PARTITIONED HIERARCHICAL ALTERNATING LEAST SQUARES ALGORITHM FOR CP TENSOR DECOMPOSITION

Anh-Huy Phan[1], *Petr Tichavský*[2] *and Andrzej Cichocki*[1]

[1]Brain Science Institute, RIKEN, Wakoshi, Japan
[2]Institute of Information Theory and Automation,
P.O.Box 18, 182 08 Prague 8, Czech Republic

ABSTRACT

Canonical polyadic decomposition (CPD), also known as PARAFAC, is a representation of a given tensor as a sum of rank-one tensors. Traditional method for accomplishing CPD is the alternating least squares (ALS) algorithm. This algorithm is easy to implement with very low computational complexity per iteration. A disadvantage is that in difficult scenarios, where factor matrices in the decomposition contain nearly collinear columns, the number of iterations needed to achieve convergence might be very large. In this paper, we propose a modification of the algorithm which has similar complexity per iteration as ALS, but in difficult scenarios it needs a significantly lower number of iterations.

Index Terms— CANDECOMP/PARAFAC, tensor decomposition, partitioning ALS

1. INTRODUCTION

Canonical polyadic decomposition (CPD), also known as PARAFAC, is a representation of a given tensor as a sum of rank-one tensors. This tensor decomposition has found numerous applications in signal processing, including the identification of independent components in multivariate data through the decomposition of higher order cumulant tensors, retrieval of signals in CDMA telecommunications, extraction of hidden components from neural data, training a dictionary in supervised learning systems, image completion and signal tracking. For a review see [1–3] and references therein.

The traditional method for accomplishing CPD is the alternating least squares (ALS) algorithm. This algorithm is easy to implement with very low computational complexity per iteration. A disadvantage is that in difficult scenarios, where factor matrices in the decomposition contain nearly collinear columns, the number of iterations needed to achieve convergence might be very large [4, 5]. Therefore, several other methods of the CP decomposition have been proposed, e.g., enhanced line search (ELS) [6], rotational ALS [7], all-at-once optimisation algorithms [8–10], joint diagonalization based methods [11–15].

Most recently, the partitioned ALS algorithm has been proposed [16]. Different from the other alternating algorithms, which update one or all components in the same factor matrix, the PALS algorithm jointly updates components in different factor matrices. For a decomposition of tensor of size $R \times R \times R$ and of rank R, the complexity per iteration of ALS is $O(R^4)$, but that of the PALS is typically $O(R^6)$.

In this paper, we propose a modification of the PALS algorithm, which has similar complexity per iteration as ALS, $O(R^4)$, but in difficult scenarios it shows to need a significantly lower number of iterations. The algorithm is somewhat similar to the Hierarchical ALS algorithm [17].

The paper is organised as follows. First, we review ALS, HALS and PALS algorithms for easy reference. Second, we present the new algorithm called PHALS, and analyse its computational complexity. Simulation section and Conclusions section follow.

2. ALS, HALS AND PALS

We consider an approximation problem of a tensor $\mathcal{Y}$ of size $I_1 \times I_2 \times I_3$ by a tensor of rank-R. The following symbols "$\circ$", "$\otimes$", "$\odot$" and "$\circledast$" represent the outer product, the Kronecker product, the Khatri-Rao and element-wise Hadamard products, respectively.

2.1. Alternating least squares (ALS) algorithm

Fitting a tensor $\mathcal{Y}$ by a rank-R tensor can be achieved by minimising the Frobenius norm of the error given in a form as

$$\min \quad D \quad = \quad \|\mathcal{Y} - \sum_{r=1}^{R} \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r\|_F^2 . \qquad (1)$$

The simplest way to update components $\boldsymbol{a}_r$ is to rewrite the objective function as a quadratic form of the factor matrix $\mathbf{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_R]$, e.g.,

$$\min \quad D \quad = \quad \|\mathbf{Y}_{(1)} - \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T\|_F^2$$

where $\mathbf{Y}_{(1)}$ denotes mode-1 matricization of the tensor $\mathcal{Y}$. By this way, $\mathbf{A}$ can be updated in closed-form as

$$\mathbf{A} = \mathbf{Y}_{(1)}(\mathbf{C} \odot \mathbf{B})((\mathbf{C}^T\mathbf{C}) \circledast (\mathbf{B}^T\mathbf{B}))^{-1} .$$

This work was supported by the Czech Science Foundation through Project No. 17-00902S.

When columns of either one or two factor matrices $\mathbf{B}$ and $\mathbf{C}$ are highly collinear, the matrix $\mathbf{\Gamma}_2 = (\mathbf{C}^T\mathbf{C}) \circledast (\mathbf{B}^T\mathbf{B})$ becomes ill-conditioned. Hence the ALS update faces the problem of numerical instability, and the algorithm converges slowly. For such a case, updating all factor matrices using the second-order optimisation method becomes useful, as in the fast damped Gauss-Newton algorithm [9]. The algorithm costs $O(R^6)$ for order-3 tensor [9, 18], and might be computationally demanding when the rank R is high.

