ON FAST ALGORITHMS FOR ORTHOGONAL TUCKER DECOMPOSITION

Anh-Huy Phan[‡], *Andrzej Cichocki*[‡*], *Petr Tichavský*[•†]

[‡]Brain Science Institute, RIKEN, Wakoshi, Japan
[•]Institute of Information Theory and Automation, Prague, Czech Republic

ABSTRACT

We propose algorithms for Tucker tensor decomposition, which can avoid computing singular value decomposition or eigenvalue decomposition of large matrices as in the work-horse higher order orthogonal iteration (HOOI) algorithm. The novel algorithms require computational cost of $O(I^3R)$, which is cheaper than $O(I^3R + IR^4 + R^6)$ of HOOI for multilinear rank-(R, R, R) tensors of size $I \times I \times I$.

Index Terms— tensor decomposition, Tucker decomposition, orthogonality constraint, Cayley transform, Crank-Nicholson-like scheme

1. INTRODUCTION

Tucker tensor decomposition is the most well-known tensor decomposition together with the CANDECOMP/PARAFAC decomposition. This tensor decomposition is originally introduced in psychometrics [1, 2], and later has found many applications in numerous inter-disciplinary areas [3–9]. The Tucker decomposition is a multilinear extension of principle component analysis, or MultiLinear Singular Value Decomposition (MLSVD) [10]. The Tucker decomposition can compress data into a tensor of smaller size, which is represented by the core tensor, while its factor matrices span the subspace occupied by fibers of the data. Owing to these properties, the compressed data can be considered features for classification, recognition and clustering. For example, Tucker decomposition generalized the eigenfaces to tensorfaces in face recognition for different illuminations, poses, and expressions [5]. Tucker decompositions with various constraints such as orthogonality, non-negativity, discriminant using category information are suggested for classification, clustering [6]. Tucker decomposition (compression) is often used as a preprocessing for other tensor decompositions such as CANDECOMP/PARAFAC [11], decomposition into direct components (DEDICOM) [12, 13]. Tucker decomposition can be used for signal filtering [14], image denoising [15, 16].

In general, the Tucker decomposition is simply not unique. An unconstrained Tucker decomposition can always be converted to an orthogonal Tucker decomposition with an equivalent approximation error, which can be solved efficiently using the higher order SVD (HOSVD) algorithm or the Higher Order Orthogonal Iteration algorithm (HOOI) [17]. Hence, in practice, Tucker decomposition is always with orthogonality constraints. The HOSVD algorithm is a non-iterative algorithm, whose factor matrices are leading-left singular vectors of mode-n matricizations of the data tensor, whereas, factor matrices in the HOOI algorithm are iteratively updated in closed-form given by leading singular vectors of matricizations of the compressed data by all but one factor matrices. HOSVD is often used to initialize HOOI. This makes the HOOI algorithm becomes a "work-horse" algorithm for Tucker decomposition. In [18, 19], algorithms exploiting second-order information have also been proposed for Tucker decomposition. The algorithms require less number of iterations than HOOI, but they are much more expensive than HOOI. So far, there is not a comparable algorithm to HOOI in the sense of simplicity and efficiency.

In the HOOI algorithm, finding factor matrices through EVD of matrices of size $I_n \times I_n$ may become a weak point of this algorithm, when the data dimensions I_n are large. In this paper, we propose an algorithm based on the Crank-Nicholson-like scheme and the curvilinear search approach in [20],which has low computational cost, while its performance is comparable to HOOI.

Throughout the paper, we shall denote tensors by bold calligraphic letters, e.g., $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, matrices by bold capital letters, e.g., $\mathbf{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_R] \in \mathbb{R}^{I \times R}$, and vectors by bold italic letters, e.g., $\boldsymbol{a}_j$. The mode-n matricization of tensor $\boldsymbol{\mathcal{Y}}$ is a matrix $\mathbf{Y}_{(n)}$ of size $I_n \times (\prod_{k \neq n} I_k)$ [3]. The mode-n multiplication of a tensor $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ by a matrix $\mathbf{U} \in \mathbb{R}^{I_n \times R}$ is denoted by $\boldsymbol{\mathcal{Z}} = \boldsymbol{\mathcal{Y}} \times_n \mathbf{U} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times R \times I_{n+1} \times \cdots \times I_N}$ which is in mode-n matricization given by $\mathbf{Z}_{(n)} = \mathbf{U}\,\mathbf{Y}_{(n)}$.

