Convergence Analysis of an SVD-based Algorithm for the Best Rank-1 Tensor Approximation

Yu Guan[a,*], Moody T. Chu[b,1], Delin Chu[a,2]

[a]*Department of Mathematics, National University of Singapore, Singapore 119076*
[b]*Department of Mathematics, North Carolina State University, Raleigh, NC 27695-8205.*

Abstract

This paper revisits the classical problem of finding the best rank-1 approximation to a generic tensor. The main focus is on providing a mathematical proof for the convergence of the iterates of an SVD-based algorithm. In contrast to the conventional approach by the so called alternating least squares (ALS) method that works to adjust one factor a time, the SVD-based algorithms improve two factors simultaneously. The ALS method is easy to implement, but suffers from slow convergence and easy stagnation at a local solution. It has been suggested recently that the SVD-algorithm might have a better limiting behavior leading to better approximations, yet a theory of convergence has been elusive in the literature. This note proposes a simple tactics to partially close that gap.

Keywords: best rank-1 tensor approximation, singular value decomposition
2010 MSC: 15A15, 15A09, 15A23

1. Introduction

A real-valued tensor of order k can be represented by a k-way array

$$T = [\tau_{i_1,\ldots,i_k}] \in \mathbb{R}^{I_1\times I_2\times\ldots\times I_k}$$

with elements $\tau_{i_1,\ldots,i_k}$ accessed via k indices. A tensor of the form

$$\bigotimes_{\ell=1}^{k}\mathbf{u}^{(\ell)} = \mathbf{u}^{(1)}\otimes\ldots\otimes\mathbf{u}^{(k)} := [u^{(1)}_{i_1}\ldots u^{(k)}_{i_k}],$$

where elements are the products of entries from vectors $\mathbf{u}^{(\ell)} \in \mathbb{R}^{I_\ell}$, $\ell = 1,\ldots,k$, is said to be of rank one. The problem of finding a best rank-1 approximation to T is to determine unit vectors $\mathbf{u}^{(\ell)} \in \mathbb{R}^{I_\ell}$, $\ell = 1,\ldots k$, and a scalar λ such that the functional

$$f(\lambda,\mathbf{u}^{(1)},\ldots,\mathbf{u}^{(k)}) := \|T-\lambda\bigotimes_{\ell=1}^{k}\mathbf{u}^{(\ell)}\|_F^2 = \sum_{i_1,i_2,\ldots,i_k}(\tau_{i_1,\ldots,i_k}-\lambda u^{(1)}_{i_1}\ldots u^{(k)}_{i_k})^2 \tag{1}$$

is minimized. For any fixed unit vectors $\mathbf{u}^{(1)},\ldots,\mathbf{u}^{(k)}$, the optimal value of λ for (1) is given precisely by the length of the projection of the “vector" T onto the direction of the “unit vector" $\bigotimes_{\ell=1}^{k}\mathbf{u}^{(\ell)} \in \mathbb{R}^{I_1\times I_2\times\ldots\times I_k}$, i.e.,

$$\lambda = \lambda(\mathbf{u}^{(1)},\ldots,\mathbf{u}^{(k)}) = \langle T,\bigotimes_{\ell=1}^{k}\mathbf{u}^{(\ell)}\rangle. \tag{2}$$

[*]Corresponding author
Email addresses: a0123906@u.nus.edu (Yu Guan), chu@math.ncsu.edu (Moody T. Chu), matchudl@nus.edu.sg (Delin Chu)

[1]This research is supported in part by the National Science Foundation under grant DMS-1316779.

[2]This research is supported in part by NUS research grant R-146-000-236-114.

Thus, minimizing the orthogonal component of T, as is desired in (1), is equivalent to maximizing the length $|\lambda|$ of the parallel component. In [25], the expression (2) is called the generalized Rayleigh quotient of T relative to $\{\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(k)}\}$. Switching the signs of the variables $\mathbf{u}^{(\ell)}$ if necessary, we may restrict our attention without loss of generality to the case that $\lambda > 0$ only.

Many efforts for finding the extreme values of (2) have been made in the literature, yet the problem is still not settled. See, for example, [3, 13, 14, 15, 16, 22, 25]. The difficulty is partly due to the curse of dimensionality, whence the rapid growth of computational overhead, and partly the nonlinearity, whence the stagnation at a local solution. For example, the alternating least squares (ALS) method works on improving one factor $\mathbf{u}^{(\ell)}$ a time. Assuming the form as a high-order power method, the ALS is easy to implement and has been conventionally employed as the workhorse for low rank tensor approximation. However, the method suffers from slow convergence and easy stagnation at a local solution. Thus it is appealing that maybe alternating two factors simultaneously by employing the singular value decomposition (SVD) as the two-in-one optimization mechanism could give rise to better performance. The idea was mentioned in [9, Section 3.3] with no particular elaboration, and was more carefully postulated in [11] with numerical testing on some synthetic and real data sets of third-order tensors. This approach has the obvious advantage that, starting from the same point, one step of SVD-based iteration is superior to two consecutive steps of ALS iteration. There is no theory at present to support that the improvement by the SVD-based iteration will continue to be superior in the long run. Through numerical experiments, however, it has been suggested that for large scale data the SVD-based method might have better limiting behavior leading to better approximations [11, Section 5].

This paper is not concerned about how fast the different algorithms perform, nor what quality they achieve. Rather, we are curious about the more fundamental question of whether the iteration converges at all. Recall that the convergence theory for the ALS method was established much later than the method had been put into practice [6, 21, 22]. A similar concern is raised for the SVD-based algorithm — the convergence of the generalized Rayleigh quotients is obvious, but the convergence analysis for the iterates themselves has been elusive in the literature [11, Page 947]. In this paper we provide a rigorous mathematical proof for the convergence of iterates from a specific SVD-based algorithm, which thus complements the theory. We learn recently that an independent work in the report [24] also investigates the convergence theory by using the Łojasiewicz gradient inequality [5, 17, 18]. Indeed, we have employed a similar technique in proving the global convergence of the ALS method in [22]. The tactics we develop in this paper for the SVD-based algorithm is an entirely different approach. Our approach relies on only the continuity of singular vectors and real analysis, which, in our opinion, is much more straightforward.

This paper is organized as follows. We begin with a brief review of some basic operations in Section 2 to prepare for the discussion. We describe two variants of SVD-based algorithms in Section 3. The difference is at where the SVD is to be applied. Our main result is presented in Section 4 where we explain the meaning of a tensor being generic and argue the convergence for the most basic algorithm. Finally, though it is not the main objective of this paper, we carry out some exploratory experiments in Section 5 to compare performance between ours and other types of SVD-based algorithms.

2. Basics

Tensors have multiple facets, so we need discern what kind of tensor multiplication is taking place. To facilitate the subsequent discussion, we first introduce a simple notation system that generalizes what we already know from the matrix theory. In the passing, we also mention a few useful tools.

Let the symbol $[\![m]\!]$ denote henceforth the set of integers $\{1, \ldots m\}$ for a given positive integer m. Suppose that the set $[\![k]\!]$ is partitioned as the union of two disjoint nonempty subsets $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_s\}$ and $\boldsymbol{\beta} = \{\beta_1, \ldots, \beta_t\}$, where $s + t = k$. Choosing different partitions of $[\![k]\!]$ offers a convenient tool to dissect a high-dimensional T and exam its cross-sections from different perspectives. An element in the tensor $T \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_k}$ can be identified as $\tau_{[\mathcal{I}|\mathcal{J}]}^{(\boldsymbol{\alpha},\boldsymbol{\beta})}$ where $\mathcal{I} := (i_1, \ldots, i_s)$ and $\mathcal{J} := (j_1, \ldots, j_t)$ contain those indices at locations $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, respectively. Each index in the arrays $\mathcal{I}$ and $\mathcal{J}$ should be within the corresponding range of integers, e.g., $i_1 \in [\![I_{\alpha_1}]\!]$ and so on. Without causing ambiguity, we abbreviate the element as $\tau_{[\mathcal{I}|\mathcal{J}]}$, when the reference to a specific partitioning $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is clear. The representation $\tau_{[\mathcal{I}|\mathcal{J}]}$ is $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ specific, but there is no preference of $\boldsymbol{\alpha}$ over $\boldsymbol{\beta}$. The partition $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ may be regarded as generalizing the familiar notion of rows and columns for matrices.