2.2. Hierarchical ALS

Alternatively, one can avoid updating all components in one factor matrix. For example, updating only one component per iteration as in the Hierarchical ALS algorithm [17]. In this algorithm, the global cost function (1) for the rank-R tensor is converted to local cost functions for R rank-1 tensors

$$\min \quad D \quad = \quad \| \operatorname{vec}(\mathcal{Y}_r) - (\boldsymbol{c}_r \otimes \boldsymbol{b}_r \otimes \mathbf{I}_{I_1})\, \boldsymbol{a}_r\|_F^2$$

where $r = 1, 2, \ldots, R$ and $\mathcal{Y}_r = \mathcal{Y} - \sum_{k \neq r} \boldsymbol{a}_k \circ \boldsymbol{b}_k \circ \boldsymbol{c}_k$. As a result, the component $\boldsymbol{a}_r$ can be updated in closed-form as

$$\boldsymbol{a}_r \quad = \quad \frac{1}{\gamma_r}(\mathbf{Y}_{(1)}(\boldsymbol{c}_r \otimes \boldsymbol{b}_r) - \mathbf{A}_{\bar{r}}\, \boldsymbol{\gamma}_{1,\bar{r}})\,, \qquad (2)$$

where $\gamma_r = (\boldsymbol{c}_r^T \boldsymbol{c}_r)(\boldsymbol{b}_r^T \boldsymbol{b}_r)$, and $\boldsymbol{\gamma}_{\bar{r}} = (\mathbf{B}_{\bar{r}}^T \boldsymbol{b}_r) \circledast (\mathbf{C}_{\bar{r}}^T \boldsymbol{c}_r)$, $\bar{r} = [1, \ldots, r-1, r+1, \ldots, R]$. The algorithm need not matrix inversions in the ALS algorithm, and is particularly useful for the constrained CPD, when closed-form update for the factor matrix does not exist, e.g., for the nonnegative CPD [17]. However, the algorithm requires more iterations, and therefore, it becomes less efficient for the non-constrained CPD with highly collinear components.

2.3. Partitioned ALS

Another method is the Partitioned ALS algorithm (PALS) [16], which jointly updates R components of different factor matrices, $\mathbf{A}_{\boldsymbol{r}_1}$, $\mathbf{B}_{\boldsymbol{r}_2}$ and $\mathbf{C}_{\boldsymbol{r}_3}$, where $\boldsymbol{r}_1$, $\boldsymbol{r}_2$ and $\boldsymbol{r}_3 = \{1, \ldots, R\} \setminus \{\boldsymbol{r}_1, \boldsymbol{r}_2\}$ are three disjoint index subsets. Note that one or two sets of $\boldsymbol{r}_1$, $\boldsymbol{r}_2$, and $\boldsymbol{r}_3$ can be empty. The key observation of the PALS technique is that the tensor is a linear function of $(\mathbf{A}_{\boldsymbol{r}_1}, \mathbf{B}_{\boldsymbol{r}_2}, \mathbf{C}_{\boldsymbol{r}_3})$ provided that the other part of the factor matrices are fixed. The algorithm may proceed by a cyclic update with respect to $(\mathbf{A}_{\boldsymbol{r}_1}, \mathbf{B}_{\boldsymbol{r}_2}, \mathbf{C}_{\boldsymbol{r}_3})$, $(\mathbf{A}_{\boldsymbol{r}_3}, \mathbf{B}_{\boldsymbol{r}_1}, \mathbf{C}_{\boldsymbol{r}_2})$ and $(\mathbf{A}_{\boldsymbol{r}_2}, \mathbf{B}_{\boldsymbol{r}_3}, \mathbf{C}_{\boldsymbol{r}_1})$. The partitioning $(\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3)$ is taken at random, after each cycle above. Disadvantage of the algorithm is, that if the partitioning is balanced, i.e., when the number of elements in $\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3$ are approximately equal ($\approx R/3$), complexity of each iteration is $O(R^6)$. The way to reduce the complexity is to consider unbalanced partitioning only.

3. JOINT UPDATE RULES OF COMPONENTS

In this paper, our aim is to improve the HALS algorithm using the update method of PALS. The basic idea behind our proposed algorithm is that while the component of a factor matrix, e.g., $\boldsymbol{a}_r$, is jointly updated with components in another factor matrix, e.g., $\mathbf{B}_{\bar{r}}$. We will show that the update rule of R joint components has a similar cost as that of the ALS algorithm.