The Tucker decomposition of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ can be written as

$$\boldsymbol{\mathcal{X}} \approx \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_n=1}^{R_N} g_{r_1 r_2 \ldots r_N} \, \boldsymbol{u}_{r_1}^{(1)} \circ \boldsymbol{u}_{r_2}^{(2)} \circ \cdots \circ \boldsymbol{u}_{r_N}^{(N)}, \quad (1)$$

where $\boldsymbol{\mathcal{G}} = [g_{r_1 r_2 \ldots r_N}] \in\in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$ and matrices $\mathbf{U}_n = [\boldsymbol{u}_1^{(n)}, \ldots, \boldsymbol{u}_{R_n}^{(n)}]$ are of full column rank.

2. THE PROPOSED ALGORITHMS

When the factor matrices in the Tucker decomposition $\mathbf{U}_n$ are constrained to be orthogonal matrices, the core tensor is ex-

[*]Also affiliated with Systems Research Institute, Polish Academy of Science, Poland.

[†]The work of P. Tichavsky was supported by the Czech Science Foundation through project No. 14–13713S.

pressed in closed-form as

$$\mathcal{G} = \mathcal{Y} \times_1 \mathbf{U}_1^T \times_2 \mathbf{U}_2^T \cdots \times_N \mathbf{U}_N^T. \quad (2)$$

Therefore, the decomposition can be achieved through minimizing the least squares cost function

$$\begin{aligned} D &= \frac{1}{2} \|\mathcal{Y} - \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \cdots \times_N \mathbf{U}_N\|_F^2 \\ &= \frac{1}{2} \left(\|\mathcal{Y}\|_F^2 - \|\mathcal{G}\|_F^2\right), \end{aligned} \quad (3)$$

which is equivalent to the problem

$$\begin{aligned} &\text{minimize} \quad \frac{-1}{2}\|\mathcal{Y} \times_1 \mathbf{U}_1^T \times_2 \mathbf{U}_2^T \cdots \times_N \mathbf{U}_N^T\|_F^2 \\ &\text{subject to} \quad \mathbf{U}_n^T \mathbf{U}_n = \mathbf{I}_{R_n}, \quad n = 1, \ldots, N, \end{aligned} \quad (4)$$

or to the optimization problem when expressing tensors by mode-n matricization

$$\begin{aligned} &\text{maximize} \quad \operatorname{tr}\{\mathbf{U}_n^T \mathbf{C}_n \mathbf{U}_n\} \\ &\text{subject to } \mathbf{U}_n^T \mathbf{U}_n = \mathbf{I}_{R_n}, n = 1, \ldots, N, \end{aligned} \quad (5)$$

where $\mathbf{C}_n = \mathbf{Y}_{(n)} \left(\bigotimes_{k \neq n} \mathbf{U}_k \mathbf{U}_k{}^T\right) \mathbf{Y}_{(n)}^T$ of size $I_n \times I_n$, $\otimes$ represents the Kronecker product. We can see that $\mathbf{U}_n$ comprise R_n-leading eigencomponents of the matrices $\mathbf{C}_n$, or R_n-leading left singular vectors of the matrix $\mathbf{Y}_{(n)} \left(\bigotimes_{k \neq n} \mathbf{U}_k\right)$. This motivates the Higher Order Orthogonal Iteration algorithm (HOOI) [17], an alternating algorithm which estimates $\mathbf{U}_n$ while fixing other factor matrices. The HOOI algorithm is the most widely-used algorithm for the Tucker decomposition. However, despite its simplicity and efficient implementation, the HOOI algorithm involves SVD or EVD of large matrices if I_n are relatively large. This algorithm costs $O(I^3R + IR^4 + R^6)$ [19] for order-3 tensor of size $I_1 = I_2 = I_3 = I$ and $R_1 = R_2 = R_3 = R$.

2.1. A Crank-Nicholson-like algorithm

In order to derive the new algorithm, we construct Lagrange functions of the cost function (4) with the orthogonality constraints,

$$L(\mathbf{U}_1, \ldots, \mathbf{U}_N, \mathbf{\Lambda}_1, \ldots, \mathbf{\Lambda}_N) = \\ \frac{-1}{2}\|\mathcal{Y} \times_1 \mathbf{U}_1^T \cdots \times_N \mathbf{U}_N^T\|_F^2 - \frac{1}{2}\sum_{n=1}^{N} \operatorname{tr}\left(\mathbf{\Lambda}_n \left(\mathbf{U}_n^T \mathbf{U}_n - \mathbf{I}_{R_n}\right)\right)$$

where $\mathbf{\Lambda}_n$ of size $R_n \times R_n$ are Lagrange multipliers. The gradient of the Lagrangian with respect to the $\mathbf{U}_n$ is given by

$$\mathbf{G}_n = -\mathbf{C}_n \mathbf{U}_n - \mathbf{U}_n \mathbf{\Lambda}_n . \quad (6)$$

Since $\mathbf{U}_n^T \mathbf{U}_n = \mathbf{I}_{R_n}$, by setting $\mathbf{G}_n$ to zero, we obtain

$$\mathbf{\Lambda}_n = -\mathbf{U}_n^T \mathbf{C}_n \mathbf{U}_n. \quad (7)$$

We replace $\mathbf{\Lambda}_n$ into the gradient (6) to derive a simple steepest descent update rules for $\mathbf{U}_n$ as

$$\mathbf{U}_n^{(k)} = \mathbf{U}_n^{(k-1)} - \eta\, \mathbf{G}_n^{(k-1)}, \quad (8)$$

where step size $\eta > 0$. In practice, the above update rule may converge slowly, and the new point may not preserve the orthogonality constraint. To this end, we apply the Crank-Nicholson-like scheme [20] to derive update rule for $\mathbf{U}_n$.