Given a fixed partitioning $[\![k]\!] = \boldsymbol{\alpha} \cup \boldsymbol{\beta}$, we shall regard an order-k tensor $T \in \mathbb{R}^{I_1 \times \ldots \times I_k}$ as a "matrix representation" of a linear operator mapping order-s tensors to order-t tensors [22]. Specifically, we identify T with the linear map

$$\mathscr{T}_{\boldsymbol{\beta}} : \mathbb{R}^{I_{\alpha_1}\times\ldots\times I_{\alpha_s}} \to \mathbb{R}^{I_{\beta_1}\times\ldots\times I_{\beta_t}}, \tag{3}$$

such that for any $S \in \mathbb{R}^{I_{\alpha_1}\times\ldots\times I_{\alpha_s}}$, we have

$$\mathscr{T}_{\boldsymbol{\beta}}(S) := T \circledast_{\boldsymbol{\beta}} S = [\langle \tau_{[:|\ell_1,\ldots,\ell_t]}, S\rangle] \in \mathbb{R}^{I_{\beta_1}\times\ldots\times I_{\beta_t}} \tag{4}$$

where

$$\langle \tau_{[:|\ell_1,\ldots,\ell_t]}, S\rangle := \sum_{i_1=1}^{I_{\alpha_1}} \ldots \sum_{i_s=1}^{I_{\alpha_s}} \tau_{[i_1,\ldots,i_s|\ell_1,\ldots,\ell_t]} s_{i_1,\ldots,i_s} \tag{5}$$

is the Frobenius inner product generalized to multi-dimensional arrays. The $\boldsymbol{\beta}$-product defined by (4) is a natural generalization of the usual matrix-vector multiplication in the sense that if an order-2 tensor $T \in \mathbb{R}^{m\times n}$ is regarded as a matrix, and if the column is identified by the pointer $\boldsymbol{\alpha} = \{1\}$ and the row by $\boldsymbol{\beta} = \{2\}$ so that $\tau_{ij} = \tau_{[i|j]}^{(\{2\},\{1\})}$, then with respect to given column vectors $\mathbf{z} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^m$ we can write

$$\begin{cases} T\mathbf{z} &= T \circledast_2 \mathbf{z}, \\ T^\top \mathbf{y} &= T \circledast_1 \mathbf{y}. \end{cases} \tag{6}$$

This notation is handy in our convergence analysis later.

We should also carefully differentiate this $\boldsymbol{\beta}$-product $\circledast_{\boldsymbol{\beta}}$ from the so called d-mode product $\times_d$ used in the literature [2, 16]. First, recall that the d-mode product of a tensor with a matrix maintains the same order of the original tensor, but the $\boldsymbol{\beta}$-product reduces the order from $k = s + t$ to t. Second, recall that the d-mode product of a tensor with a vector is indeed a contraction. When S is of the form $S = \mathbf{u}^{(1)} \otimes \ldots \otimes \mathbf{u}^{(s)}$, then

$$(T \circledast_{\boldsymbol{\beta}} S)_{\ell_1,\ldots,\ell_t} = \sum_{i_1=1}^{I_{\alpha_1}} \ldots \sum_{i_s=1}^{I_{\alpha_s}} \tau_{[i_1,\ldots,i_s|\ell_1,\ldots,\ell_t]} u_{i_1}^{(1)} \ldots u_{i_s}^{(s)} = T \times_{\alpha_1} \mathbf{u}^{(1)} \times_{\alpha_2} \mathbf{u}^{(2)} \ldots \times_{\alpha_s} \mathbf{u}^{(s)}.$$

However, we are not aware of a consistent way to define the d-mode product when S is a general order-s tensor.

The following basic facts will be used in the subsequent discussion.

Lemma 2.1. *Given a general tensor $T \in \mathbb{R}^{I_1\times I_2\times\ldots\times I_k}$, a partitioning $[\![k]\!] = \boldsymbol{\alpha} \cup \boldsymbol{\beta}$, and vectors $\mathbf{u}^{(\ell)} \in \mathbb{R}^{I_\ell}$, $\ell = 1, \ldots, k$, then it holds that*

$$\langle T, \bigotimes_{\ell=1}^{k} \mathbf{u}^{(\ell)}\rangle = \langle T \circledast_{\boldsymbol{\beta}} \bigotimes_{i=1}^{s} \mathbf{u}^{(\alpha_i)}, \bigotimes_{j=1}^{t} \mathbf{u}^{(\beta_j)}\rangle. \tag{7}$$

Lemma 2.2. *Given a general tensor $T \in \mathbb{R}^{I_1\times I_2\times\ldots\times I_k}$, arbitrary vectors $\mathbf{u}^{(\alpha_i)} \in \mathbb{R}^{I_{\alpha_i}}$, $i \in [\![k-2]\!]$, $\mathbf{v} \in \mathbb{R}^{I_{\beta_2}}$, and $\mathbf{w} \in \mathbb{R}^{I_{\beta_1}}$, then*

$$(T \circledast_{\{\beta_1,\beta_2\}} \bigotimes_{i=1}^{k-2} \mathbf{u}^{(\alpha_i)}) \circledast_{\beta_1} \mathbf{v} = (T \circledast_{\{\beta_1,\alpha_j\}} \bigotimes_{i=1}^{j-1} \mathbf{u}^{(\alpha_i)} \otimes \mathbf{v} \otimes \bigotimes_{i=j+1}^{k-2} \mathbf{u}^{(\alpha_i)}) \circledast_{\beta_1} \mathbf{u}^{(\alpha_j)} \tag{8}$$

$$(T \circledast_{\{\beta_1,\beta_2\}} \bigotimes_{i=1}^{k-2} \mathbf{u}^{(\alpha_i)}) \circledast_{\beta_2} \mathbf{w} = (T \circledast_{\{\alpha_j,\beta_2\}} \bigotimes_{i=1}^{j-1} \mathbf{u}^{(\alpha_i)} \otimes \mathbf{w} \otimes \bigotimes_{i=j+1}^{k-2} \mathbf{u}^{(\alpha_i)}) \circledast_{\beta_2} \mathbf{u}^{(\alpha_j)} \tag{9}$$

for any $j \in [\![k-2]\!]$.

Lemma 2.3. *[19, Lemma 4.10] Assume that a^* is an isolated accumulation point of a sequence $\{a_k\}$ such that for every subsequence $\{a_{k_j}\}$ converging to a^*, there is an infinite subsequence $\{a_{k_{j_i}}\}$ such that $|a_{k_{j_i}+1} - a_{k_{j_i}}| \to 0$. Then the whole sequence $\{a_k\}$ converges to a^*.*

Lemma 2.4. *[10] Given a matrix $A \in \mathbb{R}^{m\times n}$, then the global maximum of the generalized Rayleigh quotient*

$$\max_{\substack{\mathbf{y}\in\mathbb{R}^m,\ \|\mathbf{y}\|=1 \\ \mathbf{z}\in\mathbb{R}^n,\ \|\mathbf{z}\|=1}} \mathbf{y}^\top A\mathbf{z} \tag{10}$$

is precisely the largest singular value σ_1 of A, where the global maximizer $(\mathbf{y}_1, \mathbf{z}_1)$ consists of precisely the corresponding left and right singular vectors. The best rank-1 approximation to A is given by $\sigma_1\mathbf{y}_1\mathbf{z}_1^\top$.