For a column index r in $[1, R]$, we will derive update rules for the r-th column $\boldsymbol{a}_r = \mathbf{A}(:, r)$ and $(R-1)$ columns $\mathbf{B}_{\bar{r}} = \mathbf{B}(:, \bar{r})$, while $\mathbf{C}$, $\mathbf{A}_{\bar{r}}$ and $\boldsymbol{b}_r$ are fixed. This is a particular case of the Partitioning ALS (PALS) [16] with a partition of $r_1 = 1$, $\boldsymbol{r}_2 = [2, \ldots, R]$, and $r_3 = \emptyset$. We rewrite the objective function in (1) as

$$D \quad = \quad \| \operatorname{vec}(\mathcal{Y}) - (\boldsymbol{c}_r \otimes \boldsymbol{b}_r \otimes \mathbf{I}_{I_1})\, \boldsymbol{a}_r - \sum_{k \in \bar{r}} (\boldsymbol{c}_k \otimes \mathbf{I}_{I_2} \otimes \boldsymbol{a}_k)\, \boldsymbol{b}_k\|_F^2\,,$$

i.e., in quadratic form of the interest variables $\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{a}_r \\ \operatorname{vec}(\mathbf{B}_{\bar{r}}) \end{bmatrix}$ $\in \mathbb{R}^{I_1 + (R-1)I_2}$. Hence, $\boldsymbol{\theta}$ can be updated in closed-form as

$$\boldsymbol{\theta} \quad = \quad \mathbf{H}^{-1}\, \boldsymbol{g}\,, \qquad (3)$$

where the Hessian $\mathbf{H} = \mathbf{J}^T \mathbf{J}$, the Jacobian is given by

$$\begin{aligned} \mathbf{J} \quad &= \quad \left[\frac{\partial \operatorname{vec}(\mathcal{Y})}{\partial \boldsymbol{a}_r}, \frac{\partial \operatorname{vec}(\mathcal{Y})}{\partial \mathbf{B}_{\bar{r}}}\right] \\ &= \quad [(\boldsymbol{c}_r \otimes \boldsymbol{b}_r \otimes \mathbf{I}_{I_1}), \ldots, (\boldsymbol{c}_k \otimes \mathbf{I}_{I_2} \otimes \boldsymbol{a}_k) \ldots] \end{aligned}$$

and $\boldsymbol{g} = \mathbf{J}^T \operatorname{vec}(\mathcal{Y})$ comprises gradients of the objective function with respect to $\boldsymbol{a}_r$ and $\mathbf{B}_{\bar{r}}$, respectively computed as,

$$\boldsymbol{g}_1 = \frac{\partial D}{\partial \boldsymbol{a}_r} = \mathcal{Y}_{(1)}\,(\boldsymbol{c}_r \otimes \boldsymbol{b}_r),\ \mathbf{G}_2 = \frac{\partial D}{\partial \mathbf{B}_{\bar{r}}} = \mathcal{Y}_{(2)}\,(\mathbf{C}_{\bar{r}} \odot \mathbf{A}_{\bar{r}}). \quad (4)$$

The Hessian $\mathbf{H}$ is of size $(I_1 + (R-1)I_2) \times (I_1 + (R-1)I_2)$, and its inverse may be expensive when the tensor size is large. For a simple case when $I_1 = I_2 = I$, its inverse has a cost of order $O(I^3 R^3)$, which is much more expensive than the cost of the ALS update to invert matrices of size $R \times R$. Fortunately, $\mathbf{H}$ can be expressed in an adjustment form of rank-$2(R-1)$, and thereby R joint components $\boldsymbol{a}_r$ and $\mathbf{B}_{\bar{r}}$ can be updated with a low computational cost.

Theorem 1 (Joint update rules for $\boldsymbol{a}_r$ and $\mathbf{B}_{\bar{r}}$)**.**

$$\boldsymbol{a}_r \quad = \quad \frac{1}{\gamma_r}\,(\boldsymbol{g}_1 - \mathbf{A}_{\bar{r}}\, \mathbf{D}_r \boldsymbol{u})\;, \qquad (5)$$

$$\mathbf{B}_{\bar{r}} \quad = \quad \left(\mathbf{I}_{I_2} - \frac{\boldsymbol{b}_r\, \boldsymbol{b}_r^T}{\boldsymbol{b}_r^T \boldsymbol{b}_r}\right) \mathbf{G}_2\, \mathbf{\Gamma}_{\bar{r}}^{-1} + \frac{\boldsymbol{b}_r\, \boldsymbol{u}^T}{\boldsymbol{b}_r^T \boldsymbol{b}_r}\,, \qquad (6)$$

where

$$\boldsymbol{u} = \left(\mathbf{D}_r \mathbf{\Omega}_{\bar{r}} \mathbf{D}_r - (\boldsymbol{c}_r^T \boldsymbol{c}_r)\, \mathbf{\Gamma}_{\bar{r}}\right)^{-1} \left(\mathbf{D}_r \mathbf{A}_{\bar{r}}^T \boldsymbol{g}_1 - (\boldsymbol{c}_r^T \boldsymbol{c}_r)\, \mathbf{G}_2^T \boldsymbol{b}_r\right)$$

$\mathbf{\Omega}_{\bar{r}} = \mathbf{A}_{\bar{r}}^T \mathbf{A}_{\bar{r}}$, $\mathbf{\Gamma}_{\bar{r}} = (\mathbf{A}_{\bar{r}}^T \mathbf{A}_{\bar{r}}) \circledast (\mathbf{C}_{\bar{r}}^T \mathbf{C}_{\bar{r}})$, *and* $\mathbf{D}_r = \operatorname{diag}(\mathbf{C}_{\bar{r}}^T \boldsymbol{c}_r)$.

