.e., it is the transpose of the k-th row of A. Equation (5.14) means that one finds a common subspace in $\mathbb{R}^{n_1\times\cdots\times n_d}$, which is spanned by the basis $\bigotimes_{j=1}^{d} \mathbf{u}_{j,i}$, $i = 1, \ldots, R$, and projects the samples onto this subspace. $\mathbf{a}^k$ can be seen as a representation of $\mathcal{A}_k$. This can be regarded as a dimension reduction procedure [32, 35]. By stacking $\mathcal{A}_k$'s into a $(d+1)$-th order tensor $\mathcal{A}$ with $\mathcal{A}(k, :, \ldots, :) = \mathcal{A}_k$, and denoting $\mathbf{a}_i$ as the i-th column of A, the objective function of (5.14) can be

Table 1 ϵ-ALS initialized by different strategies

n		Yang2020	Algorithm 1 (A)	Algorithm 1 (B)	Algorithm 1 (C)	Random
60	Rel.err. (rel.err^0)	0.0139 (0.0366)	0.0140 (0.0295)	**0.0137** (0.0348)	0.0218 (0.0680)	0.0291
	Iter.	7.32	6.26	8.70	8.04	18.72
	Time (time^0)	0.5864 (0.1670)	**0.5155** (0.1527)	0.5645 (0.0931)	0.5340 (0.0810)	0.92
70	Rel.err. (rel.err^0)	**0.0125** (0.0255)	**0.0125** (0.0177)	0.0190 (0.0252)	0.0197 (0.0489)	0.0285
	Iter.	4.50	4.34	4.38	4.68	32.92
	Time (time^0)	0.7188 (0.3631)	0.6209 (0.2644)	**0.5205** (0.1646)	0.5440 (0.1558)	2.7100
80	Rel.err. (rel.err^0)	0.0191 (0.0239)	0.0186 (0.0216)	0.0189 (0.0218)	**0.0114** (0.0343)	0.0193
	Iter.	19.20	9.24	9.72	24.44	35.08
	Time (time^0)	3.1558 (0.6198)	1.7679 (0.3633)	**1.7281** (0.2647)	3.5075 (0.2460)	4.5828
90	Rel.err. (rel.err^0)	0.0030 (0.0096)	0.0030 (0.0057)	0.0030 (0.0109)	0.0030 (0.0314)	0.0108
	Iter.	5.52	4.92	7.56	10.38	32.10
	Time (time^0)	2.4842 (1.0172)	**1.8817** (0.5410)	2.2418 (0.3979)	2.7513 (0.3620)	6.4116

Bold values indicate that they are the best among the compared ones

$t = 3$ and $R = 10$ cases. 'rel.err^0' and 'time^0', respectively, represent the relative error and CPU time evaluated at the initializers

Table 2 ϵ-ALS initialized by different strategies

		Yang2020	Algorithm 1 (A)	Algorithm 1 B	Algorithm 1 (C)	Random
60	Rel.err. (rel.err^0)	0.0211 (0.0328)	0.0214 (0.0263)	**0.0133** (0.0320)	0.0209 (0.0658)	0.0706
	Iter.	13.42	12.04	11.32	18.56	36.68
	Time (time^0)	0.8780 (0.1883)	0.7927 (0.1580)	**0.6884** (0.0900)	1.0134 (0.0835)	1.7535
70	Rel.err. (rel.err^0)	0.0129 (0.0300)	0.0129 (0.0209)	**0.0126** (0.0318)	0.0132 (0.0560)	0.0369
	Iter.	31.62	23.34	18.58	28.94	45.86
	Time(time^0)	3.1625 (0.3609)	2.2063 (0.2252)	**1.7603** (0.1550)	2.5664 (0.1429)	4.0330
80	Rel.err. (rel.err^0)	0.0279 (0.0347)	**0.0276** (0.0294)	0.0279 (0.0312)	0.0280 (0.0477)	0.0361
	Iter.	16.84	9.58	13.28	18.52	24.76
	Time (time^0)	2.9619 (0.6234)	**1.8031** (0.3555)	2.1936 (0.2619)	2.7764 (0.2482)	3.3043
90	Rel.err. (rel.err^0)	0.0029 (0.0064)	0.0029 (0.0041)	0.0029 (0.0079)	0.0029 (0.0414)	0.0194
	Iter.	30.56	11.48	20.06	16.88	43.46
	Time (time^0)	6.9453 (0.9814)	**3.0044** (0.5021)	4.4624 (0.3835)	3.8632 (0.3499)	8.3136

Bold values indicate that they are the best among the compared ones
$t = 4$ and $R = 10$ cases. ‘rel.err^0’ and ‘time^0’, respectively, represent the relative error and CPU time evaluated at the initializers

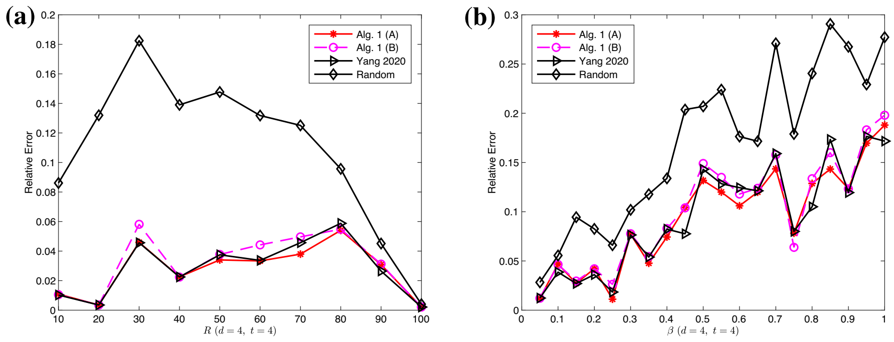


Fig. 4 Comparisons of relative error of ϵ-ALS initialized by Algorithm 1 (A) (red and star markers), Algorithm 1 (B) (magenta and circle markers), [48, Procedure 3.1] (black and right arrow markers), and random initializers (black and diamond markers). Left panel: n is fixed to 100, $\beta = 0.1$, $t = 4$, while R varies from 10 to 100. Right panel: n is fixed to 50, $t = 4$, $R = 10$, while the noise level β varies from 0.05 to 1