Since $\mathbf{G}_n^T \mathbf{U}_n = \mathbf{0}$, the gradient in (6) is equivalently expressed as

$$\begin{aligned} \mathbf{G}_n &= -\mathbf{C}_n \mathbf{U}_n + \mathbf{U}_n \mathbf{U}_n^T \mathbf{C}_n \mathbf{U}_n \\ &= (\mathbf{G}_n \mathbf{U}_n^T - \mathbf{U}_n \mathbf{G}_n^T) \mathbf{U}_n. \end{aligned} \quad (9)$$

After replacing the gradient in (8) by that in (9), and $\mathbf{U}_n^{(k-1)}$ by $\frac{1}{2}(\mathbf{U}_n^{(k-1)} + \mathbf{U}_n^{(k)})$, a new point at the iteration k $\mathbf{U}_n^{(k)}$ is generated following the Crank-Nicholson-like scheme [20]

$$\mathbf{U}_n^{(k)} = \mathbf{U}_n^{(k-1)} - \eta(\mathbf{G}_n^{(k-1)}\mathbf{U}_n^{(k-1)^T} - \mathbf{U}_n^{(k-1)}\mathbf{G}_n^{(k-1)^T})\frac{\left(\mathbf{U}_n^{(k-1)} + \mathbf{U}_n^{(k)}\right)}{2}$$

such that it preserves $\mathbf{U}_n^{(k)^T} \mathbf{U}_n^{(k)} = \mathbf{U}_n^{(k-1)^T} \mathbf{U}_n^{(k-1)}$. It can be shown that $\mathbf{U}_n^{(k)}$ can be updated using the following rule

$$\mathbf{U}_n \leftarrow \left(\mathbf{I}_{I_n} + \frac{\eta}{2}\mathbf{A}_n\right)^{-1} \left(\mathbf{I}_{I_n} - \frac{\eta}{2}\mathbf{A}_n\right) \mathbf{U}_n , \quad (10)$$

where $\mathbf{A}_n = \mathbf{G}_n \mathbf{U}_n^T - \mathbf{U}_n \mathbf{G}_n^T$, and $\mathbf{I}_J$ represents an identity matrix of size $J \times J$. For simplicity, the superscript which indicates the number of iteration has been suppressed in (10). Since $\mathbf{A}_n$ is a skew-symmetric matrix, $\mathbf{A}_n^T = -\mathbf{A}_n$, the matrix $\mathbf{Q} = (\mathbf{I} + \eta\mathbf{A}_n)^{-1} (\mathbf{I} - \eta\,\mathbf{A}_n)$ is orthogonal, $\mathbf{Q}^T \mathbf{Q} = \mathbf{I}$, and (10) is known as the Cayley transform, which can be derived when solving the optimization on Stiefel manifolds, see [21].

The update rule (10) can be efficiently implemented to avoid the matrix inverse $\left(\mathbf{I}_{I_n} + \frac{\eta}{2}\mathbf{A}_n\right)^{-1}$ for large I_n. For example, [20] and [21] propose a method which inverses matrices of size $2R_n \times 2R_n$

$$\mathbf{U}_n \leftarrow \mathbf{U}_n - \frac{\eta}{2}\,\mathbf{F}\left(\mathbf{I}_{2R_n} + \frac{\eta}{4}\mathbf{K}^T \mathbf{F}\right)^{-1} \mathbf{K}^T \mathbf{U}_n , \quad (11)$$

where $\mathbf{F} = [\mathbf{G}_n, \mathbf{U}_n]$ and $\mathbf{K} = [\mathbf{U}_n, -\mathbf{G}_n]$. We will show that $\mathbf{U}_n$ in (10) can be updated faster through inverse of matrices of size $R_n \times R_n$.

Lemma 1 (Fast update rule)**.** *Let $\mathbf{\Gamma}_n = \mathbf{G}_n^T \mathbf{G}_n$, the update rule (10) is equivalent to the following update rule*

$$\mathbf{U}_n \leftarrow -\mathbf{U}_n + (2\,\mathbf{U}_n - \eta\,\mathbf{G}_n)\left(\mathbf{I}_{R_n} + \frac{\eta^2}{4}\,\mathbf{\Gamma}_n\right)^{-1} . \quad (12)$$

Proof of Lemma 1 is given in Appendix. Since computation of $\mathbf{G}_n$ in (6) is of the same complexity as the computation of $\mathbf{C}_n$ in (5), the computational cost of (12) at each iteration is $O(I^3R)$ for order-3 multlinear rank-(R, R, R) tensors.