The proof is in Appendix. Note that due to scaling ambiguity, we can always normalise components $\boldsymbol{b}_r$, $\mathbf{A}_{\bar{r}}$ and $\mathbf{C}$ to unit-length vectors. Hence, $\boldsymbol{b}_r^T \boldsymbol{b}_r = \boldsymbol{c}_r^T \boldsymbol{c}_r = 1$, and $\gamma_r = 1$. This will simplify the above update rules. The update rules in Theorem 1 inverts two matrices of size $(R-1) \times (R-1)$. When the rank R exceeds the dimension I_1, the update rule for $\boldsymbol{a}_r$ and $\mathbf{B}_{\bar{r}}$ can be replaced by the ones in Theorem 2, which requires inversions of a matrix $\mathbf{\Gamma}_{\bar{r}}$ of size $(R-1) \times (R-1)$ and a matrix $\mathbf{\Delta}$ of size $(I_1 \times I_1)$.

Algorithm 1: PHALS

```
Input: Data tensor $\mathcal{Y}$: $(I_1 \times I_2 \times I_3)$, and rank $R$
Output: $\mathcal{X} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ of rank $R$ such that $\min \|\mathcal{Y} - \mathcal{X}\|_F^2$
begin
    Initialize $\mathcal{X}$
    repeat
        for $r = 1, 2, \ldots, R$ do
            Joint update $\boldsymbol{a}_r$ and $\mathbf{B}_{\bar{r}}$
            Joint update $\boldsymbol{b}_r$ and $\mathbf{C}_{\bar{r}}$
            Joint update $\boldsymbol{c}_r$ and $\mathbf{A}_{\bar{r}}$
    until a stopping criterion is met
```

Theorem 2.

$$\boldsymbol{a}_r \leftarrow \boldsymbol{\Delta}^{-1}\left(\boldsymbol{g}_1 - \mathbf{A}_{\bar{r}}\mathbf{D}_r\boldsymbol{\Gamma}_{\bar{r}}^{-1}\mathbf{G}_2^T\boldsymbol{b}_r\right) \tag{7}$$

$$\mathbf{B}_{\bar{r}} \leftarrow \left(\mathbf{G}_2 - \boldsymbol{b}_r\left((\boldsymbol{c}_r^T\mathbf{C}_{\bar{r}}) \circledast (\boldsymbol{a}_r^T\mathbf{A}_{\bar{r}})\right)\right)\boldsymbol{\Gamma}_{\bar{r}}^{-1} \tag{8}$$

where $\boldsymbol{\Delta} = \gamma_r\mathbf{I}_{I_1} - (\boldsymbol{b}_r^T\boldsymbol{b}_r)\,\mathbf{A}_{\bar{r}}\mathbf{D}_r\boldsymbol{\Gamma}_{\bar{r}}^{-1}\mathbf{D}_r\mathbf{A}_{\bar{r}}^T$.

The update rule for $\boldsymbol{a}_r$ in (5) is in similar form as the HALS update in (2), but different by the term $\boldsymbol{u}$. This is considered an improvement of the HALS update of $\boldsymbol{a}_r$, but also an improvement of the ALS update for $\mathbf{B}$ or $\mathbf{C}$ by not updating their all columns.

Computation of the gradients $\boldsymbol{g}_1$ and $\mathbf{G}_2$ has a cost of $\mathcal{O}(RI_1I_2I_3)$, which is similar to the update of R components in the ALS algorithm. Besides this, PHALS inverts two matrices of size $(R-1) \times (R-1)$, while ALS inverts a matrix of size $R \times R$. Hence, the computational costs of the proposed algorithm and ALS are of the same order.

The proposed algorithm is briefly listed in Algorithm 1. At each inner iteration for r running from 1 to R, the algorithm jointly updates the component $\boldsymbol{a}_r$ and $\mathbf{B}_{\bar{r}}$, and update $\boldsymbol{b}_r$ and $\mathbf{C}_{\bar{r}}$, then $\boldsymbol{c}_r$ and $\mathbf{A}_{\bar{r}}$. This completes one update round of the proposed algorithm.

