JSIAM Letters Vol.7 (2015) pp.5–8 ©2015 Japan Society for Industrial and Applied Mathematics

A fast alternating least squares method for third-order tensors based on a compression procedure

Tomonori Murakoshi[1] and Takayasu Matsuo[1]

[1] Department of Mathematical Informatics, Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan

E-mail *tomonori_murakoshi@mist.i.u-tokyo.ac.jp*

Received September 30, 2014, Accepted November 13, 2014

Abstract

The alternating least squares (ALS) method is frequently used for the computation of the canonical polyadic decomposition (CPD) of tensors. It generally gives accurate solutions, but demands much time. An alternative is the alternating slice-wise diagonalization (ASD) method, which provides an efficient way for third-order tensors, utilizing compression based on matrix singular value decomposition. In this paper, we propose a new simple algorithm, Reduced ALS, which employs the same compression procedure as ASD, but applies it more directly to ALS. Numerical experiments show that Reduced ALS runs as fast as ASD, avoiding instability ASD sometimes exhibits.

Keywords third-order tensors, CP decomposition, canonical decomposition (CANDECOMP), parallel factors (PARAFAC), alternating least squares (ALS)

Research Activity Group Algorithms for Matrix / Eigenvalue Problems and their Applications

1. Introduction

Tensors (i.e. data arrays with multiple indices) are frequently used in a wide range of applications such as computer vision, signal processing, and data mining. There, often "tensor decompositions" are utilized to analyze and/or compress tensors.

One of the tensor decompositions is the canonical polyadic decomposition (CPD; also known as CANDECOMP/PARAFAC), first proposed in [1] (see also [2] and the references therein). Due to its theoretical and practical importance, there have been vast amount of studies on the computation of CPD. The alternating least squares (ALS) method, proposed in [3, 4], is the oldest and still strongest algorithm for computing CPD. It generally enjoys high accuracy, but demands many iterations to converge. To overcome this difficulty, many variants of ALS (modification of the algorithm itself, or combinations with other techniques for speeding up) have been proposed (see, for example, [5–8] and their references). Other iterative methods for CPD includes, for example, [9] which employed nonlinear least squares approach. Recently, direct methods utilizing matrix singular value decomposition (SVD) have been also studied (see, for example, [10]).

In this paper, we focus on the alternating slice-wise diagonalization (ASD) method by [11], which is an efficient variant of ALS. It limits its targets only to third-order tensors, which still cover many practical applications, and then incorporates a compression technique based on SVD. Thanks to this, the sizes of the matrices appearing in the iterative process are greatly reduced, and the overall computational complexity is decreased. In several review papers, the practical efficiency of ASD was confirmed [5, 7]. It was also pointed out, however, that sometimes ASD exhibited instability and provided unacceptable results.

The goal of this paper is to deliver a new algorithm which lies somewhere between ALS and ASD. There, the essential part of the compression technique in ASD is extracted, and then more simply applied to ALS. The new, simple algorithm is confirmed to work as well as ASD by numerical experiments, and interestingly, sometimes avoid the instability that ASD suffers.

We define $\mathcal{T} \in \mathbb{R}^{I\times J\times K}$ as a real $I \times J \times K$ tensor. The frontal slices of $\mathcal{T}$ are K matrices $\boldsymbol{T}_1, \dots, \boldsymbol{T}_K$. Let $\boldsymbol{x}_1, \dots, \boldsymbol{x}_R \in \mathbb{R}^I, \boldsymbol{y}_1, \dots, \boldsymbol{y}_R \in \mathbb{R}^J, \boldsymbol{z}_1, \dots, \boldsymbol{z}_R \in \mathbb{R}^K$ be column real vectors. Then CPD is expressed as

$$\mathcal{T} \approx \sum_{r=1}^{R} \boldsymbol{x}_r \circ \boldsymbol{y}_r \circ \boldsymbol{z}_r, \quad (\boldsymbol{x} \circ \boldsymbol{y} \circ \boldsymbol{z})_{ijk} = x_i y_j z_k. \qquad (1)$$

Determining the "exact" rank, the minimum R which realizes (1) with strict equality, is NP-complete [12]. In practice, the expression (1) is used in such a way that we first fix a small R, and seek for $\boldsymbol{x}_r, \boldsymbol{y}_r$ and $\boldsymbol{z}_r$ that satisfy the equality as good as possible. In what follows, we suppose $R < I, J$, following ASD, with the application to the low-rank approximation of large third-order tensors in mind.