written as $\left\| \mathcal{A} - \sum_{i=1}^{R} \mathbf{a}_i \otimes \mathbf{u}_{1,i} \otimes \cdots \otimes \mathbf{u}_{d,i} \right\|_F^2$, and so (5.14) reduces to (2.2). Once $A = [\mathbf{a}_1, \ldots, \mathbf{a}_R]$ is obtained, we use K-means for clustering, with the transpose of the rows of A, namely, $\mathbf{a}^k$'s being the new samples. The cluster error is defined as follows: Let $K \geq 2$ be the given clusters, $\phi_0 : \mathbb{R}^{n_1 \times \cdots \times n_d} \to \{1, \ldots, K\}$ be the true clustering mapping, and ϕ be the estimated one. Denote card$(\cdot)$ the cardinality of a set and $I(\cdot)$ the indicator function. The following metric is used to measure clustering performance [43]:

$$\text{cluster err.} := \binom{N}{2}^{-1} \text{card}\left(\{i, j\} : I\left(\psi(\mathcal{A}_i) = \psi(\mathcal{A}_j)\right) \neq I\left(\psi_0(\mathcal{A}_i) = \psi_0(\mathcal{A}_j)\right), i < j\right).$$

We solve (5.14) by ϵ-ALS initialized by Algorithm 1 (A), (B), and by random initialization, with $R = 30$ for the problem. For the first two initializations, the max iterations of ϵ-ALS are set to 10, while it is 1000 for the random initialization. We also compare them with vanilla K-means, namely the original samples $\mathcal{A}_k$'s are first vectorized and then clustered by K-means. The dataset COIL-100 http://www.cs.columbia.edu/CAVE/software/softlib/coil-100.php is used, which consists of 100 objects, each containing 72 images of size 128×128 viewing from different angles. In our experiment, we each time randomly select $K = \{5, 7, 9, 11, 15, 20\}$ objects, each object randomly selecting $M = 50$ images, resulting into a third-order tensor $\mathcal{A} \in \mathbb{R}^{50K \times 128 \times 128}$ that consists of $50K$ samples. For each case we run 50 instances and present the averaged results in Table 3.

Considering the clustering error, we observe from the table that armed with the approximation algorithm, CP approximation-based method achieves better performance than the vanilla K-means, while Algorithm 1 (A) is slightly better than Algorithm 1 (B). However, if starting from a random initializer, CP approximation based method is worse than vanilla K-means. This shows the usefulness of the introduced approximation algorithm. Considering the computational time, we see that the first two methods are also better than vanilla K-means. This is because ϵ-ALS is stopped within 10 iterations and because the sample size after reduction is $R = 30$. On the other side, the sample size for the vanilla K-means is 128^2. We also observe

Table 3 CP approximation for clustering via first solving (5.14) by Algorithm 1 (A) + ϵ-ALS, Algorithm 1 (B) + ϵ-ALS, and random + ϵ-ALS

K	Algorithm 1 (A)		Algorithm 1 (B)		Random		vanilla K-means	
	Cluster err.	Time	Cluster err.	Time	Cluster err.	Time.	Cluster err.	Time
5	**1.10E−01**	0.33	1.12E−01	0.24	1.44E−01	6.25	1.33E−01	0.44
7	**9.60E−02**	0.40	1.04E−01	0.31	1.64E−01	5.12	1.08E−01	0.79
9	**8.24E−02**	0.50	9.12E−02	0.42	1.40E−01	11.74	1.00E−01	1.25
11	**7.48E−02**	0.58	8.08E−02	0.50	1.21E−01	15.26	8.54E−02	1.76
15	**6.31E−02**	0.78	6.92E−02	0.68	9.56E−02	14.86	7.25E−02	3.23
20	5.36E−02	1.04	**5.23E−02**	0.98	7.84E−02	24.14	5.56E−02	5.58

Bold values indicate that they are the best among the compared ones

The iterative algorithm of the first two stops within 10 iterations while that for the third is 1000; then, the K-means is performed to the reduced samples $\mathbf{a}^k$'s. We also use the vanilla K-means as the baseline

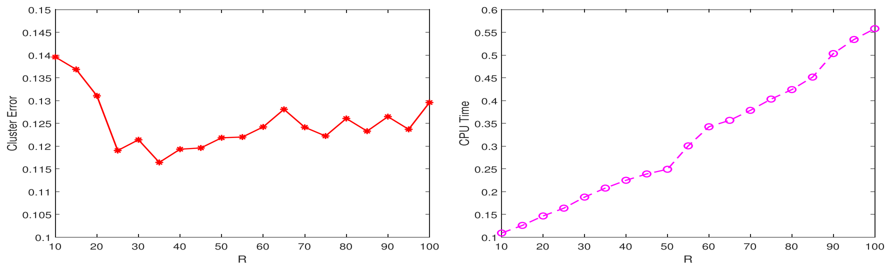


Fig. 5 Cluster error and CPU time of CP approximation for clustering of one instance with varying R from 10 to 100, and $K = 5$. Left: cluster error versus R; right: CPU time versus R

that Random + ϵ-ALS usually cannot stop within 1000 iterations, making it the slowest one.