3. SVD-based best rank-1 approximation

We now investigate a possible application of the singular value decomposition (SVD) to the best rank-1 approximation of a generic tensor. We shall explain the kind of generic property we need in the context. Our goal is to achieve the best rank-1 approximation by improving two components a time via the SVD.

3.1. SVD certification

Suppose that $\lambda \bigotimes_{\ell=1}^{k} \overline{\mathbf{u}}^{(\ell)}$ is the best rank-1 approximation to a given order-k tensor T. By (2), the generalized Rayleigh quotient $\lambda = \langle T, \bigotimes_{\ell=1}^{k} \overline{\mathbf{u}}^{(\ell)} \rangle$ is positive and maximal. Consider an arbitrary partitioning $[\![k]\!] = \boldsymbol{\alpha} \cup \boldsymbol{\beta}$ with the cardinality $|\boldsymbol{\beta}| = 2$. By Lemma 2.1, we can write

$$\lambda = \langle T \circledast_{\boldsymbol{\beta}} \bigotimes_{i=1}^{k-2} \overline{\mathbf{u}}^{(\alpha_i)}, \bigotimes_{j=1}^{2} \overline{\mathbf{u}}^{(\beta_j)} \rangle.$$

The product $C_{\boldsymbol{\beta}} := T \circledast_{\boldsymbol{\beta}} \bigotimes_{i=1}^{k-2} \overline{\mathbf{u}}^{(\alpha_i)}$ is a matrix in $\mathbb{R}^{I_{\beta_1} \times I_{\beta_2}}$. Since λ is the maximal generalized Rayleigh quotient, by Lemma 2.4, we conclude that $\overline{\mathbf{u}}^{(\beta_1)}$ and $\overline{\mathbf{u}}^{(\beta_2)}$ must be the left and the right singular vectors associated with the largest singular value λ of $C_{\boldsymbol{\beta}}$ for any $\boldsymbol{\beta}$. This is the SVD certification of the best rank-1 approximation to a given tensor T.

We are thus motivated to formulate an SVD-based approach to calculate the best rank-1 approximation by iterations. Depending on the choice of $\boldsymbol{\beta}$ which dictates where the certification is to be checked, the approach may appear in different variants. For order-4 tensors, for example, only the two pairs $\boldsymbol{\beta} = (1, 2)$ and $(3, 4)$ are alternatingly checked in [24], whereas all six combinations in the order $\boldsymbol{\beta} = (1, 2), (3, 4), (1, 3), (2, 4), (1, 4), (2, 3)$ are checked in [11]. We propose two alternatives. In Algorithm 1, we circulate through k pairs of $\boldsymbol{\beta}$ in the order $(1, 2), (2, 3), \ldots, (k-1, k)$ and $(1, k)$. In Algorithm 2, we propose a random choice of $\boldsymbol{\beta} = (\sigma_{k-1}, \sigma_k)$ where σ is an arbitrary permutation of $[\![k]\!]$. We do not think that there is a significant difference in the performance among the variants, but the true verdict is yet to be further investigated. In all algorithms, the most fundamental concern is a proof of convergence for generic tensors.

3.2. Algorithm description

The most basic SVD-based approach is outlined in Algorithm 1. Two types of dynamics are involved in this and all other algorithms. One is the dynamics of the objective values, of which the analysis is straightforward. The other is the dynamics of the iterates, which is much harder to characterize. We will discuss the convergence in the next section.

To convey the idea, we adopt the subscript $_{[p]}$ in Algorithm 1 to indicate the quantity at the p-th iteration. Each sweep of p at Line 1 in Algorithm 1 involves k pairs of $\boldsymbol{\beta}$ ranging circularly from $(1, 2), (2, 3), \ldots, (k-1, k)$ and $(1, k)$. It is tricky that last pair has to be in the order $(1, k)$, as the reversal $(k, 1)$ will not work. Each $\mathbf{u}_{[p+1]}^{(\ell)}$ is updated twice. The first updates for $\ell = 2, \ldots, k$, denoted by $\widehat{\mathbf{u}}_{[p+1]}^{(\ell)}$ at Line 10, are not essential and can be completely removed from the algorithm, but its presence helps bridge the monotonicity. The update $\widehat{\mathbf{u}}_{[p+1]}^{(1)}$ is temporarily overwritten as $\mathbf{u}_{[p+1]}^{(1)}$ at Line 9 for the computation of $C_{[p]}^{(\ell)}$ at Line 4 for $\ell = 2, \ldots, k-1$, but will be updated again at Line 17. The switch of signs at Line 7 conditioned upon Line 6 is to ensure that the iterates are aligned in one direction and thus avoid discontinuous jumps. The continuity of the dominant singular value and the associated singular vector is critical to convergence. The intermediate values $\lambda_{[p+1]}^{(\ell)}$ are registered in the algorithm as well, even though only $\lambda_{[p+1]}^{(k)}$ at the final stage is crucial.

The above algorithm is still alternating in nature, but is different from the alternating least squares (ALS) approach that has been popular for computing the best rank-1 approximation [7, 14, 25]. The most significant difference is that, since the dominant singular vector $\mathbf{u}_{[p+1]}^{(\ell)}$ and $\mathbf{v}_{[p+1]}^{(\ell)}$ of the matrix $C_{[p]}^{(\ell)}$ gives rise to the absolute maximal value $\lambda_{[p+1]}^{(\ell)}$ for the functional

$$g(\mathbf{x}, \mathbf{y}) := \langle T, \bigotimes_{i=1}^{\ell-1} \mathbf{u}_{[p+1]}^{(i)} \otimes \mathbf{x} \otimes \mathbf{y} \otimes \bigotimes_{i=\ell+2}^{k} \mathbf{u}_{[p]}^{(i)} \rangle \tag{11}$$

Algorithm 1 (Best rank-1 approximation via SVD updating with cyclic progression.)

Require: An order-k tensor $T \in \mathbb{R}^{I_1 \times \dots \times I_k}$ and k starting unit vectors $\mathbf{u}_{[0]}^{(\ell)} \in \mathbb{R}^{I_\ell}$, $\ell \in [\![k]\!]$
Ensure: A local best rank-1 approximation to T

```
1:  for $p = 0, 1, \cdots,$ do
2:      for $\ell = 1, 2, \cdots, k-1,$ do
3:          $\boldsymbol{\beta}_\ell = (\ell, \ell+1)$
4:          $C_{[p]}^{(\ell)} = T \circledast_{\boldsymbol{\beta}_\ell} \bigotimes_{i=1}^{\ell-1} \mathbf{u}_{[p+1]}^{(i)} \otimes \bigotimes_{i=\ell+2}^{k} \mathbf{u}_{[p]}^{(i)}$    {A matrix of size $I_\ell \times I_{\ell+1}$}
5:          $[\mathbf{u}, s, \mathbf{v}] = \mathsf{svds}(C_{[p]}^{(\ell)}, 1)$    {Dominant singular value triplet via Matlab routine svds; assume
            uniqueness}
6:          if $u_1 < 0$ then
7:              $\mathbf{u} = -\mathbf{u}, \mathbf{v} = -\mathbf{v}$    {Assume the generic case that $\mathbf{u}_1 \neq 0$; otherwise, use another entry.}
8:          end if
9:          $\mathbf{u}_{[p+1]}^{(\ell)} := \mathbf{u}$    {If $\ell = 1$, this is $\widehat{\mathbf{u}}_{[p+1]}^{(1)}$; otherwise this is the second update $\mathbf{u}_{[p+1]}^{(\ell)}$, if $2 \leq \ell < k$.}
10:         $\widehat{\mathbf{u}}_{[p+1]}^{(\ell+1)} := \mathbf{v}$    {Skipping this step will not affect $C_{[p]}^{(\ell+1)}$ at Line 4.}
11:         $\lambda_{[p+1]}^{(\ell)} := s$
12:     end for
13:     $\boldsymbol{\beta}_k = (1, k)$    {Not $(k, 1)$!}
14:     $C_{[p]}^{(k)} = T \circledast_{\boldsymbol{\beta}_k} \bigotimes_{i=2}^{k-1} \mathbf{u}_{[p+1]}^{(i)}$    {A matrix of size $I_1 \times I_k$}
15:     $[\mathbf{u}, s, \mathbf{v}] = \mathsf{svds}(C_{[p]}^{(k)}, 1)$    {Dominant singular value triplet via Matlab routine svds; assume uniqueness}
16:     $\mathbf{u}_{[p+1]}^{(k)} := \mathbf{v}$    {After adjusting the signs of $\mathbf{u}$ and $\mathbf{v}$ properly as in Line 6.}
17:     $\mathbf{u}_{[p+1]}^{(1)} := \mathbf{u}$
18:     $\lambda_{[p+1]}^{(k)} := s$
19: end for
```