2.2. Choosing step size

The step size η in (12) at iteration-k can be chosen using the Barzilai-Borwein method [22, 23] defined as

$$\eta_k = \frac{\mathbf{s}_{k-1}^T\, \mathbf{s}_{k-1}}{\mathbf{s}_{k-1}^T\, \mathbf{y}_{k-1}}, \quad (13)$$

Algorithm 1: `Crank-Nicholson-like algorithm`

Input: Data tensor $\mathcal{Y}$: $(I_1 \times I_2 \times \cdots \times I_N)$, rank R
Output: A multilinear rank-$(R_1, R_2, \ldots, R_N)$ tensor $[\![\mathcal{G}; \{\mathbf{U}\}]\!]$
begin
  Initialize $\mathbf{U}_n$
  repeat
    for $n = 1, 2, \ldots, N$ **do**
      $\mathcal{X} = \mathcal{Y} \times_1 \mathbf{U}_1^T \cdots \times_{n-1} \mathbf{U}_{n-1}^T \times_{n+1} \mathbf{U}_{n+1}^T \cdots \times_N \mathbf{U}_N^T$
      $\mathbf{C}_n = \mathbf{X}_{(n)} \mathbf{X}_{(n)}^T$
      repeat
        $\mathbf{G}_n = -\mathbf{C}_n \mathbf{U}_n + \mathbf{U}_n \mathbf{U}_n^T \mathbf{C}_n \mathbf{U}_n$
        $\mathbf{U}_n \leftarrow -\mathbf{U}_n + (2\,\mathbf{U}_n - \eta\, \mathbf{G}_n) \left(\mathbf{I} + \frac{\eta^2}{4} \mathbf{G}_n^T \mathbf{G}_n\right)^{-1}$
      until *a stopping criterion is met*
  until *a stopping criterion is met*
  $\mathcal{G} = \mathcal{X} \times_N \mathbf{U}_N^T$

where $\boldsymbol{s}_{k-1} = \operatorname{vec}\left(\mathbf{U}_n^{(k)} - \mathbf{U}_n^{(k-1)}\right) = -\eta_{k-1} \operatorname{vec}\left(\mathbf{G}_n^{(k-1)}\right)$, $\boldsymbol{y}_{k-1} = \operatorname{vec}\left(\mathbf{G}_n^{(k)} - \mathbf{G}_n^{(k-1)}\right)$. The factor matrices $\mathbf{U}_n$ are iteratively updated in an inner loop with a small number of iterations.

An alternative method to select step size η is that we replace $\mathbf{U}_n$ in (5) by $(\mathbf{U}_n \mathbf{W}_n - \eta\, \mathbf{G}_n \boldsymbol{\Omega}_n)$ given in (12) where $\boldsymbol{\Omega}_n = (\mathbf{I} + \frac{\eta^2}{4} \boldsymbol{\Gamma}_n)^{-1}$, $\mathbf{W}_n = 2\,\boldsymbol{\Omega}_n - \mathbf{I}$, and construct the cost function to find η

$$\begin{aligned} D_n(\eta) &= \operatorname{tr}\{(\mathbf{W}_n^T \mathbf{U}_n^T - \eta\, \boldsymbol{\Omega}_n \mathbf{G}_n^T)\, \mathbf{C}_n\, (\mathbf{U}_n \mathbf{W}_n - \eta\, \mathbf{G}_n \boldsymbol{\Omega}_n)\} \\ &= \operatorname{tr}\{\mathbf{W}_n^T \mathbf{T}_1 \mathbf{W}_n - 2\eta\, \mathbf{W}_n^T (\mathbf{U}_n^T \mathbf{C}_n \mathbf{G}_n) \boldsymbol{\Omega}_n + \eta^2\, \boldsymbol{\Omega}_n \mathbf{T}_2 \boldsymbol{\Omega}_n\} \end{aligned}$$

where $\mathbf{T}_1 = \mathbf{U}^T \mathbf{C}_n \mathbf{U}_n$, $\mathbf{T}_2 = \mathbf{G}_n^T \mathbf{C}_n \mathbf{G}_n$. We have $\mathbf{W}_n^2 = \mathbf{I} + 4\,\boldsymbol{\Omega}_n(\boldsymbol{\Omega}_n - \mathbf{I}) = \mathbf{I} - \eta^2\, \boldsymbol{\Omega}_n^2 \boldsymbol{\Gamma}_n$, and $\mathbf{U}_n^T \mathbf{C}_n \mathbf{G}_n = -\boldsymbol{\Gamma}_n$ because $(\mathbf{G}_n^T + \mathbf{U}_n^T \mathbf{C}_n)\, \mathbf{G}_n = \mathbf{U}_n^T \mathbf{C}_n \mathbf{U}_n \mathbf{U}_n^T \mathbf{G}_n = 0$.