We next show the influence of R on the performance. We fix an instance with $K = 5$, and vary R from 10 to 100. We plot the cluster error and CPU time of Algorithm 1 (A) + ϵ-ALS in Fig. 5. We see that the cluster error does not change a lot when R varies, while around $R = 30$, the cluster error seems to be slightly better than the other cases. This together with the CPU time shown in the right panel explains why we choose $R = 30$ in the experiment.

6 Conclusions

In [48], an approximation procedure was proposed for orthogonal low-rank tensor approximation, while the approximation bound was only established when the number of latent orthonormal factors is one. To fill this gap, a modified approximation algorithm was developed in this work. It allows either deterministic or randomized procedures to solve some key subproblems in the algorithm, therefore giving more flexibility. The approximation bound, both in the deterministic and the expected senses, has been established regardless of how many orthonormal factors there are. Moreover, compared with Yang [48, Procedure 3.1] which requires t SVDs of size $n \times n^{d-1}$, the modified algorithm involves only one SVD of this size (plus other operations of smaller size), making it more efficient. Numerical tests were provided to verify the usefulness of the algorithm, and its performance is favorably compared with Yang [48, Procedure 3.1]. One of the future work is that in Algorithm 1, instead of obtaining $\mathcal{B}_{j,i}$ from $\mathcal{A} \times_{j+1} \mathbf{u}_{j+1,i}^\top \cdots \times_d \mathbf{u}_{d,i}^\top$, other choices of mixing $\mathcal{A}$ and $\mathbf{u}_{j,i}$ are possible. On the other hand, approaches for finding global solutions can be studied, where a possible way is to use convex relaxation as those for rank-1 approximation [17, 30, 49]. Another possible research thread is to extend the notion of best rank-1 approximation ratio of a tensor space [25, 33] to the best rank-R approximation ratio setting and study its properties.

Acknowledgements We thank the editor and the anonymous reviewers for their insightful comments and suggestions that helped improve this manuscript. We thank Mr. Xianpeng Mao for his help in the experiments. This work was supported by National Natural Science Foundation of China Grants 11801100 and 12171105, and the Fok Ying Tong Education Foundation Grant 171094.

A Proofs

Proof of Lemma 2.3 One notices that

$$\sum_{i=1}^R \lambda_i(A_{(d)})^2 = \max_{X_i} \sum_{i=1}^R \left\langle A_{(d)}, X_i \right\rangle^2 \text{ s.t. } \|X_i\|_F = 1, \operatorname{rank}(X_i) = 1,$$
$$\left\langle X_i, X_j \right\rangle = 0, i \neq j,$$

where $X_i \in \mathbb{R}^{n_d \times \prod_{k=1}^{d-1} n_k}$. The above problem is equivalent to

$$\max_{U,V} \sum_{i=1}^R \left(\mathbf{u}_i^\top A_{(d)} \mathbf{v}_i\right)^2 \text{ s.t. } U^\top U = I, \|\mathbf{v}_i\| = 1,$$

where $U = [\mathbf{u}_1, \ldots, \mathbf{u}_R] \in \mathbb{R}^{n_d \times R}$ and $V = [\mathbf{v}_1, \ldots, \mathbf{v}_R] \in \mathbb{R}^{\prod_{k=1}^{d-1} n_k \times R}$. Note that the above maximization problem is a relaxation of (2.2); thus $\sum_{i=1}^R \lambda_i(A_{(d)})^2 \geq G_{\max}$. □

Proof of Lemma 4.5 Since the orthonormal U_j's, $j = d-t+1, \ldots, d-1$, are given by the polar decomposition of V_j, from Lemma 2.1, we have $\left\langle \mathbf{u}_{j,i}, \mathbf{v}_{j,i} \right\rangle \geq 0$. Using the relation that $a_1^2 + \cdots + a_R^2 \geq \frac{1}{R}(a_1 + \cdots + a_R)^2$ for nonnegative a_i's, we have

$$\sum_{i=1}^R \left\langle \mathbf{u}_{j,i}, \mathbf{v}_{j,i} \right\rangle^2 \geq \frac{1}{R}\left(\sum_{i=1}^R \left\langle \mathbf{u}_{j,i}, \mathbf{v}_{j,i} \right\rangle\right)^2 = \frac{1}{R}\left\langle U_j, V_j \right\rangle^2$$

for each j. Lemma 2.2 shows that $\left\langle U_j, V_j \right\rangle = \left\| V_j \right\|_*$; the norm relationship gives that $\left\| V_j \right\|_* \geq \left\| V_j \right\|_F$. Thus,

$$\sum_{i=1}^R \left\langle \mathbf{u}_{j,i}, \mathbf{v}_{j,i} \right\rangle^2 \geq \frac{1}{R}\left\| V_j \right\|_F^2 .$$

We distinguish the cases that $j \leq d-2$ and $j = d-1$. For the former case,

$$\begin{aligned}
\sum\nolimits_{i=1}^{R} \left\langle \mathbf{u}_{j,i}, \mathbf{v}_{j,i} \right\rangle^2 &\geq \frac{1}{R} \left\| V_j \right\|_F^2 = \frac{1}{R} \sum\nolimits_{i=1}^{R} \left\| \mathbf{v}_{j,i} \right\|^2 \\
\text{(by (4.12))} \quad &\geq \frac{\beta_j}{R} \sum\nolimits_{i=1}^{R} \left\| M_{j,i} \right\|_F^2 \\
&= \frac{\beta_j}{R} \sum\nolimits_{i=1}^{R} \left\| \mathcal{A} \times_{j+1} \mathbf{u}_{j+1,i}^{\top} \times_{j+2} \cdots \times_d \mathbf{u}_{d,i}^{\top} \right\|_F^2 \\
&\geq \frac{\beta_j}{R} \sum\nolimits_{i=1}^{R} \left\langle \mathcal{A} \times_{j+1} \mathbf{u}_{j+1,i}^{\top} \times_{j+2} \cdots \times_d \mathbf{u}_{d,i}^{\top}, \mathcal{Y}_{j+1,i} \right\rangle^2 \\
\text{(by (4.7))} \quad &= \frac{\beta_j}{R} \sum\nolimits_{i=1}^{R} \left\langle \mathbf{u}_{j+1,i}, \mathbf{v}_{j+1,i} \right\rangle^2,
\end{aligned} \tag{A.15}$$