among all possible vectors $\mathbf{x}$ and $\mathbf{y}$, the mechanism of updating $\mathbf{x}$ and $\mathbf{y}$ simultaneously in Algorithm 1 is going to increase the generalized Rayleigh quotient faster than the combination of two applications of ALS approach to $\mathbf{x}$ followed by $\mathbf{y}$ in one step, provided that the initial information is the same. The two-in-one gain is also better than the maximum of updating $\mathbf{x}$ or $\mathbf{y}$ separately [11, Proposition 4]. We stress that such an advantage happens only when the comparison is made at the same point. There is no general theory at present to support that the SVD update will continue to be superior to the power update in the long run, once they depart toward different directions from the same starting point.

Other than for systematically bookkeeping the progression of $\boldsymbol{\beta}$, there is no particular reason that we have to cycle through the ℓ-loop as is indicated in Algorithm 1. An alternative way is to shuffle the columns $\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(k)}$ by a random permutation σ and generate a matrix C for updating. This randomized procedure is modified at Line 7 in Algorithm 2. To avoid confusion with data generated from Algorithm 1, we employ a slightly different notation when describing the progression in this algorithm. For simplicity, we always choose to update the last two vectors $\mathbf{u}^{(\sigma_{k-1})}, \mathbf{u}^{(\sigma_k)}$ after the permutation. It is known in probability theory that the expected number of trials for a permutation to recur is $\frac{k(k-1)}{2}$. Nonetheless, by the time that a repetition of permutation $\boldsymbol{\beta}_t$ occurs, the vectors $\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(k)}$ should have been changed. Repeating the random permutations sufficiently many times should get the iteration to move forward. The concern of reiterating with the same matrix C_t at Line 7 should be nominal. It is interesting to note from our numerical experiments in Section 5 that this randomized algorithm turns out to be the most efficient when comparing with other variants.

4. Convergence analysis

In this section, we analyze the convergence for the above algorithms. First, because the SVD at each update always selects the dominant singular value and the corresponding left and right singular vectors, each of the two algorithms enjoys the property that the corresponding sequence of the generalized Rayleigh quotients is bounded and monotone increasing. The convergence of the objective values (2) is obvious.

Algorithm 2 (Best rank-1 approximation via SVD updating with randomization.)

Require: An order-k generic tensor T and k starting unit vectors $\mathbf{u}^{(1)},\dots,\mathbf{u}^{(k)} \in \mathbb{R}^n$
Ensure: A local best rank-1 approximation to T

```
1: $t \leftarrow 0$
2: $\lambda_0 \leftarrow \langle T, \bigotimes_{\ell=1}^{k} \mathbf{u}^{(\ell)} \rangle$
3: repeat
4:     $t \leftarrow t + 1$
5:     $\sigma \leftarrow$ random permutation of $\{1,\dots,k\}$
6:     $\boldsymbol{\beta}_t \leftarrow (\sigma_{k-1}, \sigma_k)$
7:     $C_t \leftarrow T \circledast_{\boldsymbol{\beta}_t} \bigotimes_{i=1}^{k-2} \mathbf{u}^{(\sigma_i)}$
8:     $[\mathbf{u}_t, s_t, \mathbf{v}_t] = \mathsf{svds}(C_t, 1)$  {Dominant singular value triplet via Matlab routine svds, assume uniqueness}
9:     if $(\mathbf{u}_t)_1 < 0$ then
10:        $\mathbf{u} = -\mathbf{u}_t, \mathbf{v} = -\mathbf{v}_t$        {Assume the general case that $(\mathbf{u}_t)_1 \neq 0$; otherwise, use another entry}
11:    end if
12:    $\lambda_t \leftarrow s_t$
13:    $\mathbf{u}^{(\sigma_{k-1})} \leftarrow \mathbf{u}_t, \mathbf{u}^{(\sigma_k)} \leftarrow \mathbf{v}_t$
14: until $\lambda_t$ meets convergence criteria
```

Lemma 4.1. *The scalars $\{\lambda_{[p]}^{(\ell)}\}$ generated in Algorithm 1 form a monotone convergent sequence for each $\ell = 1,\dots,k$ and all converge to the same value.*

Lemma 4.2. *The scalars $\{\lambda_t\}$ generated in Algorithm 2 form a monotone convergent sequence.*

It remains to prove the convergence of iterates themselves under generic conditions [24, Assumption 3.1]. What happens is that there are cases where the iterates do not converge [16], but these cases form algebraic varieties, i.e., zeros of a certain polynomial system, that are of measure zero in the space of general tensors. The complement of this zero measure set is open and dense under the Zariski topology [20], which is what we referred to as generic. To avoid using jargons from algebraic geometry, we shall be more specific in the following argument when generic properties are required.

We learn recently that authors of the report [24] independently prove the convergence of their variant of an SVD-based algorithm by exploiting the monotone convergence of values $\lambda_{[p]}^{(k)}$ and λ_t. Their proof relies on the framework developed in [1] and utilizes the the Łojasiewicz gradient inequality. A similar idea has been employed in our earlier work in [22]. Our contribution in this paper is a new, shorter, and more direct proof. In either case, the analysis should fulfill what was declared as "we do not have a complete understanding when this will happen" in [11, Page 947].

4.1. Convergence of Algorithm 1

The two SVD-based algorithms outlined in the proceeding section differ by the way $\boldsymbol{\beta}$ is specified. To convey our idea, we first characterize the limiting behavior of Algorithm 1 where $\boldsymbol{\beta}$ is changed systematically in a cyclic pattern. Observe that for each fixed ℓ, because $\|\mathbf{u}_{[p]}^{(\ell)}\|_2 = 1$ for all p, the collection $\{\mathbf{u}_{[p]}^{(\ell)}\}$ must have a convergent subsequence. There are only finitely many ℓ. Selecting a subsequence of a subsequence if necessary, we can find a common subset $\{p_j\}$ of nonnegative integers so that $\{\mathbf{u}_{[p_j]}^{(\ell)}\}$ converges simultaneously for all $\ell \in [\![k]\!]$.