Let σ_r, $\boldsymbol{v}_r$ be eigenvalues and eigenvectors of $\boldsymbol{\Gamma}_n = \sum_{r=1}^{R_n} \sigma_r\, \boldsymbol{v}_r\, \boldsymbol{v}_r^T$. Hence, $\boldsymbol{\Omega}_n = \sum_{r=1}^{R_n} \frac{4}{4+\eta^2 \sigma_r} \boldsymbol{v}_r \boldsymbol{v}_r^T$, and $\mathbf{W}_n = \sum_{r=1}^{R_n} \frac{4-\eta^2 \sigma_r}{4+\eta^2 \sigma_r} \boldsymbol{v}_r \boldsymbol{v}_r^T$. The cost function $D_n(\eta)$ is then rewritten as

$$\begin{aligned} D_n(\eta) &= \operatorname{tr}\{\mathbf{T}_1 + \eta^2\, \boldsymbol{\Omega}_n^2 (\mathbf{T}_2 - \boldsymbol{\Gamma}_n \mathbf{T}_1) + 2\eta\, \mathbf{W}_n \boldsymbol{\Gamma}_n \boldsymbol{\Omega}_n\} \\ &= \operatorname{tr}\{\mathbf{T}_1\} + \sum_{r=1}^{R_n} \frac{16\eta^2 d_r}{(4 + \eta^2\, \sigma_r)^2} + \sum_{r=1}^{R_n} \frac{8\eta\, \sigma_r (4 - \eta^2 \sigma_r)}{(4 + \eta^2\, \sigma_r)^2} \end{aligned}$$

where $d_r = \boldsymbol{v}_r^T (\mathbf{T}_2 - \sigma_r \mathbf{T}_1)\, \boldsymbol{v}_r$. This is equivalent to the problem with $\eta \geq 0$

$$\text{maximize} \quad f(\eta) = \sum_{r=1}^{R_n} \frac{-\sigma_r^2\, \eta^3 + 2\, d_r\, \eta^2 + 4\sigma_r\, \eta}{(4 + \eta^2\, \sigma_r)^2}. \tag{14}$$

Assuming that $0 \leq \eta \ll 1$, the criterion in (14) can be approximated as

$$f(\eta) \approx \frac{-3}{16} a_3\, \eta^3 + \frac{1}{8} a_2\, \eta^2 + \frac{1}{4} a_1\, \eta \tag{15}$$

where $a_3 = \sum_r \sigma_r^2$, $a_2 = \sum_r d_r$, and $a_1 = \sum_r \sigma_r$. Computation of the optimum η thus can be done in closed form, but it involves eigendecomposition of $\boldsymbol{\Gamma}_n$. In practice, the improvement of convergence compared to the Barzilai-Borwein step size method is marginal. Although the update (12) preserves orthogonality of $\mathbf{U}_n$, in some cases, $\mathbf{U}_n$ may not be perfectly orthogonal due to problems of numerical precision and truncation error through number of iterations, $\mathbf{U}_n$ should be replaced by its orthogonal basis vectors $\mathbf{U}_n = \mathbf{A}\boldsymbol{\Sigma}\mathbf{B}^T$, $\mathbf{U}_n \leftarrow \mathbf{A}\mathbf{B}^T$.

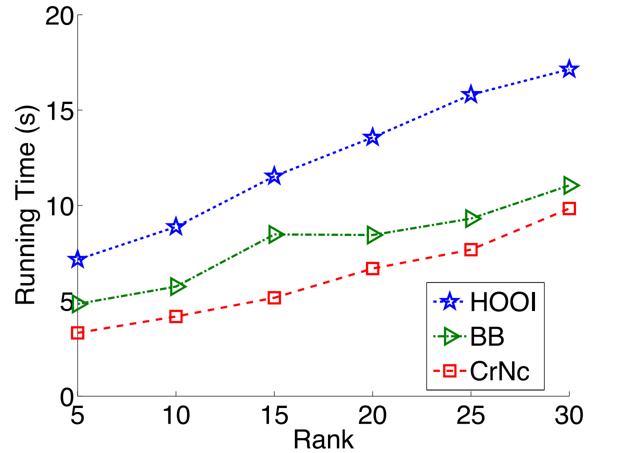


Fig. 1. Comparison of running times of algorithms when their approximation errors were 99.9% of of the final relative errors achieved by the HOOI algorithm.