where the equality in the third line follows from that $M_{j,i}$ is the unfolding of $\mathcal{B}_{j,i} = \mathcal{A} \times_{j+1} \mathbf{u}_{j+1,i}^{\top} \times_{j+2} \cdots \times_d \mathbf{u}_d^{\top}$, and the third inequality uses the fact that $\left\| \mathcal{Y}_{j+1,i} \right\|_F = 1$ where $\mathcal{Y}_{j,i}$ was defined in (4.8).

When $j = d-1$, since $\mathcal{Y}_{d,i}$ is not defined, there is a slight difference after the third line of (A.15). In this case, we have

$$\begin{aligned}
\sum\nolimits_{i=1}^{R} \left\langle \mathbf{u}_{d-1,i}, \mathbf{v}_{d-1,i} \right\rangle^2 &\geq \frac{\beta_{d-1}}{R} \sum\nolimits_{i=1}^{R} \left\| \mathcal{A} \times_d \mathbf{u}_{d,i}^{\top} \right\|_F^2 \\
&= \frac{\beta_{d-1}}{R} \sum\nolimits_{i=1}^{R} \lambda_i (A_{(d)})^2 = \frac{\beta_{d-1}}{R} \sum\nolimits_{i=1}^{R} \left\langle \mathbf{u}_{d,i}, \mathbf{v}_{d,i} \right\rangle^2,
\end{aligned}$$

where the first inequality comes from a similar analysis as (A.15), the first equality follows from the definition of U_d, and the last one is due to the definition of V_d in (4.9). The desired inequality thus follows. □

Proof of Lemma 4.6 If $\mathbf{v}_{j,i}$ is generated by Procedure A, we have

$$\left\| \mathbf{v}_{j,i} \right\|^2 = \lambda_{\max}^2 (M_{j,i}) \geq \frac{1}{n_j} \left\| M_{j,i} \right\|_F^2,$$

where $\lambda_{\max}(\cdot)$ denotes the largest singular value of a matrix.

If $\mathbf{v}_{j,i}$ is generated by Procedure B, we have

$$\left\| \mathbf{v}_{j,i} \right\|^2 = \left\| M_{j,i} \mathbf{y}_{j,i} \right\|^2 = \sum_{k=1}^{n_j} \left\langle \mathbf{m}_{j,i}^{k}, \frac{\mathbf{m}_{j,i}^{\bar{k}}}{\left\| \mathbf{m}_{j,i}^{\bar{k}} \right\|} \right\rangle^2 \geq \left\| \mathbf{m}_{j,i}^{\bar{k}} \right\|^2 \geq \frac{1}{n_j} \left\| M_{j,i} \right\|_F^2,$$

where we recall that $\mathbf{m}_{j,i}^{k}$ denotes the k-th row of $M_{j,i}$, and $\mathbf{m}_{j,i}^{\bar{k}}$ is the row with the largest magnitude.

If $\mathbf{v}_{j,i}$ is generated by Procedure C, since $\bar{k}$ is uniformly and randomly chosen from $\{1, \ldots, n_j\}$, this means that with equal probability, the random variable $\left\| \mathbf{v}_{j,i} \right\|^2$ takes the value $\sum_{k=1}^{n_j} \left\langle \mathbf{m}_{j,i}^k, \mathbf{m}_{j,i}^s / \left\| \mathbf{m}_{j,i}^s \right\| \right\rangle^2$, $s = 1, \ldots, n_j$, i.e.,

$$\mathsf{Prob}\left\{ \left\| \mathbf{v}_{j,i} \right\|^2 = \sum_{k=1}^{n_j} \left\langle \mathbf{m}_{j,i}^k, \frac{\mathbf{m}_{j,i}^s}{\left\| \mathbf{m}_{j,i}^s \right\|} \right\rangle^2 \right\} = \frac{1}{n_j},\ s = 1, \ldots, n_j.$$

Therefore,

$$\mathsf{E} \left\| \mathbf{v}_{j,i} \right\|^2 = \frac{1}{n_j} \sum_{s=1}^{n_j} \sum_{k=1}^{n_j} \left\langle \mathbf{m}_{j,i}^k, \frac{\mathbf{m}_{j,i}^s}{\left\| \mathbf{m}_{j,i}^s \right\|} \right\rangle^2 \geq \frac{1}{n_j} \sum_{s=1}^{n_j} \left\| \mathbf{m}_{j,i}^s \right\|^2 = \frac{1}{n_j} \left\| M_{j,i} \right\|_F^2 .$$