4. SIMULATIONS

4.1. Decomposition of tensors of highly collinear components

In this section, we will illustrate efficiency of the proposed method. The considered tensors were of size $I \times I \times I$ and rank-R, where $I = 5R$ or $I = 10R$ and $R = 6$, 10 or 15. Components of the second and third factor matrices were highly collinear, with $\boldsymbol{b}_r^T\boldsymbol{b}_s = \boldsymbol{c}_r^T\boldsymbol{c}_s = 0.95$ for all $r \neq s$, whereas $\boldsymbol{a}_r^T\boldsymbol{a}_s = 0.9$. All components are unit-length vectors. The factor matrices with specific correlation coefficients were generated using the subroutine "gen_matrix" in the TENSORBOX [19] (see generation of such factor matrices in Appendix F in [20]). The Gaussian noise was added into the tensor $\mathcal{Y}$ so that the Signal-Noise-Ratios SNR = 20 and 50 dB.

We compare the proposed algorithm with the HALS, ALS, and ALS with enhanced or exact line search algorithms (ALS+ELS). Initial values were generated using the DTLD algorithm [21]. The PHALS algorithm stopped when the relative errors $\varepsilon = \frac{\|\mathcal{Y} - \widehat{\mathcal{Y}}\|_F}{\|\mathcal{Y}\|_F}$ were lower than 10^{-8}, while the other algorithms stopped when their approximation errors reached the last approximation error achieved by PHALS.

Table 1. Comparison of execution times in second and the number of iterations of algorithms considered in Example 4.1.

	$(R, I =)$	(6,30)	(10,50)	(15,75)	(6,60)	(10,100)	(15,150)
Execution time (s)	SNR = 20 dB						
	HALS	26.0	58.2	131	54.1	309	656
	ALS	1.5	3.1	10.6	3.7	22.3	60.1
	PHALS	0.61	1.5	6.1	2.7	16.4	37.9
	ALS+ELS	0.41	2.4	16.4	7.3	56.5	113
	PHALS+ELS	0.38	1.0	4.5	2.6	14.2	29.9
	SNR = 50 dB						
	HALS	26.6	77.4	268	30.6	307	919
	ALS	1.4	3.4	12.6	2.5	22.4	66.3
	PHALS	0.54	1.6	7.4	1.7	15.8	42.5
	ALS+ELS	0.24	1.6	13.4	3.0	32.7	76.6
	PHALS+ELS	0.23	0.98	5.0	1.1	11.1	29.0
Number of Iterations	SNR = 20 dB						
	HALS	15521	21941	21317	15025	20971	20362
	ALS	1524	1886	2170	1299	1701	1931
	PHALS	578	701	794	470	615	691
	ALS+ELS	391	517	622	328	443	525
	PHALS+ELS	338	405	462	271	357	412
	SNR = 50 dB						
	HALS	15915	29385	44190	8701	21169	29095
	ALS	1445	2062	2584	899	1697	2173
	PHALS	520	756	955	324	621	803
	ALS+ELS	226	359	503	142	272	370
	PHALS+ELS	207	379	506	128	287	413

In Table 1, we compare execution times and the number of iterations of the considered algorithms, which were averaged over 100 independent runs for each test cases. Here, by "one iteration" we mean that the algorithm completes the updates of all parameters, i.e., $3R$ components. The ALS+ELS algorithm updates all parameters in one iteration using the ALS updates, then using the ELS update in another iteration. The PHALS algorithm updates all components for each index r running from 1 to R, until convergence.

For most the test cases, especially when the rank was high $R = 10, 15$ and the tensor size was large $I \geq 50$, the HALS algorithm executed a huge number of iterations, and was the most time consuming algorithm. The ALS algorithm required on average 2.65 times more number of iterations than PHALS, and it was two times slower than PHALS. When using with ELS, the ALS algorithm converged sooner only for the test cases with relatively small rank and tensor sizes, e.g., $(R, I) = (6, 30)$ or $(10, 50)$. When the tensor size was 60 or larger, ALS+ELS achieved the target approximation errors faster with a smaller number of iterations than ALS. However, this combined algorithm was even slower than ALS. This is because of high computational cost of the line-search.

The results indicate that PHALS was only slower than ALS+ELS for decomposition of tensors of small size and low rank, e.g., $(R, I) = (6, 30)$, despite requiring more number of iterations then ALS+ELS. An important observation is that

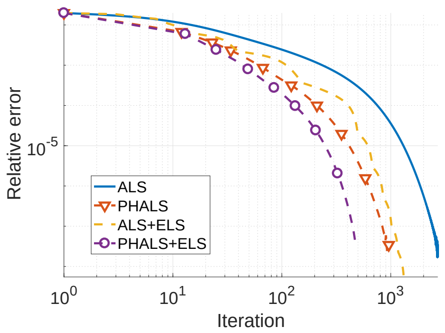

Fig. 1. Convergence behaviour of the considered algorithms in decomposition of the multiplication tensor of size $6 \times 6 \times 4$ of rank-11.