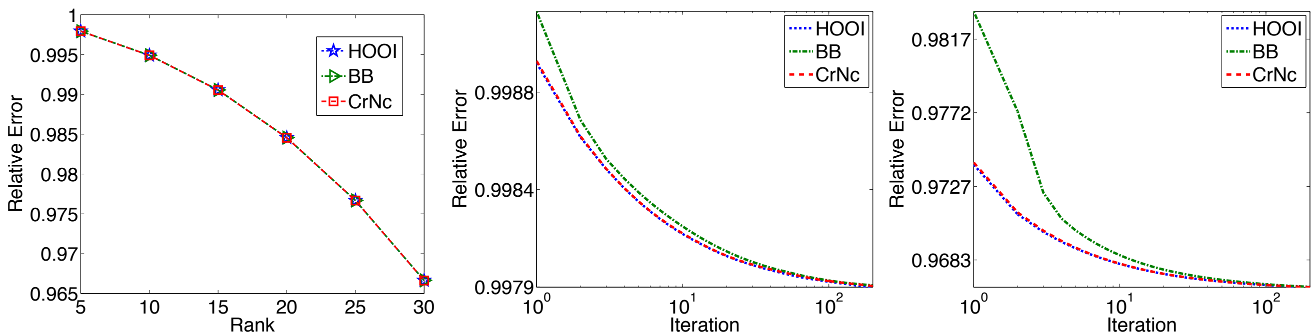
3. SIMULATIONS

In this section, the proposed algorithms are verified through decomposition of order-3 tensors of size $I \times I \times I$ randomly generated from the normal distribution with zero mean and unique standard deviation. Tensors were approximated by multilinear rank-(R, R, R) tensors with $R = 5, 10, \ldots, 30$. In addition to comparing the Crank-Nicholson-like (CrNc) algorithm with the HOOI algorithm, the simple steepest descent algorithm in (8) with step size chosen using the Barzilai-Borwein method (BB) was also considered in the simulations. The step size in CrNc was also chosen using the BB method. The relative approximation errors were used to evaluate performance of algorithms

$$\varepsilon = \frac{\|\mathcal{Y} - \mathcal{G} \times \{\mathbf{U}\}\|_F}{\|\mathcal{Y}\|_F} = \frac{\sqrt{\|\mathcal{Y}\|_F^2 - \|\mathcal{G}\|_F^2}}{\|\mathcal{Y}\|_F}. \tag{16}$$

The HOOI algorithm and the proposed algorithms were initialized by the same values using the HOSVD algorithm or orthogonal random matrices. All algorithms were set to run in 200 iterations. At the end of the iterative process, differences between the consecutive relative approximation errors were lower than 10^{-6}, indicating that algorithms converged to sufficient precision for comparison.

As seen in Fig. 2(a), all algorithms achieved almost the same relative approximation errors for different ranks. The



(a) Approximation error. (b) Approximation errors when $R = 5$. (c) Approximation errors when $R = 30$.

Fig. 2. Performance comparison of algorithms in decomposition of random tensors of size $100 \times 100 \times 100$.

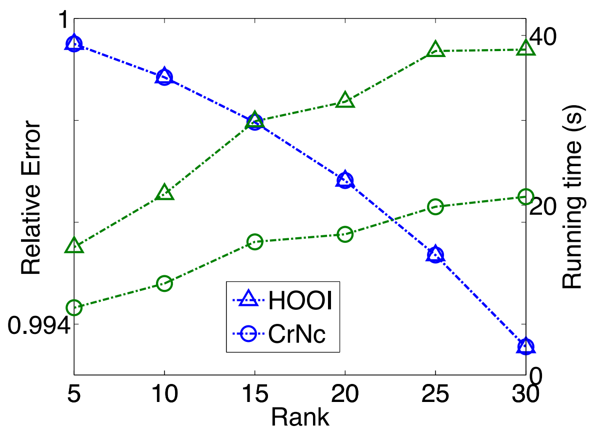


Fig. 3. Performance comparison of HOOI and CrNc in decomposition of tensors of size $200 \times 200 \times 200$.

results were averaged over 100 runs. In Figs. 2(b) and 2(c), we show the approximation errors as functions of the number of iterations. Approximation errors of the BB algorithm were far from those of the HOOI algorithm, but this algorithm approached the HOOI after 10-20 iterations. The CrNc algorithm was much better than the BB algorithm, and its accuracy was very close to that of the HOOI algorithm. Fig. 1 compares average running time of algorithms, when their approximation errors were approximately 99.9% of the final relative error of the HOOI algorithm achieved in 200 iterations. The CrNc algorithm were approximately 3-10 seconds faster than the HOOI algorithm.