The proof has been completed. □

Proof of Proposition 4.2 To achieve this, we first show that V_j generated by Algorithm 1 satisfies

$$V_j = U_j^* \cdot \Sigma,\ j = d - t + 1, \ldots, d, \tag{A.16}$$

where $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_R)$. If (A.16) holds, then since U_j generated by the algorithm is given by the polar decomposition of V_j, by Assumption 4.1, it follows

$$U_j = U_j^*,\ j = d - t + 1, \ldots, d. \tag{A.17}$$

(A.16) can be proved by the induction method starting from $j = d$. When $j = d$, this is in fact given by (4.9). To see this, we only need to show that $\lambda_i(A_{(d)}) = \sigma_i$ where we recall that $\lambda_i(\cdot)$ denotes the i-th largest singular value of a matrix. Let $U_{-d}^* \in \mathbb{R}^{\prod_{j=1}^{d-1} n_j \times R}$ be the matrix whose i-th column is the vectorization of $\bigotimes_{j=1}^{d-1} \mathbf{u}_{j,i}^*$. Under Assumption 4.1 when $t \geq 2$, U_{-d}^* is orthonormal, giving that $U_d^* \Sigma (U_{-d}^*)^\top$ is a reduced SVD of $A_{(d)}$, and the results follow. Assume that (A.16) holds when $j = d, d-1, \ldots, m$; then $U_j = U_j^*$, $j = d, d-1, \ldots, m$. Therefore, it follows from the algorithm that

$$\mathcal{B}_{m-1,i} = \mathcal{A} \times_m \mathbf{u}_{m,i}^\top \times_{m+1} \times \cdots \times_d \mathbf{u}_{d,i}^\top = \sigma_i \bigotimes_{j=1}^{m-1} \mathbf{u}_{j,i}^*,$$

which is a rank-1 tensor. Since $M_{j-1,i}$ is the unfolding of $\mathcal{B}_{m-1,i}$, it can be seen that $\mathbf{y}_{m-1,i}$ generated by Procedures A–C is just the vectorization of $\bigotimes_{j=1}^{m-2} \mathbf{u}_{j,i}^*$, and so $\mathbf{v}_{m-1,i} = \sigma_i \mathbf{u}_{m-1,i}^*$, which demonstrates that (A.16) holds when $j = m - 1$. Thus, (A.16) and (A.17) are valid for $j = d - t + 1, \ldots, d$.

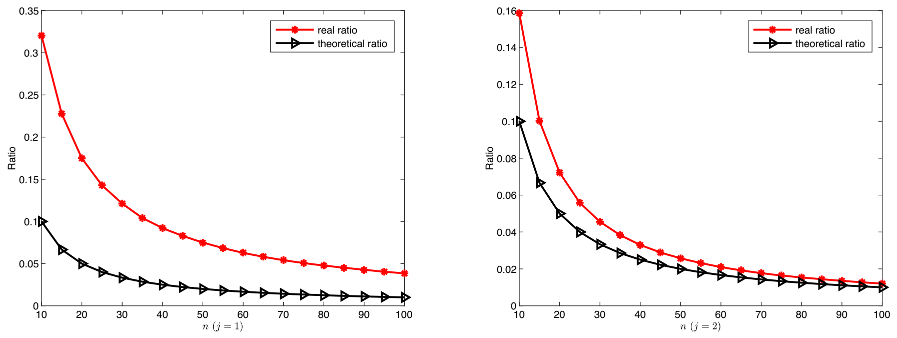
Since $\mathcal{A} \times_{d-t+1} \mathbf{u}_{d-t+1,i}^\top \times_{m+1} \times \cdots \times_d \mathbf{u}_{d,i}^\top$ is of rank-1, the rank-1 approximation procedure generates that $\mathbf{u}_{j,i} = \mathbf{u}_{j,i}^*$, $j = 1, \ldots, d - t$. Hence, $G(U_1, \ldots, U_d) = G(U_1^*, \ldots, U_d^*) = \sum_{i=1}^R \sigma_i^2$. This completes the proof. □

B Empirical Discussions on the Approximation Bound

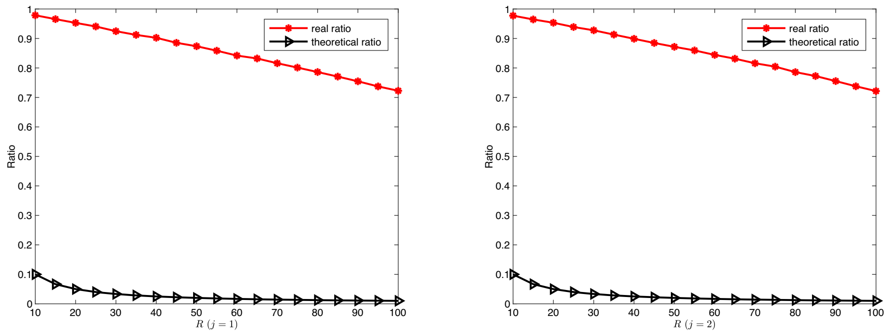
Lemma 4.5 shows that the approximation ratio arises from two folds of each factor: the lower bound of $\left\|\mathbf{v}_{j,i}\right\|^2 / \left\|M_{j,i}\right\|_F^2$ and the lower bound of $\sum_{i=1}^R \left\langle\mathbf{u}_{j,i}, \mathbf{v}_{j,i}\right\rangle^2 / \left\|V_j\right\|_F^2$. The former reaches the worst bound $1/n_j$ when $M_{j,i}$ is row-wisely orthogonal and each row has equal magnitude; the latter attains the worst bound $1/R$ when V_j is of rank-1 and each column admits equal magnitude. In reality, these two worst cases rarely occur simultaneously, meaning that it is possible to further improve the theoretical lower bound. Here, we show the real ratio and the theoretical ratio via examples as follows.

We generate tensors $\mathcal{A} \in \mathbb{R}^{n\times n\times n\times n}$, where every entry obeys the standard normal distribution. We set $t = 4$ for the problem. n varies from 10 to 100, where for each n, 50 instances are generated. We use Procedure A in the algorithm. In Fig. 6(a), we plot the curve of the real ratio $\left(\sum_{i=1}^R \left\|\mathbf{v}_{j,i}\right\|^2 / \left\|M_{j,i}\right\|_F^2\right)/R$ and the curve of the theoretical ratio $1/n$ when $j = 1, 2$. The figure shows that although the real ratio is better, it still approaches the theoretical one as n increases. In Fig. 6b, we fix $n = 100$ and vary R from 10 to 100. We plot the real ratio $\sum_{i=1}^R \left\langle\mathbf{u}_{j,i}, \mathbf{v}_{j,i}\right\rangle^2 / \left\|V_j\right\|_F^2$ and the theoretical one $1/R$. The figure shows that the real ratio is far better, while it also decreases as R increases.