Lemma 4.3. *If subsequences $\{\mathbf{u}_{[p_j]}^{(\ell)}\}$ generated by Algorithm 1 converge simultaneously for all $\ell \in [\![k]\!]$, then so do subsequences $\{C_{[p_j]}^{(\ell)}\}$ and $\{\mathbf{u}_{[p_j+1]}^{(\ell)}\}$.*

Proof. The simultaneous convergence of $\{\mathbf{u}_{[p_j]}^{(\ell)}\}$ for $\ell = 3,\dots k$ implies that the subsequence $\{C_{[p_j]}^{(1)}\}$ converges. By the continuity inherited in the SVD [4, 23], the subsequence of the left singular vectors $\{\widehat{\mathbf{u}}_{[p_j+1]}^{(1)}\}$ of $C_{[p_j]}^{(1)}$ converges also since we have already aligned them in one direction. But then by definition, $\{C_{[p_j]}^{(2)}\}$ converges and, thus, so does $\{\mathbf{u}_{[p_j+1]}^{(2)}\}$. We can repeat this argument by cycling through the ℓ-loop in Algorithm 1. At the end, the matrices $\{C^{(k)}_{[p_j]}\}$ together with the corresponding left singular vectors $\{\mathbf{u}^{(1)}_{[p_j+1]}\}$ and the right singular vectors $\{\mathbf{u}^{(k)}_{[p_j+1]}\}$ must also converge. $\square$

Denote the respective limit points of the above subsequences by

$$\left\{\begin{array}{lcl} \lim_{j\to\infty}\mathbf{u}^{(\ell)}_{[p_j]} &=& \mathbf{u}^{(\ell)}_{\natural}, \\ \lim_{j\to\infty} C^{(\ell)}_{[p_j]} &=& C^{(\ell)}, \\ \lim_{j\to\infty}\widehat{\mathbf{u}}^{(\ell)}_{[p_j+1]} &=& \widehat{\mathbf{u}}^{(\ell)}_{\sharp}, \\ \lim_{j\to\infty}\mathbf{u}^{(\ell)}_{[p_j+1]} &=& \mathbf{u}^{(\ell)}_{\sharp}, \end{array}\right. \quad \ell\in [\![k]\!], \tag{12}$$

where the subscript $_\sharp$ is a handy way to distinguish, at least for now, that the limit points corresponding to the subsequence $_{[p_j+1]}$ might be different from those, denoted by the subscript $_\natural$, of the original subsequence $_{[p_j]}$. By the way $C^{(\ell)}_{[p_j]}$ is defined, it follows that

$$C^{(\ell)} = T \circledast_{\boldsymbol{\beta}_\ell} (\bigotimes_{i=1}^{\ell-1}\mathbf{u}^{(i)}_{\sharp}\otimes\bigotimes_{i=\ell+2}^{k}\mathbf{u}^{(i)}_{\natural}). \tag{13}$$

We already point out in Lemma 4.1 that all matrices $C^{(\ell)}$, $\ell\in[\![k]\!]$ share the same dominant singular value $\widetilde{\lambda}$. We now explore the relationships among dominant singular vectors of all $C^{(\ell)}$ matrices.

With respect to a given convergent subsequence generated by Algorithm 1, the following result asserts that all dominant (left) singular vectors are the same.

Lemma 4.4. *Assume that T is such that, with respect to the given simultaneously convergent subsequences $\{\mathbf{u}^{(\ell)}_{[p_j]}\}$ generated by Algorithm 1, the dominant singular value $\widetilde{\lambda}$ of the corresponding limit point $C^{(\ell)}\in\mathbb{R}^{I_\ell\times I_{\ell+1}}$ defined in (12) is simple for all $\ell\in[\![k]\!]$. Then the limit points defined in (12) satisfy the relationships that*

$$\left\{\begin{array}{lcl} \mathbf{u}^{(\ell)}_{\natural} &=& \widehat{\mathbf{u}}^{(\ell)}_{\sharp} = \mathbf{u}^{(\ell)}_{\sharp}, \\ C^{(\ell)} &=& T\circledast_{\boldsymbol{\beta}_\ell}(\bigotimes_{i=1}^{\ell-1}\mathbf{u}^{(i)}_{\natural}\otimes\bigotimes_{i=\ell+2}^{k}\mathbf{u}^{(i)}_{\natural}), \end{array}\right. \quad \ell\in[\![k]\!]. \tag{14}$$

Proof. For convenience, we employ the abbreviations $\circledast_1$ and $\circledast_2$ to indicate, respectively, the row-matrix and matrix-column multiplications already delineated in (6). In reality, it must be noted that we are dealing with multiplications of matrices and vectors of different sizes.

By using Lemma 2.2, we first observe the equalities

$$\begin{array}{rcl} C^{(\ell)}_{[p_j]}\circledast_1\mathbf{u}^{(\ell)}_{[p_j+1]} &=& \lambda^{(\ell)}_{[p_j+1]}\widehat{\mathbf{u}}^{(\ell+1)}_{[p_j+1]} \\ &=& (T\circledast_{\boldsymbol{\beta}_\ell}(\widehat{\mathbf{u}}^{(1)}_{[p_j+1]}\otimes\bigotimes_{i=2}^{\ell-1}\mathbf{u}^{(i)}_{[p_j+1]}\otimes\bigotimes_{i=\ell+2}^{k}\mathbf{u}^{(i)}_{[p_j]}))\circledast_1\mathbf{u}^{(\ell)}_{[p_j+1]} \\ &=& (T\circledast_{\boldsymbol{\beta}_{\ell+1}}(\widehat{\mathbf{u}}^{(1)}_{[p_j+1]}\otimes\bigotimes_{i=2}^{\ell}\mathbf{u}^{(i)}_{[p_j+1]}\otimes\bigotimes_{i=\ell+3}^{k}\mathbf{u}^{(i)}_{[p_j]}))\circledast_2\mathbf{u}^{(\ell+2)}_{[p_j]} \\ &=& C^{(\ell+1)}_{[p_j]}\circledast_2\mathbf{u}^{(\ell+2)}_{[p_j]}, \quad \ell = 2\dots k-2. \end{array} \tag{15}$$

Similarly,

$$\begin{array}{rcl} C^{(1)}_{[p_j]}\circledast_1\widehat{\mathbf{u}}^{(1)}_{[p_j+1]} &=& \lambda^{(1)}_{[p_j+1]}\widehat{\mathbf{u}}^{(2)}_{[p_j+1]} \\ &=& (T\circledast_{\boldsymbol{\beta}_1}(\bigotimes_{i=3}^{k}\mathbf{u}^{(i)}_{[p_j]}))\circledast_1\widehat{\mathbf{u}}^{(1)}_{[p_j+1]} \\ &=& (T\circledast_{\boldsymbol{\beta}_2}(\widehat{\mathbf{u}}^{(1)}_{[p_j+1]}\otimes\bigotimes_{i=4}^{k}\mathbf{u}^{(i)}_{[p_j]}))\circledast_2\mathbf{u}^{(3)}_{[p_j]} = C^{(2)}_{[p_j]}\circledast_2\mathbf{u}^{(3)}_{[p_j]}. \end{array} \tag{16}$$

The special "twist" at Lines 16-17 with $\boldsymbol{\beta}_k = (1,k)$ in Algorithm 1 allows us to have the identities

$$\begin{aligned} C^{(k)}_{[p_j-1]} \circledast_2 \mathbf{u}^{(k)}_{[p_j]} &= \lambda^{(k)}_{[p_j]} \mathbf{u}^{(1)}_{[p_j]} \\ &= (T \circledast_{\boldsymbol{\beta}_k} (\bigotimes_{i=2}^{k-1} \mathbf{u}^{(i)}_{[p_j]})) \circledast_2 \mathbf{u}^{(k)}_{[p_j]} \\ &= (T \circledast_{\boldsymbol{\beta}_1} (\bigotimes_{i=3}^{k} \mathbf{u}^{(i)}_{[p_j]})) \circledast_2 \mathbf{u}^{(2)}_{[p_j]} = C^{(1)}_{[p_j]} \circledast_2 \mathbf{u}^{(2)}_{[p_j]}. \end{aligned} \tag{17}$$

Finally, we also have

$$\begin{aligned} C^{(k-1)}_{[p_j]} \circledast_1 \mathbf{u}^{(k-1)}_{[p_j+1]} &= \lambda^{(k-1)}_{[p_j+1]} \widehat{\mathbf{u}}^{(k)}_{[p_j+1]} \\ &= (T \circledast_{\boldsymbol{\beta}_{k-1}} (\widehat{\mathbf{u}}^{(1)}_{[p_j+1]} \otimes \bigotimes_{i=2}^{k-2} \mathbf{u}^{(i)}_{[p_j+1]})) \circledast_1 \mathbf{u}^{(k-1)}_{[p_j+1]} \\ &= (T \circledast_{\boldsymbol{\beta}_k} (\bigotimes_{i=2}^{k-1} \mathbf{u}^{(i)}_{[p_j+1]})) \circledast_1 \widehat{\mathbf{u}}^{(1)}_{[p_j+1]} = C^{(k)}_{[p_j]} \circledast_1 \widehat{\mathbf{u}}^{(1)}_{[p_j+1]}. \end{aligned} \tag{18}$$