In a second example, we decomposed tensors of size $200\times 200\times 200$ composed from randomly generated factor matrices of size $I \times R$, and random core tensors of size $R \times R \times R$. The tensors were corrupted by heavy additive Gaussian noise at signal-noise ratio SNR = -50 dB. Algorithms were run in 100 iterations, but could stop earlier when the consecutive relative error was lower than 10^{-7}. Fig. 3 shows results averaged over 200 independent runs, which indicate that CrNc and HOOI achieved comparable relative errors, but CrNc was faster than HOOI.

4. CONCLUSIONS

We have shown that the CrNc algorithm can quickly attain the performance of the HOOI algorithm after a few iterations, while the simple steepest descent algorithm can require higher number of iterations. Since inverses of $R_n \times R_n$ matrices are relatively cheap, the CrNc algorithm is more promising. The CrNc algorithm can be extended to decompose tensor with missing entries. The proposed algorithm is implemented in the Matlab package TENSORBOX which is available online at: `http://www.bsp.brain.riken.jp/~phan/tensorbox.php`.

Appendix: Proof of Lemma 1

Let $\mathbf{\Omega}_n = \left(\mathbf{I}_{R_n} + \frac{\eta^2}{4}\mathbf{\Gamma}_n\right)^{-1}$. Using the fact that $\mathbf{A}_n\mathbf{U}_n = \mathbf{G}_n$ and $\mathbf{A}_n\mathbf{G}_n = -\mathbf{U}_n\mathbf{\Gamma}_n$, it can be shown that

$$\begin{aligned}
\frac{\eta}{2}\mathbf{A}_n &= \mathbf{U}_n\mathbf{U}_n^T + \frac{\eta}{2}\mathbf{G}_n\mathbf{U}_n^T - \mathbf{U}_n\left(\mathbf{I} + \frac{\eta^2}{4}\mathbf{\Gamma}_n\right)\mathbf{\Omega}_n\,(\mathbf{U}_n + \frac{\eta}{2}\mathbf{G}_n)^T \\
&= \mathbf{U}_n\mathbf{U}_n^T + \frac{\eta}{2}\mathbf{A}_n\mathbf{U}_n\mathbf{U}_n^T - \frac{\eta^2}{4}\mathbf{U}_n\mathbf{\Gamma}_n\mathbf{\Omega}_n(\mathbf{U}_n + \frac{\eta}{2}\mathbf{G}_n)^T \\
&\quad - (\mathbf{U}_n - \frac{\eta}{2}\mathbf{G}_n + \frac{\eta}{2}\mathbf{G}_n)\,\mathbf{\Omega}_n\,(\mathbf{U}_n + \frac{\eta}{2}\mathbf{G}_n)^T \\
&= \left(\mathbf{I} + \frac{\eta}{2}\mathbf{A}_n\right)\mathbf{U}_n\mathbf{U}_n^T + \frac{\eta^2}{4}\mathbf{A}_n\mathbf{G}_n\mathbf{\Omega}_n(\mathbf{U}_n + \frac{\eta}{2}\mathbf{G}_n)^T \\
&\quad - (\mathbf{U}_n - \frac{\eta}{2}\mathbf{G}_n + \frac{\eta}{2}\mathbf{A}_n\mathbf{U}_n)\,\mathbf{\Omega}_n\,(\mathbf{U}_n + \frac{\eta}{2}\mathbf{G}_n)^T \\
&= \left(\mathbf{I} + \frac{\eta}{2}\mathbf{A}_n\right)\left(\mathbf{U}_n\mathbf{U}_n^T - (\mathbf{U}_n - \frac{\eta}{2}\mathbf{G}_n)\,\mathbf{\Omega}_n\,(\mathbf{U}_n + \frac{\eta}{2}\mathbf{G}_n)^T\right)
\end{aligned}$$

Since $\mathbf{G}_n^T\mathbf{U}_n = 0$ and $\mathbf{U}_n^T\mathbf{U}_n = \mathbf{I}$, from (10) we have

$$\begin{aligned}
\mathbf{U}_n &\leftarrow \left(\mathbf{I} - \eta\left(\mathbf{I} + \frac{\eta}{2}\mathbf{A}_n\right)^{-1}\mathbf{A}_n\right)\mathbf{U}_n \\
&= \left(\mathbf{I} - 2\left(\mathbf{U}_n\mathbf{U}_n^T - (\mathbf{U}_n - \frac{\eta}{2}\mathbf{G}_n)\,\mathbf{\Omega}_n\,(\mathbf{U}_n + \frac{\eta}{2}\mathbf{G}_n)^T\right)\right)\mathbf{U}_n \\
&= -\mathbf{U}_n + (2\,\mathbf{U}_n - \eta\,\mathbf{G}_n)\,\mathbf{\Omega}_n\,.
\end{aligned}$$

5. REFERENCES

[1] L.R. Tucker, "The extension of factor analysis to three-dimensional matrices," in *Contributions to Mathematical Psychology*, H. Gulliksen and N. Frederiksen, Eds., pp. 110–127. Holt, Rinehart and Winston, New York, 1964.