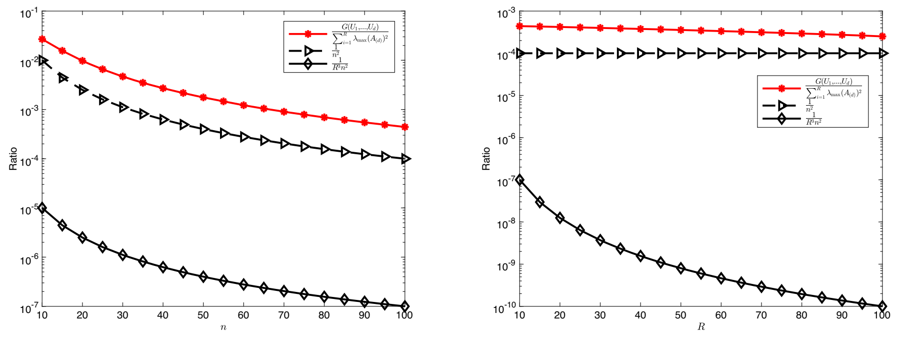
To further investigate the influence of n and R on the bound, we also plot the total real ratio $\frac{G(U_1,\ldots,U_d)}{\sum_{i=1}^R \lambda_{\max}(A_{(d)})^2}$ and the theoretical one $1/R^3n^2$ in Fig. 6c. The tensors are the same as above, where in the left panel, we vary n from 10 to 100 with $R = 10$ fixed; in the right one, R varies from 10 to 100 with $n = 100$ fixed. The real ratio is in red, and the theoretical ratio is in black with diamond markers. We see that the real ratio decreases as n increases, confirming the observation of Fig. 6a. Moreover, the real ratio is almost parallel to the theoretical ratio, showing that the theoretical ratio is reasonable up to a constant (if we view R as a constant). The right panel shows that when R increases, the real ratio does not decrease as much as the theoretical one, showing that the term $1/R^3$ might be too loose in the theoretical ratio. Thus we also plot the curve of $\frac{1}{n^2}$, which is in black with right arrow markers. We see that $1/n^2$ is still under the curve of the real ratio, meaning that it is possible to improve the theoretical ratio in the absence of R.



(a) Plots of the real ratio $\left(\sum_{i=1}^{R} \|\mathbf{v}_{j,i}\|^2 \,/\, \|M_{j,i}\|_F^2\right)/R$ (in red) and the theoretical ratio $1/n$ (in black) for fourth-order tensors. n varies from 10 to 100. Left: $j=1$; right: $j=2$.



(b) Plots of the real ratio $\sum_{i=1}^{R} \langle \mathbf{u}_{j,i}, \mathbf{v}_{j,i}\rangle^2 \,/\, \|V_j\|_F^2$ (in red) and the theoretical ratio $1/R$ (in black) where $\mathcal{A} \in \mathbb{R}^{100\times100\times100\times100}$. R varies from 10 to 100. Left: $j=1$; right: $j=2$.



(c) Plots of real ratio versus theoretical ratio. Left: $\mathcal{A} \in \mathbb{R}^{n\times n\times n\times n}$ with n varying from 10 to 100; right: $\mathcal{A} \in \mathbb{R}^{100\times100\times100\times100}$ with R varying from 10 to 100. In the figures: Red: real ratio $\frac{G(U_1,\ldots,U_d)}{\sum_{i=1}^{R}\lambda_i(A_{(d)})^2}$; black with right arrow markers: $\frac{1}{n^2}$; black with diamond markers: $\frac{1}{R^3n^2}$.

Fig. 6 Real ratio versus theoretical ratio

C Another `get_v_from_M` Procedure

Procedure $\mathbf{v} = \texttt{get\_v\_from\_M}(M)$ (D)

Randomly and uniformly draw a vector $\mathbf{y} \in \mathbb{S}^{m-1}$, where $\mathbb{S}^{m-1}$ denotes the unit sphere in $\mathbb{R}^m$; return $\mathbf{v} = M\mathbf{y}$.

Proposition C.6 *Let $\mathbf{v} \in \mathbb{R}^n$ be generated from $M \in \mathbb{R}^{n\times m}$ by Procedure D. Then it holds that $\mathsf{E}\,\|\mathbf{v}\|^2 = \frac{1}{m}\,\|M\|_F^2$.*

Proof The proof is based on Lemmas C.7 and C.8. □

Lemma C.7 *Let $\mathbf{y}$ be randomly and uniformly drawn from the unit sphere $\mathbb{S}^{m-1}$ in $\mathbb{R}^m$. Then, it holds that*

$$\mathsf{E}(\mathbf{y}(k))^2 = 1/m, \ \ \mathsf{E}\,\mathbf{y}(k_1)\mathbf{y}(k_2) = 0,$$

where $k, k_1, k_2 = 1, \dots, m$, $k_1 \neq k_2$, and $\mathbf{y}(k)$ denotes the k-th entry of $\mathbf{y}$.