PHALS+ELS was even better than PHALS, and became comparable with ALS+ELS when tensors was of small size. For the test case $I = 10R$, PHALS+ELS was 2.8 times faster than ALS+ELS when SNR = 50 dB, and 3.5 times when SNR = 20 dB.

4.2. Decomposition of tensor for the matrix multiplications $(2 \times 3) \times (3 \times 2)$

In this example, we consider a tensor $\mathcal{Y}$ of size $6\times6\times4$ which is associated with the multiplication of two matrices $\mathbf{A}$ and $\mathbf{B}$ of size (2×3) and (3×2), respectively. The tensor contains only zeros and ones, and obeys

$$\mathrm{vec}(\mathbf{AB}) = \mathcal{Y} \times_1 \mathrm{vec}\left(\mathbf{A}^T\right)^T \times_2 \mathrm{vec}\left(\mathbf{B}^T\right)^T .$$

This tensor $\mathcal{Y}$ has rank-11 [22], i.e. exceeds the tensor sizes. The relative errors of the considered algorithms are compared in Fig. 1. The results indicate that the ALS algorithm needed 2702 iterations, three times more than the PHALS algorithm.

5. CONCLUSIONS

We have presented a novel alternating least squares algorithm for CPD, which is considered a particular case of the PALS algorithm [16], but has much lower complexity than the PHALS. The proposed algorithm inverts matrices of size $(R-1)\times(R-1)$ per iteration. Hence, it has at most as the same computational cost of the ALS algorithm, which needs to invert matrices of size $R \times R$. Simulation results show that our algorithm needs a smaller number of iterations than the ALS algorithm in some difficult tensor decompositions. Moreover, the convergence of PHALS can be improved when using with ELS. The algorithm can be extended to constrained CP decompositions.

APPENDIX: Proof of Theorem 1

Proof. From the Hessian $\mathbf{H} = \mathbf{J}^T\mathbf{J}$, it can be shown that $\mathbf{H}$ can be expressed in a low-rank adjustment form as

$$\mathbf{H} \;=\; \mathbf{D} + \mathbf{Z}\,\mathbf{K}\,\mathbf{Z}^T \qquad (9)$$

where

$$\mathbf{D} = \begin{bmatrix} \gamma_r\,\mathbf{I}_{I_1} & \\ & \boldsymbol{\Gamma}_{\bar{r}} \otimes \mathbf{I}_{I_2} \end{bmatrix}, \quad \mathbf{Z} = \begin{bmatrix} \mathbf{A}_{\bar{r}} & \\ & \mathbf{I}_{R-1} \otimes \boldsymbol{b}_r \end{bmatrix},$$
$$\mathbf{K} = \begin{bmatrix} & \mathrm{diag}(\mathbf{C}_{\bar{r}}^T\,\boldsymbol{c}_r) \\ \mathrm{diag}(\mathbf{C}_{\bar{r}}^T\,\boldsymbol{c}_r) & \end{bmatrix} \in \mathbb{R}^{2(R-1)\times 2(R-1)}.$$

The above expression can also be derived by simply eliminating columns and rows in the full Hessian for all parameters stated in Theorem 4.2 in [9].

Following the matrix inversion lemma (Woodbury identity), the inversion of $\mathbf{H}$ can be proceeded as

$$\mathbf{H}^{-1} = \mathbf{D}^{-1} - \mathbf{D}^{-1}\mathbf{Z}\,(\mathbf{K}^{-1} + \mathbf{Z}^T\mathbf{D}^{-1}\mathbf{Z})^{-1}\,\mathbf{Z}^T\mathbf{D}^{-1}\,.$$

Hence, the update rule for the parameters $\boldsymbol{\theta}$ is given by

$$\begin{aligned} \boldsymbol{\theta} &= \mathbf{H}^{-1}\boldsymbol{g} \\ &= \mathbf{D}^{-1}\boldsymbol{g} - (\mathbf{D}^{-1}\mathbf{Z})\,\boldsymbol{\Psi}^{-1}\,(\mathbf{Z}^T\mathbf{D}^{-1}\boldsymbol{g}) \end{aligned} \qquad (10)$$

where $\boldsymbol{\Psi} = \mathbf{K}^{-1} + \mathbf{Z}^T\mathbf{D}^{-1}\mathbf{Z}$ is a matrix of size $2(R-1)\times2(R-1)$. Inverse of the matrix $\mathbf{K}$ is given by

$$\mathbf{K}^{-1} = \begin{bmatrix} & \mathbf{D}_r^{-1} \\ \mathbf{D}_r^{-1} & \end{bmatrix}.$$