[2] L.R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, pp. 279–311, 1966.

[3] A. Cichocki, R. Zdunek, A.-H. Phan, and S. Amari, *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*, Wiley, Chichester, 2009.

[4] M. Haardt, F. Roemer, and G. Del Galdo, "Higher-order SVD based subspace estimation to improve the parameter estimation accuracy in multi-dimensional harmonic retrieval problems," *IEEE Trans. Signal Processing*, vol. 56, pp. 3198 – 3213, July 2008.

[5] M.A.O. Vasilescu and D. Terzopoulos, "Multilinear analysis of image ensembles: Tensorfaces," in *Proc. European Conf. on Computer Vision (ECCV)*, Copenhagen, Denmark, May 2002, vol. 2350, pp. 447–460.

[6] A.-H. Phan and A. Cichocki, "Tensor decompositions for feature extraction and classification of high dimensional datasets," *Nonlinear Theory and Its Applications, IEICE*, vol. 1, no. 1, pp. 37–68, 2010.

[7] H. Lu, K.N. Plataniotis, and A.N. Venetsanopoulos, "A survey of multilinear subspace learning for tensor data," *Pattern Recognition*, vol. 44, no. 7, pp. 1540–1551, 2011.

[8] M. Mørup, L.K. Hansen, and S.M. Arnfred, "Algorithms for sparse nonnegative Tucker decompositions," *Neural Computation*, vol. 20, pp. 2112–2131, 2008.

[9] A. Cichocki, D. P. Mandic, A.-H. Phan, C. Caifa, G. Zhou, Q. Zhao, and L. De Lathauwer, "Tensor decompositions for signal processing applications. from two-way to multiway component analysis," *IEEE Signal Processing Magazine*, , no. accepted, 2014.

[10] L. De Lathauwer, B. de Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM Journal of Matrix Analysis and Applications*, vol. 21, pp. 1253–1278, 2001.

[11] P. Comon, X. Luciani, and A. L. F. de Almeida, "Tensor decompositions, alternating least squares and other tales," *Journal of Chemometrics*, vol. 23, 2009.

[12] A.-H. Phan and A. Cichocki, "Extended HALS algorithm for nonnegative Tucker decomposition and its applications for multiway analysis and classification," *Neurocomputing*, vol. 74, no. 11, pp. 1956 – 1969, 2011, Selected papers from ICONIP 2009.

[13] A.-H. Phan, A. Cichocki, and Th. Vu-Dinh, "Nonnegative DEDICOM based on tensor decompositions for social networks exploration," *Australian Journal of Intelligent Information Processing Systems (ICONIP'10)*, vol. 12, no. 1, pp. 10–15, 2010.

[14] D. Muti and S. Bourennane, "Survey on tensor signal algebraic filtering," *Signal Processing*, vol. 87, no. 2, pp. 237–249, 2007.

[15] D. Letexier and S. Bourennane, "Adaptive flattening for multidimensional image restoration," *Signal Processing Letters, IEEE*, vol. 15, pp. 229–232, 2008.

[16] A. Rajwade, A. Rangarajan, and A. Banerjee, "Image denoising using the higher order singular value decomposition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 4, pp. 849–862, 2013.

[17] L. De Lathauwer, B. De Moor, and J. Vandewalle, "On the best rank-1 and rank-(R1,R2,. . . ,RN) approximation of higher-order tensors," *SIAM Journal of Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1324–1342, 2000.

[18] B. Savas and L-H. Lim, "Quasi-Newton methods on Grassmannians and multilinear approximations of tensors," *SIAM J. Scientific Computing*, vol. 32, no. 6, pp. 3352–3393, 2010.

[19] M. Ishteva, P.-A. Absil, S. Van Huffel, and L. De Lathauwer, "Best low multilinear rank approximation of higher-order tensors, based on the Riemannian trust-region scheme," *SIAM J. Matrix Analysis Applications*, vol. 32, no. 1, pp. 115–135, 2011.

[20] Z. Wen and W. Yin, "A feasible method for optimization with orthogonality constraints," *Mathematical Programming*, pp. 1–38, 2012.

[21] H. D. Tagare, "Notes on optimization on Stiefel manifolds," Tech. Rep., Yale University, 2011.

[22] J. Barzilai and J. M. Borwein, "Two-point step size gradient methods," *IMA Journal of Numerical Analysis*, vol. 8, no. 1, pp. 141–148, Jan. 1988.

[23] M Raydan, "On the Barzilai and Borwein choice of steplength for the gradient method," *IMA Journal of Numerical Analysis*, vol. 13, no. 3, pp. 321–326, 1993.