Taking the limits, then it follows by construction and continuity that we have the relationships:

$$\widetilde{\lambda} \mathbf{u}^{(1)}_{\natural} = C^{(1)} \circledast_2 \mathbf{u}^{(2)}_{\natural}, \quad \text{(by (17))} \tag{19}$$

$$C^{(1)} \circledast_1 \widehat{\mathbf{u}}^{(1)}_{\sharp} = \widetilde{\lambda} \widehat{\mathbf{u}}^{(2)}_{\sharp} = C^{(2)} \circledast_2 \mathbf{u}^{(3)}_{\natural}, \quad \text{(by (16))} \tag{20}$$

$$C^{(\ell)} \circledast_1 \mathbf{u}^{(\ell)}_{\sharp} = \widetilde{\lambda} \widehat{\mathbf{u}}^{(\ell+1)}_{\sharp} = C^{(\ell+1)} \circledast_2 \mathbf{u}^{(\ell+2)}_{\natural}, \quad \ell = 2, \ldots, k-2, \quad \text{(by (15))} \tag{21}$$

$$C^{(k-1)} \circledast_1 \mathbf{u}^{(k-1)}_{\sharp} = \widetilde{\lambda} \widehat{\mathbf{u}}^{(k)}_{\sharp} = C^{(k)} \circledast_1 \widehat{\mathbf{u}}^{(1)}_{\sharp}, \quad \text{(by (18))} \tag{22}$$

$$C^{(k)} \circledast_1 \mathbf{u}^{(1)}_{\sharp} = \widetilde{\lambda} \mathbf{u}^{(k)}_{\sharp}. \qquad \text{(by Lines 16-17 in Algorithm 1)} \tag{23}$$

By assumption, the dominant $\widetilde{\lambda}$ is simple and the corresponding singular vector is unique up to a sign change. However, because in Lines 6 to 8 of Algorithm 1 we have already required that the first entry of the dominant singular vector be positive, such a sign change does not exist. The best rank-1 approximation to the matrix $C^{(\ell)}$ therefore is unique. Recursively, the above relationships imply that the best rank-1 approximation to the matrix $C^{(\ell)}$ can be expressed in two ways:

$$\begin{cases} C^{(1)} \approx \widetilde{\lambda} \mathbf{u}^{(1)}_{\natural} \otimes \mathbf{u}^{(2)}_{\natural} = \widetilde{\lambda} \widehat{\mathbf{u}}^{(1)}_{\sharp} \otimes \widehat{\mathbf{u}}^{(2)}_{\sharp}, \\ C^{(\ell)} \approx \widetilde{\lambda} \widehat{\mathbf{u}}^{(\ell)}_{\sharp} \otimes \mathbf{u}^{(\ell+1)}_{\natural} = \widetilde{\lambda} \mathbf{u}^{(\ell)}_{\sharp} \otimes \widehat{\mathbf{u}}^{(\ell+1)}_{\sharp}, \quad \ell = 2, \ldots, k-1, \\ C^{(k)} \approx \widetilde{\lambda} \widehat{\mathbf{u}}^{(1)}_{\sharp} \otimes \widehat{\mathbf{u}}^{(k)}_{\sharp} = \widetilde{\lambda} \mathbf{u}^{(1)}_{\sharp} \otimes \mathbf{u}^{(k)}_{\sharp}. \end{cases} \tag{24}$$

By the uniqueness of dominant singular vectors for $C^{(\ell)}$ for each $\ell \in [\![k]\!]$, the assertion (14) follows from (24). $\square$

The question is when the assumption imposed on T in Lemma 4.4 will hold. Specifically, let Ω denote the set of tensor $T \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_k}$ where there exists a convergent subsequence $\{\mathbf{u}^{(\ell)}_{[p_j]}\}$ such that the dominant singular value of the limit point $C^{(\ell)}$ of the corresponding $\{C^{(\ell)}_{[p_j]}\} \subset \mathbb{R}^{I_\ell \times I_{\ell+1}}$ is not simple. How large is the set Ω?

Consider the fact that symmetric matrices with multiply eigenvalues form an algebraic variety of codimension two [8]. For almost all matrices, therefore, the largest singular value is simple. For one particular limit point $C^{(\ell)}$ to have multiple dominant singular values, the subsequence $\{C^{(\ell)}_{[p_j]}\}$ that leads to it must approach arbitrarily close to that variety of matrices with multiple dominant singular values. But $\{C^{(\ell)}_{[p_j]}\}$ is defined in a specific algebraic way as in Line 4 of Algorithm 1. Backward tracing, the occurrence of $C^{(\ell)}$ with multiple dominant singular values depends on the set $\{\mathbf{u}^{(i)}_{[0]}, i \in [\![k]\!]\}$ of unit starting vectors, the particular subsequence $\{_{[p_j]}\}$ selected, and the underlying T. Any change of the starting vectors could alter the course of iteration. The choice of a different subsequence could lead to a different limit point. When both changes do not obliterate the appearance of dominant singular values, the tensor T itself must be something special. We thus conjecture that the set Ω should not be generic. Such a reasoning, of course, does not constitute a mathematical proof to support its genericity because we do not know of an analytic way to quantify a generic T. Thus, at the moment, we can only call it an assumption to be satisfied. Note that Lemma 4.4 is subsequence dependent. Its conclusion is with respect to only one particularly given convergent subsequence. The following condition is much stronger than what we need in Lemma 4.4.

Condition A. We say that a given order-k tensor T satisfies Condition A if for every convergent subsequences $\{\mathbf{u}^{(\ell)}_{[p_j]}\}$ generated by Algorithm 1 and the corresponding subsequence $\{C^{(\ell)}_{[p_j]}\}$, $\ell \in [\![k]\!]$, the dominant singular value $\widetilde{\lambda}$ of the limit point $C^{(\ell)} \in \mathbb{R}^{I_\ell \times I_{\ell+1}}$ defined in (12) is simple for all $\ell \in [\![k]\!]$.

We conjecture that Condition A holds for almost all order-k tensors. Even if not, keep in mind that it will be considered together with the Condition B which is generic and will be described below.

By the way the iteration is defined, and if Lemma 4.4 holds, any stationary point of Algorithm 1 necessarily satisfies the system of equations

$$T \circledast_\ell \left(\bigotimes_{i=1}^{\ell-1} \mathbf{u}^{(i)} \otimes \bigotimes_{i=\ell+1}^{k} \mathbf{u}^{(i)}\right) = \langle T, \bigotimes_{\ell=1}^{k} \mathbf{u}^{(\ell)} \rangle \mathbf{u}^{(\ell)}, \quad \ell \in [\![k]\!]. \tag{25}$$

The equation (25) is a polynomial system in the unknowns $(\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(k)}) \in \mathbb{R}^{I_1} \times \ldots \times \mathbb{R}^{I_k}$ with leading coefficients from entries of T. By the theory of parameter continuation [20, Theorem 7.1.1], we know that for almost all tensor $T \in \mathbb{C}^{I_1 \times \ldots \times I_k}$, except for an affine algebraic subset of codimension one in $\mathbb{C}^{I_1} \times \ldots \times \mathbb{C}^{I_k}$, the solutions to (25) are isolated. Together with the fact that $\mathbb{R}$ is dense in $\mathbb{C}$ under the Zariski topology, the real solutions are also isolated.

Condition B. We say that a given order k tensor T satisfies Condition B if the real solutions to the corresponding polynomial system (25) are isolated.