Hence, the matrix $\boldsymbol{\Psi}$ has inversion in block form as

$$\begin{aligned} \boldsymbol{\Psi}^{-1} &= \begin{bmatrix} \frac{1}{\gamma_r}\boldsymbol{\Omega}_{\bar{r}} & \mathbf{D}_r^{-1} \\ \mathbf{D}_r^{-1} & (\boldsymbol{b}_r^T\boldsymbol{b}_r)\,\boldsymbol{\Gamma}_{\bar{r}}^{-1} \end{bmatrix}^{-1} \\ &= \begin{bmatrix} \gamma_r\mathbf{D}_r\mathbf{F}\mathbf{D}_r & -(\boldsymbol{c}_r^T\boldsymbol{c}_r)\mathbf{D}_r\mathbf{F}\,\boldsymbol{\Gamma}_{\bar{r}} \\ -(\boldsymbol{c}_r^T\boldsymbol{c}_r)\boldsymbol{\Gamma}_{\bar{r}}\,\mathbf{D}_r^{-1}\,\mathbf{D}_r\mathbf{F}\mathbf{D}_r & \frac{1}{\boldsymbol{b}_r^T\boldsymbol{b}_r}\boldsymbol{\Gamma}_{\bar{r}} + \frac{\boldsymbol{c}_r^T\boldsymbol{c}_r}{\boldsymbol{b}_r^T\boldsymbol{b}_r}\boldsymbol{\Gamma}_{\bar{r}}\mathbf{F}\boldsymbol{\Gamma}_{\bar{r}} \end{bmatrix} \end{aligned}$$

where

$$\mathbf{F} = (\mathbf{D}_r\boldsymbol{\Omega}_{\bar{r}}\mathbf{D}_r - (\boldsymbol{c}_r^T\boldsymbol{c}_r)\,\boldsymbol{\Gamma}_{\bar{r}}\,)^{-1}\,.$$

Similarly, we can express $\mathbf{D}^{-1}\boldsymbol{g}$ and $\mathbf{Z}^T\mathbf{D}^{-1}\boldsymbol{g}$ in block forms as

$$\begin{aligned} \mathbf{D}^{-1}\boldsymbol{g} &= \begin{bmatrix} \frac{1}{\gamma_r}\boldsymbol{g}_1 \\ \mathrm{vec}\left(\mathbf{G}_2\,\boldsymbol{\Gamma}_{\bar{r}}^{-1}\right) \end{bmatrix}, \\ \boldsymbol{w} = \mathbf{Z}^T\mathbf{D}^{-1}\boldsymbol{g} &= \begin{bmatrix} \frac{1}{\gamma_r}\mathbf{A}_{\bar{r}}^T\boldsymbol{g}_1 \\ \boldsymbol{\Gamma}_{\bar{r}}^{-1}\mathbf{G}_2^T\boldsymbol{b}_r \end{bmatrix}, \end{aligned}$$

and write solution to the linear system $\boldsymbol{\Psi}^{-1}\,(\mathbf{Z}^T\mathbf{D}^{-1}\boldsymbol{g})$

$$\begin{aligned} \boldsymbol{\Psi}^{-1}\boldsymbol{w} &= \begin{bmatrix} \mathbf{D}_r\mathbf{F}\left(\mathbf{D}_r\mathbf{A}_{\bar{r}}^T\boldsymbol{g}_1 - (\boldsymbol{c}_r^T\boldsymbol{c}_r)\,\mathbf{G}_2^T\boldsymbol{b}_r\right) \\ \frac{-1}{\boldsymbol{b}_r^T\boldsymbol{b}_r}\boldsymbol{\Gamma}_{\bar{r}}\mathbf{F}\mathbf{D}_r\mathbf{A}_{\bar{r}}^T\boldsymbol{g}_1 + \frac{1}{\boldsymbol{b}_r^T\boldsymbol{b}_r}\mathbf{G}_2^T\boldsymbol{b}_r + \frac{\boldsymbol{c}_r^T\boldsymbol{c}_r}{\boldsymbol{b}_r^T\boldsymbol{b}_r}\boldsymbol{\Gamma}_{\bar{r}}\mathbf{F}\mathbf{G}_2^T\boldsymbol{b}_r \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{D}_r\boldsymbol{u} \\ \frac{1}{\boldsymbol{b}_r^T\boldsymbol{b}_r}\left(\mathbf{G}_2^T\boldsymbol{b}_r - \boldsymbol{\Gamma}_{\bar{r}}\boldsymbol{u}\right) \end{bmatrix}. \end{aligned} \qquad (11)$$

By replacing the result in (11) in to (10), and taking into account that

$$\mathbf{D}^{-1}\,\mathbf{Z}\,\boldsymbol{\Psi}^{-1}\,\boldsymbol{w} \;=\; \begin{bmatrix} \frac{1}{\gamma_1}\mathbf{A}_{\bar{r}}\,\mathbf{D}_r\,\boldsymbol{u} \\ \frac{1}{\boldsymbol{b}_r^T\boldsymbol{b}_r}\,(\boldsymbol{\Gamma}_{\bar{r}}^{-1}\mathbf{G}_2^T\boldsymbol{b}_r - \boldsymbol{u}) \otimes \boldsymbol{b}_r \end{bmatrix},$$

we obtain the update rules in (5) and (6). This completes the proof.