Proof The first property comes from He et al. [12, p. 896]. The second one uses symmetry. Let $\mathbf{z} \in \mathbb{R}^n$ be defined such that $\mathbf{z}(k) = \mathbf{y}(k)$ for all $k \neq k_1$, and $\mathbf{z}(k_1) = -\mathbf{y}(k_1)$. Then, $\mathbf{z}$ is also uniformly distributed on the unit sphere, which means that $\mathsf{E}\,\mathbf{z}(k_1)\mathbf{z}(k_2) = \mathsf{E}\,\mathbf{y}(k_1)\mathbf{y}(k_2)$. By the definition of $\mathbf{z}$, this means that $\mathsf{E}\,\mathbf{y}(k_1)\mathbf{y}(k_2) = 0$. Since k_1 and k_2 can be arbitrary, the results follow. □

Lemma C.8 *Let $\mathbf{y}$ be randomly and uniformly drawn from $\mathbb{S}^{m-1}$. For any constant vector $\mathbf{a} \in \mathbb{R}^m$, it holds that $\mathsf{E}\,\langle \mathbf{a}, \mathbf{y}\rangle^2 = m^{-1}\|\mathbf{a}\|^2$.*

Proof We have from Lemma C.7 that

$$\begin{aligned}\mathsf{E}\,\langle \mathbf{a}, \mathbf{y}\rangle^2 &= \mathsf{E}\sum\nolimits_{k_1=1}^{m}\sum\nolimits_{k_2=1}^{m} \mathbf{a}(k_1)\mathbf{y}(k_1)\mathbf{a}(k_2)\mathbf{y}(k_2)\\ &= \sum\nolimits_{k=1}^{m} (\mathbf{a}(k))^2(\mathbf{y}(k))^2 + \sum\nolimits_{k_1\neq k_2} \mathbf{a}(k_1)\mathbf{a}(k_2)\mathbf{y}(k_1)\mathbf{y}(k_2) = m^{-1}\|\mathbf{a}\|^2.\end{aligned}$$

□

Remark C.4 Proposition C.6 shows that if Procedure D is used in Algorithm 1, then the approximation bound will be worse than that based on the other three procedures, as m is usually far larger than n in our context. Thus, we only leave this procedure in the appendix for potential interest.

References

1. Ahmadi-Asl, S., Abukhovich, S., Asante-Mensah, M.G., Cichocki, A., Phan, A.H., Tanaka, T., Oseledets, I.: Randomized algorithms for computation of Tucker decomposition and higher order SVD (HOSVD). IEEE Access **9**, 28684–28706 (2021)

2. Che, M., Wei, Y., Yan, H.: The computation of low multilinear rank approximations of tensors via power scheme and random projection. SIAM J. Matrix Anal. Appl. **41**(2), 605–636 (2020)
3. Chen, J., Saad, Y.: On the tensor SVD and the optimal low rank orthogonal approximation of tensors. SIAM J. Matrix Anal. Appl. **30**(4), 1709–1734 (2009)
4. Cichocki, A., Mandic, D., De Lathauwer, L., Zhou, G., Zhao, Q., Caiafa, C., Phan, H.A.: Tensor decompositions for signal processing applications: from two-way to multiway component analysis. IEEE Signal Process. Mag. **32**(2), 145–163 (2015)
5. Comon, P.: Independent component analysis, a new concept? Signal Process. **36**(3), 287–314 (1994)
6. Comon, P.: Tensors: a brief introduction. IEEE Signal Process. Mag. **31**(3), 44–53 (2014)
7. De Lathauwer, L.: A short introduction to tensor-based methods for factor analysis and blind source separation. In: 2011 7th International Symposium on Image and Signal Processing and Analysis (ISPA), pp. 558–563. IEEE (2011)
8. De Lathauwer, L., De Moor, B., Vandewalle, J.: A multilinear singular value decomposition. SIAM J. Matrix Anal. Appl. **21**(4), 1253–1278 (2000)
9. Fu, T., Jiang, B., Li, Z.: Approximation algorithms for optimization of real-valued general conjugate complex forms. J. Glob. Optim. **70**(1), 99–130 (2018)
10. Grasedyck, L.: Hierarchical singular value decomposition of tensors. SIAM J. Matrix Anal. Appl. **31**(4), 2029–2054 (2010)
11. Guan, Y., Chu, D.: Numerical computation for orthogonal low-rank approximation of tensors. SIAM J. Matrix Anal. Appl. **40**(3), 1047–1065 (2019)
12. He, S., Jiang, B., Li, Z., Zhang, S.: Probability bounds for polynomial functions in random variables. Math. Oper. Res. **39**(3), 889–907 (2014)
13. He, S., Li, Z., Zhang, S.: Approximation algorithms for homogeneous polynomial optimization with quadratic constraints. Math. Program. Ser. B **125**, 353–383 (2010)
14. Higham, N.J.: Computing the polar decomposition-with applications. SIAM J. Sci. Stat. Comput. **7**(4), 1160–1174 (1986)
15. Hu, S., Li, G.: Convergence rate analysis for the higher order power method in best rank one approximations of tensors. Numer. Math. **140**(4), 993–1031 (2018)
16. Hu, S., Ye, K.: Linear convergence of an alternating polar decomposition method for low rank orthogonal tensor approximations. arXiv preprint arXiv:1912.04085 (2019)
17. Jiang, B., Ma, S., Zhang, S.: Tensor principal component analysis via convex optimization. Math. Program. Ser. A **150**, 423–457 (2015)
18. Kofidis, E., Regalia, P.: On the best rank-1 approximation of higher-order supersymmetric tensors. SIAM J. Matrix Anal. Appl. **23**(3), 863–884 (2002)
19. Kolda, T.G.: Orthogonal tensor decompositions. SIAM J. Matrix Anal. Appl. **23**(1), 243–255 (2001)
20. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Rev. **51**(3), 455–500 (2009)
21. Kruskal, J.B.: Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. Linear Algebra Appl. **18**(2), 95–138 (1977)
22. Kuhn, H.W.: The Hungarian method for the assignment problem. Nav. Res. Log. Q. **2**(1–2), 83–97 (1955)
23. Li, J., Usevich, K., Comon, P.: Globally convergent Jacobi-type algorithms for simultaneous orthogonal symmetric tensor diagonalization. SIAM J. Matrix Anal. Appl. **39**(1), 1–22 (2018)
24. Li, J., Zhang, S.: Polar decomposition based algorithms on the product of Stiefel manifolds with applications in tensor approximation. arXiv preprint arXiv:1912.10390 (2019)
25. Li, Z., Nakatsukasa, Y., Soma, T., Uschmajew, A.: On orthogonal tensors and best rank-one approximation ratio. SIAM J. Matrix Anal. Appl. **39**(1), 400–425 (2018)
26. Lim, L.H., Comon, P.: Nonnegative approximations of nonnegative tensors. J. Chemom. **23**(7–8), 432–441 (2009)
27. Lim, L.H., Comon, P.: Blind multilinear identification. IEEE Trans. Inf. Theory **60**(2), 1260–1280 (2014)
28. Martin, C.D.M., Van Loan, C.F.: A Jacobi-type method for computing orthogonal tensor decompositions. SIAM J. Matrix Anal. Appl. **30**(3), 1219–1232 (2008)
29. Minster, R., Saibaba, A.K., Kilmer, M.E.: Randomized algorithms for low-rank tensor decompositions in the Tucker format. SIAM J. Math. Data Sci. **2**(1), 189–215 (2020)
30. Nie, J., Wang, L.: Semidefinite relaxations for best rank-1 tensor approximations. SIAM J. Matrix Anal. Appl. **35**(3), 1155–1179 (2014)