Lemma 4.5. *For almost all tensors T, the accumulation points of the sequence $\{\mathbf{u}_t\}$ generated by Algorithm 1 and Algorithm 2 are geometrically isolated.*

Finally, we are ready to claim our major result which serves as the theoretic basis complementing the SVD-based Algorithm 1. We say that a tensor $T \in \mathbb{R}^{I_1 \times \cdots \times I_k}$ is generic if it satisfies both generic conditions A and B. The non-generic tensors must reside on some algebraic varieties and, hence, are of measure zero.

Theorem 4.6. *For almost all order-k tensors T satisfying Condition A and for arbitrary starting points, the vector sequence $\{(\mathbf{u}^{(1)}_{[p]}, \ldots, \mathbf{u}^{(k)}_{[p]})\}$ generated by Algorithm 1 converges to a local maximizer of the generalized Rayleigh quotient $\lambda(\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(k)})$ defined in (2).*

Proof. Let $\{\mathbf{u}^{(\ell)}_{[p_j]}\}$ be any simultaneously convergent subsequences for $\ell \in [\![k]\!]$. By Lemma 4.3, the subsequences $\{\mathbf{u}^{(\ell)}_{[p_j+1]}\}$ also converges. Indeed, by Lemma 4.4, both subsequences converges to the same limit point for all $\ell \in [\![k]\!]$. Thus $\|\mathbf{u}^{(\ell)}_{[p_j+1]} - \mathbf{u}^{(\ell)}_{[p_j]}\| \to 0$. On the other hand, by Lemma 4.5 we assume that the limit point is geometrically isolated. The convergence of the entire sequence $\{\mathbf{u}^{(\ell)}_{[p_j]}\}$ to the same limit point follows from Lemma 2.3. ☐

4.2. Convergence of Algorithm 2

Now we argue the convergence of Algorithm 2 where $\boldsymbol{\beta}$ is changed randomly. For clarity, enumerate the column vectors at the end of Line 13 by $\left\{\mathbf{u}^{(1)}_t, \ldots, \mathbf{u}^{(k)}_t\right\}$. By construction, for $t \geq 1$, only two of these vectors are updated by the dominate left and right singular vector of C_t while others remain the same. Now we establish the following result.

Theorem 4.7. *For almost all order-k tensors T and arbitrary starting points, the vector sequence $\{(\mathbf{u}^{(1)}_t, \ldots, \mathbf{u}^{(k)}_t)\}$ generated by Algorithm 2 converges to a local maximizer of the generalized Rayleigh quotient defined in (2).*

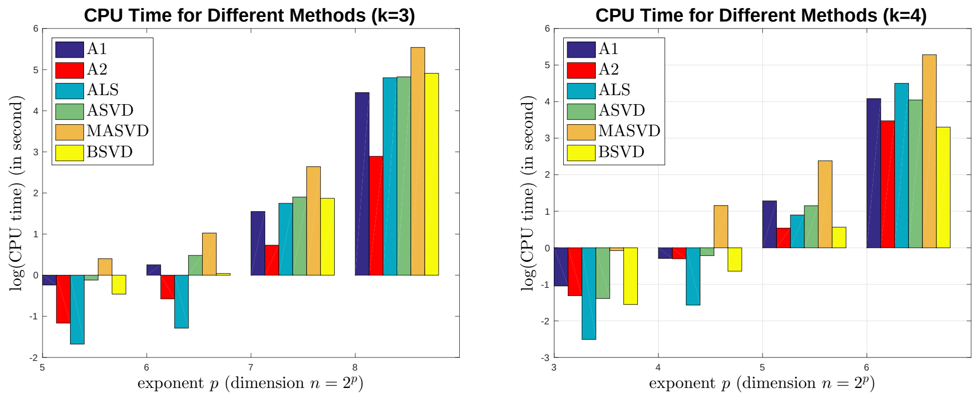

Figure 1: Comparison of CPU time among different methods.

Proof. Suppose that $\{\mathbf{u}_{t_i}^{(\ell)}\}$ is an arbitrary subsequence converging to $\overline{\mathbf{u}}^{(\ell)}$ simultaneously for $\ell \in [\![k]\!]$. Suppose also by Lemma 4.5 that the limit points $\overline{\mathbf{u}}^{(\ell)}$, $\ell \in [\![k]\!]$, are geometrically isolated. By construction, i.e., Line 7 in Algorithm 2, the subsequence $\{C_{t_i+1}^{(\ell)}\}$ of matrices converges. By continuity, the subsequence $\left\{\mathbf{u}_{t_i+1}^{(\ell)}\right\}$ must also converge to $\overline{\mathbf{u}}^{(\ell)}$. In particular, $\left\|\mathbf{u}_{t_i+1}^{(\ell)} - \mathbf{u}_{t_i}^{(\ell)}\right\| \to 0$. The condition in Lemma 2.3 therefore is satisfied. It follows that the whole sequence $\{\mathbf{u}_t^{(\ell)}\}$ converges to $\overline{\mathbf{u}}^{(\ell)}$. $\square$

We remark that in our recent work for symmetric tensors [12], we have also proposed an SVD-based algorithm by using a mechanism of random update similar to that adopted in Algorithm 2. The analysis there, in order to maintain symmetry, is much more involved than what we have shown above for non-symmetric tensors.

5. Numerical Experiments

The idea of updating two factors simultaneously by taking advantage of the two-in-one global optimization property of SVD is appealing. Thus we investigate to furnish a theoretic justification that the two variants of SVD-based methods described in this paper indeed converge. Two questions naturally arise. First, is there a significant difference in performance among different SVD-based algorithms? Second, is the SVD-based algorithm always superior to the conventional ALS method? Although rigorous numerical testing is not our objective in this paper, we carry out some preliminary experiments with the hope of partially satisfying our own curiosity.

Experiment 1. To our knowledge, there are at least five variants of SVD-based algorithms. These are the ASVD [11], the MASVD [11], the block SVD method [24], our Algorithm 1 and Algorithm 2. Without delving into the details to fine tune the programs, we implement all these methods based on what we understand from the literature. We are interesting in comparing the CPU time required for the iterates to meet the same stopping criteria – the iteration terminates automatically when the generalized Rayleigh quotients do not vary more than the tolerance 10^{-5} in three consecutive iterations. As a general reference point, also included is the performance of the conventional ALS method. To check the scalability, we consider a the case where all factors are of same dimension and vary the size of problem as $n = 2^p$ where we choose $p = 5, \ldots, 8$ when $k = 3$ and $p = 3, \ldots, 6$ when $k = 4$. Each case of p is repeatedly tested 20 times with random starting unit vectors. We plot the average as the running time in Figure 1. It should be pointed out that even from the same starting points, different methods may converge to different limit points because they involve different dynamics. Regardless, the comparison is based on the same starting points subject to the same stopping criteria for convergence.