□

6. REFERENCES

[1] T.G. Kolda and B.W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, September 2009.

[2] 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.

[3] 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*, vol. 32, no. 2, pp. 145–163, 2015.

[4] P. Comon, X. Luciani, and A. L. F. de Almeida, "Tensor decompositions, alternating least squares and other tales," *Journal of Chemometrics*, vol. 23, no. 7-8, pp. 393–405, 2009.

[5] A. Uschmajew, "Local convergence of the alternating least squares algorithm for canonical tensor approximation," *SIAM J. Matrix Anal. Appl.*, vol. 33, no. 2, pp. 639–652, 2012.

[6] M. Rajih, P. Comon, and R. A. Harshman, "Enhanced line search: A novel method to accelerate PARAFAC," *SIAM Journal of Matrix Analysis and Applications*, vol. 30, no. 3, pp. 1128–1147, 2008.

[7] P. Paatero, C. Navasca, and P. Hopke, "Fast rotationally enhanced alternating-least-squares," Workshop on Tensor Decompositions and Applications (TDA 2010), SIAM, 2010.

[8] P. Paatero, "A weighted non-negative least squares algorithm for three-way PARAFAC factor analysis," *Chemometrics Intelligent Laboratory Systems*, vol. 38, no. 2, pp. 223–242, 1997.

[9] A.-H. Phan, P. Tichavský, and A. Cichocki, "Low complexity damped Gauss-Newton algorithms for CANDECOMP/PARAFAC," *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 1, pp. 126–147, 2013.

[10] L. Sorber, M. Van Barel, and L. De Lathauwer, "Structured data fusion," *IEEE J. Selected Topics in Signal Processing*, vol. 9, pp. 586–600, 2015.

[11] P. Comon, "Tensor diagonalization, a useful tool in signal processing," in *10th International Federation of Automatic Control Symposium on System Identification*, 1994, pp. 77–82.

[12] L. De Lathauwer, "A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization," *SIAM Journal of Matrix Analysis and Applications*, vol. 28, pp. 642–666, 2006.

[13] Florian Roemer and Martin Haardt, "A semi-algebraic framework for approximate CP decompositions via simultaneous matrix diagonalizations (SECSI)," *Signal Processing*, vol. 93, no. 9, pp. 2722–2738, 2013.

[14] K. Naskovska, M. Haardt, P. Tichavský, G. Chabriel, and J. Barrere, "Extension of the semi-algebraic framework for approximate cp decompositions via nonsymmetric simultaneous matrix diagonalization," in *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Shanghai, China, 2016, pp. 2971–2975.

[15] P. Tichavský, A.-H. Phan, and A. Cichocki, "Tensor diagonalization-a new tool for PARAFAC and block-term decomposition," *arXiv preprint*, vol. arXiv:1402.1673, 2014, 2014.

[16] P. Tichavský, A.-H. Phan, and A. Cichocki, "Partitioned alternating least squares technique for canonical polyadic tensor decomposition," *IEEE Signal Processing Letters*, vol. 23, no. 7, pp. 993–997, July 2016.

[17] A. Cichocki and A.-H. Phan, "Fast local algorithms for large scale nonnegative matrix and tensor factorizations," *IEICE Transactions*, vol. 92-A, no. 3, pp. 708–721, 2009.

[18] P. Tichavský, A.-H. Phan, and A. Cichocki, "A further improvement of a fast damped GAUSS-NEWTON algorithm for CANDECOMP-PARAFAC tensor decomposition," in *Proc. of IEEE Int. Conf. Acoustics, Speech, Signal Processing, ICASSP-2013*, 2013, pp. 5964–5968.

[19] A.-H. Phan, P. Tichavský, and A. Cichocki, "MATLAB TENSORBOX package," http://www.bsp.brain.riken.jp/ phan/tensorbox.php, 2012.

[20] A.-H. Phan, P. Tichavský, and A. Cichocki, "Tensor deflation for CANDECOMP/PARAFAC. Part 2: Initialization and Error analysis.," *IEEE Transaction on Signal Processing*, vol. 63, no. 12, pp. 5939–5950, 2015.

[21] E. Sanchez and B.R. Kowalski, "Tensorial resolution: a direct trilinear decomposition," *J. Chemometrics*, vol. 4, pp. 29–45, 1990.

[22] P. Tichavský, A.-H. Phan, and A. Cichocki, "Numerical cp decomposition of some difficult tensors.," *J. Computational and Applied Mathematics*, vol. 317, pp. 362–370, 2017.