31. Pan, J., Ng, M.K.: Symmetric orthogonal approximation to symmetric tensors with applications to image reconstruction. Numer. Linear Algebra Appl. **25**(5), e2180 (2018)
32. Pesquet-Popescu, B., Pesquet, J.C., Petropulu, A.P.: Joint singular value decomposition-a new tool for separable representation of images. In: Proceedings 2001 International Conference on Image Processing, vol. 2, pp. 569–572. IEEE (2001)
33. Qi, L.: The best rank-one approximation ratio of a tensor space. SIAM J. Matrix Anal. Appl. **32**(2), 430–442 (2011)
34. Qi, Y., Comon, P., Lim, L.H.: Semialgebraic geometry of nonnegative tensor rank. SIAM J. Matrix Anal. Appl. **37**(4), 1556–1580 (2016)
35. Shashua, A., Levin, A.: Linear image coding for regression and classification using the tensor-rank principle. In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, vol. 1, pp. I–I. IEEE (2001)
36. Sidiropoulos, N.D., Bro, R.: On the uniqueness of multilinear decomposition of N-way arrays. J. Chemom. **14**(3), 229–239 (2000)
37. Sidiropoulos, N.D., De Lathauwer, L., Fu, X., Huang, K., Papalexakis, E., Faloutsos, C.: Tensor decomposition for signal processing and machine learning. IEEE Trans. Signal Process. **65**(13), 3551–3582 (2017)
38. Sidiropoulos, N.D., Giannakis, G.B., Bro, R.: Blind PARAFAC receivers for DS-CDMA systems. IEEE Trans. Signal Process. **48**(3), 810–823 (2000)
39. da Silva, A.P., Comon, P., de Almeida, A.L.F.: A finite algorithm to compute rank-1 tensor approximations. IEEE Signal Process. Lett. **23**(7), 959–963 (2016)
40. So, A.M.C.: Deterministic approximation algorithms for sphere constrained homogeneous polynomial optimization problems. Math. Program., Ser. B **129**(2), 357–382 (2011)
41. Sørensen, M., De Lathauwer, L., Comon, P., Icart, S., Deneire, L.: Canonical polyadic decomposition with a columnwise orthonormal factor matrix. SIAM J. Matrix Anal. Appl. **33**(4), 1190–1213 (2012)
42. Sørensen, M., De Lathauwer, L., Deneire, L.: PARAFAC with orthogonality in one mode and applications in DS-CDMA systems. In: 2010 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 4142–4145. IEEE (2010)
43. Sun, W., Wang, J., Fang, Y.: Regularized k-means clustering of high-dimensional data and its asymptotic consistency. Electron. J. Stat. **6**, 148–167 (2012)
44. Uschmajew, A.: Local convergence of the alternating least squares algorithm for canonical tensor approximation. SIAM J. Matrix Anal. Appl. **33**(2), 639–652 (2012)
45. Vannieuwenhoven, N., Vandebril, R., Meerbergen, K.: A new truncation strategy for the higher-order singular value decomposition. SIAM J. Sci. Comput. **34**(2), A1027–A1052 (2012)
46. Vervliet, N., Debals, O., Sorber, L., Van Barel, M., De Lathauwer, L.: Tensorlab 3.0 (2016). http://www.tensorlab.net
47. Wang, L., Chu, M.T., Yu, B.: Orthogonal low rank tensor approximation: alternating least squares method and its global convergence. SIAM J. Matrix Anal. Appl. **36**(1), 1–19 (2015)
48. Yang, Y.: The epsilon-alternating least squares for orthogonal low-rank tensor approximation and its global convergence. SIAM J. Matrix Anal. Appl. **41**(4), 1797–1825 (2020)
49. Yang, Y., Feng, Y., Huang, X., Suykens, J.A.K.: Rank-1 tensor properties with applications to a class of tensor optimization problems. SIAM J. Optim. **26**(1), 171–196 (2016)
50. Zhang, J., Saibaba, A.K., Kilmer, M.E., Aeron, S.: A randomized tensor singular value decomposition based on the t-product. Numer. Linear Algebra Appl. **25**(5), e2179 (2018)
51. Zhang, X., Qi, L., Ye, Y.: The cubic spherical optimization problems. Math. Comput. **81**(279), 1513–1525 (2012)