We can almost explain about why the performance of the various algorithms is like what we have observed in Figure 1. For problems of modest sizes, e.g., $n = 2^6$ when $k = 3$ and $n = 2^4$ when $k = 4$, the cost of SVD computation outruns that of the high-order power method. Thus the ALS method uses less time. Even at these modest dimensions, however, note that it amounts to a full and dense tensor with approximately 2^{16} to 2^{18} entries. For odd order tensors, the block structure in the BSVD necessarily updates one vector via the ALS algorithm, which slows down its convergence. Thus we see that for order-3 tensors, our Algorithm 2 outperforms the BSVD when $p > 6$. For order-4 tensors, both Algorithm 2 and the BSVD method in [24] update two distinct vectors simultaneously, thus are about equally fast. On the other hand, the cyclic progression of Algorithm 1 updates each

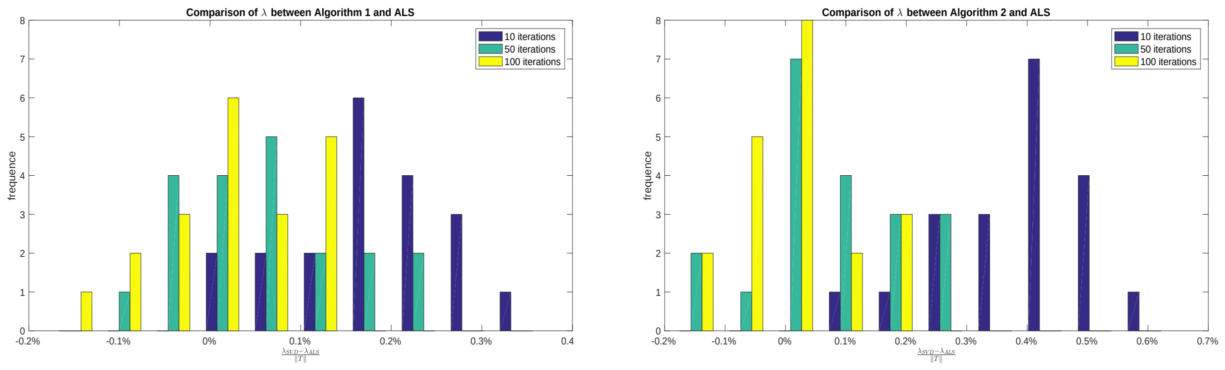

Figure 2: Comparison of quality between SVD and ALS.

factor twice but only the second time counts. So, it literately updates one factor a time. As such, it should always be less effective than Algorithm 2. The MASVD requires multiple ASVD calculation, so it is more expensive than ASVD. The ASVD checks through all possible permutations, so its performance is about the same as that of the Algorithm 1.

Experiment 2. In this experiment, we want to assess the quality of the iterates generated by the SVD-based method and the ALS method. To fix the idea, we consider a given order 4 tensor with $n = 2^6$. We use the ratio $\frac{\lambda_{SVD}-\lambda_{ALS}}{\|T\|}$ as the measurement of quality. The idea is that the larger the generalized Rayleigh quotient, the better the quality of the iteration. A positive ratio means that the SVD method is improving better than the ALS method, and vice versa. We perform the iteration 10, 50, and 100 times for each of the 20 randomly generated initial vector for the same tensor and measure the ratios. These numbers of iterations are far less than those taken for convergence to actually happen, but speak of an important trend. Plotted in Figure 2 are the histograms of ratios in 10 bins, when the comparison is made with respect to Algorithm 1 and Algorithm 2, respectively. Note that in the initial 10 iterations, the (dark blue) lobe of ratios leans toward the right of 0, indicating that the SVD method gives better improvement than the ALS method. However, when sufficiently many iterations are taken, the statistics in Figure 2 clearly shows that the (yellow) lobe shifts toward the left, indicating that the ALS method is gradually catching up the quality. In certain case, the ratio is negative, indicating that the ALS method might lead to a better local optimum eventually, although we have not seen the ultimate convergence yet. Comparing the two drawings in Figure 2, we also notice that Algorithm 2 generally keeps more positive ratios than Algorithm 1 does.

6. Conclusion

In contrast to the conventional ALS method that updates one factor a time for the rank-1 tensor approximation, the SVD-based method updates two factors simultaneously. This paper proves that the iteration by such a mechanism does converge for almost all tensors under Condition A. It is conjectured that tensors satisfying Condition A are generic, but an analytic proof is yet to be further investigated.

For large scale problems, numerical experiments suggest that the SVD-based methods do have the advantage of saving the computational time. On the other hand, partly due to the nonlinearity of the objective function, the SVD-based methods do not necessarily provide a better approximation in the long run.

References

[1] H. Attouch, J. Bolte, and B. F. Svaiter, *Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward-backward splitting, and regularized Gauss-Seidel methods*, Math. Program., 137 (2013), pp. 91–129.

[2] B. W. Bader and T. G. Kolda, *Algorithm 862: MATLAB tensor classes for fast algorithm prototyping*, ACM Trans. Math. Software, 32 (2006), pp. 635–653.

[3] J. Brachat, P. Comon, B. Mourrain, and E. Tsigaridas, *Symmetric tensor decomposition*, Linear Algebra Appl., 433 (2010), pp. 1851–1872.

[4] A. Bunse-Gerstner, R. Byers, V. Mehrmann, and N. K. Nichols, *Numerical computation of an analytic singular value decomposition of a matrix valued function*, Numer. Math., 60 (1991), pp. 1–40.

[5] R. Chill, *On the Łojasiewicz-Simon gradient inequality*, J. Funct. Anal., 201 (2003), pp. 572–601.

[6] L. X. Comon, P. and A. L. F. de Almeida, *Tensor decompositions, alternating least squares and other tales*, J. Chemometrics, 23 (2009), pp. 393–405.

[7] P. Comon, G. Golub, L.-H. Lim, and B. Mourrain, *Symmetric tensors and symmetric tensor rank*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1254–1279.

[8] M. Dana and K. D. Ikramov, *On the codimension of the variety of symmetric matrices with multiple eigenvalues*, Zap. Nauchn. Sem. S.-Peterburg. Otdel. Mat. Inst. Steklov. (POMI), 323 (2005), pp. 34–46, 224.

[9] L. De Lathauwer, B. De Moor, and J. Vandewalle, *On the best rank-1 and rank-(r_1, r_2,..., r_n) approximation of higher-order tensors*, SIAM journal on Matrix Analysis and Applications, 21 (2000), pp. 1324–1342.

[10] C. Eckart and G. Young, *The approximation of one matrix by another of lower rank*, Psychometrika, 1 (1936), pp. 211–218.

[11] S. Friedland, V. Mehrmann, R. Pajarola, and S. K. Suter, *On best rank one approximation of tensors*, Numer. Linear Algebra Appl., 20 (2013), pp. 942–955.

[12] Y. Guan, M. T. Chu, and D. Chu, *SVD-based algorithms for the best rank-1 approximation of a symmetric tensor*, preprint, North Carolina State University, 2017.

[13] M. Ishteva, P.-A. Absil, and P. Van Dooren, *Jacobi algorithm for the best low multilinear rank approximation of symmetric tensors*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 651–672.

[14] E. Kofidis and P. A. Regalia, *On the best rank-1 approximation of higher-order supersymmetric tensors*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 863–884.

[15] T. G. Kolda, *Numerical optimization for symmetric tensor decomposition*, Math. Program., 151 (2015), pp. 225–248.

[16] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500.

[17] S. Łojasiewicz, *Une propriété topologique des sous-ensembles analytiques réels*, in Les Équations aux Dérivées Partielles (Paris, 1962), Éditions du Centre National de la Recherche Scientifique, Paris, 1963, pp. 87–89.

[18] S. Łojasiewicz and M.-A. Zurro, *On the gradient inequality*, Bull. Polish Acad. Sci. Math., 47 (1999), pp. 143–145.

[19] J. J. Moré and D. C. Sorensen, *Computing a trust region step*, SIAM J. Sci. Statist. Comput., 4 (1983), pp. 553–572.

[20] A. J. Sommese and C. W. Wampler, II, *The numerical solution of systems of polynomials arising in engineering and science*, World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2005.

[21] A. Uschmajew, *Local convergence of the alternating least squares algorithm for canonical tensor approximation*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 639–652.

[22] L. Wang and M. T. Chu, *On the global convergence of the alternating least squares method for rank-one approximation to generic tensors*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 1058–1072.

[23] K. Wright, *Differential equations for the analytic singular value decomposition of a matrix*, Numer. Math., 3 (1992), pp. 283–295.

[24] Y. Yang, S. Hu, L. De Lathauwer, and J. A. Suykens, *Convergence study of block singular value maximization methods for rank-1 approximation to higher order tensors*, tech. report, Internal Report 16-149, ESAT-SISTA, KU Leuven, 2016.

[25] T. Zhang and G. H. Golub, *Rank-one approximation to high order tensors*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 534–550 (electronic).