2. Existing works

2.1 ALS

Let $\boldsymbol{X} = (\boldsymbol{x}_1 \dots \boldsymbol{x}_R)$, $\boldsymbol{Y} = (\boldsymbol{y}_1 \dots \boldsymbol{y}_R)$, $\boldsymbol{Z} = (\boldsymbol{z}_1 \dots \boldsymbol{z}_R)$ be $I \times R$, $J \times R$, and $K \times R$ matrices respectively. The idea of ALS is to solve the least squares problem:

$$\min_{\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z}} \left\| \mathcal{T} - \sum_{r=1}^{R} \boldsymbol{x}_r \circ \boldsymbol{y}_r \circ \boldsymbol{z}_r \right\|^2, \quad \|\mathcal{T}\| = \sqrt{\sum_{i,j,k} (t_{ijk})^2}.$$

In this paper, by $\|\cdot\|$ we mean the vector 2-norm, or the matrix/tensor Frobenius norms depending on its context. If $\boldsymbol{Y}$ and $\boldsymbol{Z}$ are fixed, this problem is equivalent to

$$\min_{\boldsymbol{X}} \left\| \boldsymbol{T}_{(1)} - \boldsymbol{X}(\boldsymbol{Z} \odot \boldsymbol{Y})^\top \right\|^2, \tag{2}$$

where $\boldsymbol{T}_{(n)}$ $(n = 1, 2, 3)$ is the mode-n matricization of $\mathcal{T}$, and "$\odot$" is the matrix Khatri–Rao product (see [2]). At this point, $\boldsymbol{T}_{(1)}$ is just an $I \times JK$ matrix, and $\boldsymbol{Z} \odot \boldsymbol{Y}$ is a $JK \times R$ matrix. Then the optimal solution of (2) can be easily found to be

$$\boldsymbol{X} = \boldsymbol{T}_{(1)}[(\boldsymbol{Z} \odot \boldsymbol{Y})^\top]^\dagger = \boldsymbol{T}_{(1)}(\boldsymbol{Z} \odot \boldsymbol{Y})(\boldsymbol{Z}^\top\boldsymbol{Z} * \boldsymbol{Y}^\top\boldsymbol{Y})^{-1}, \tag{3}$$

where $\dagger$ is the Moore–Penrose inverse, and "$*$" is the Hadamard product. Similarly, $\boldsymbol{Y}$ (fixing $\boldsymbol{X}$ and $\boldsymbol{Z}$) and $\boldsymbol{Z}$ (fixing $\boldsymbol{X}$ and $\boldsymbol{Y}$) can be updated.

The above procedure, incorporated with a balancing process of the sizes of $\|\boldsymbol{X}\|$, $\|\boldsymbol{Y}\|$, and $\|\boldsymbol{Z}\|$, can be summarized as follows.

1. Update $\boldsymbol{X}$, $\boldsymbol{Y}$, and $\boldsymbol{Z}$ in turn.

2. Calculate $\lambda_r = \|\boldsymbol{x}_r\|\|\boldsymbol{y}_r\|\|\boldsymbol{z}_r\|$, $\quad (r = 1, \dots, R)$.

3. Normalize the columns of $\boldsymbol{X}$, $\boldsymbol{Y}$, and $\boldsymbol{Z}$.

4. Multiply $\boldsymbol{x}_r$, $\boldsymbol{y}_r$, and $\boldsymbol{z}_r$ by $\sqrt[3]{\lambda_r}$ respectively.

5. Repeat **1**–**4**.

ALS monotonically reduces the squared error in each iteration, and eventually provides an accurate decomposition. Numerical examples in [5, 7] confirm this, but at the same time reveal that ALS requires many iterations. Moreover, ALS involves calculations with large matrices; recall that the sizes of $\boldsymbol{T}_{(1)}$ and $\boldsymbol{Z} \odot \boldsymbol{Y}$ in (3) are $I \times JK$ and $JK \times R$, which can be huge for large I, J. In this way, ALS does not fit practical applications which demand speed rather than accuracy.

2.2 ASD

ASD focuses on the fact that third-order tensors can be viewed as a collection of matrices, by slicing. Then it utilizes a standard compression procedure to the sliced matrices. By this compression, the matrices appearing in the iterative process are greatly simplified, and in fact, the numerical examples in [5, 7] confirm the efficiency of ASD.

Let $L = \left\| \mathcal{T} - \sum_{r=1}^{R} \boldsymbol{x}_r \circ \boldsymbol{y}_r \circ \boldsymbol{z}_r \right\|^2$ be the squared error. First, the squared error is transformed as follows:

$$\begin{aligned} L &= \sum_{k=1}^{K} \sum_{r=1}^{R} \left\| \boldsymbol{T}_k - z_{kr}\boldsymbol{x}_r\boldsymbol{y}_r^\top \right\|^2 \\ &= \sum_{k=1}^{K} \left\| \boldsymbol{T}_k - \boldsymbol{X}\operatorname{diag}(\boldsymbol{z}_{(k)})\boldsymbol{Y}^\top \right\|^2, \end{aligned} \tag{4}$$

where $\boldsymbol{z}_{(k)}$ denotes the k-th row vector of $\boldsymbol{Z}$. Next, supposing we can find such matrices $\boldsymbol{P}$ (size $I \times R$) and $\boldsymbol{Q}$ $(J \times R)$ that satisfy $\boldsymbol{P}^\top\boldsymbol{X} = \boldsymbol{I}_R$ and $\boldsymbol{Y}^\top\boldsymbol{Q} = \boldsymbol{I}_R$, we modify the expression in (4) as follows.

$$\begin{aligned} L \approx & \sum_{k=1}^{K} \left\| \boldsymbol{P}^\top\boldsymbol{T}_k\boldsymbol{Q} - \operatorname{diag}(\boldsymbol{z}_{(k)}) \right\|^2 \\ & + \lambda \left\| \boldsymbol{P}^\top\boldsymbol{X} - \boldsymbol{I}_R \right\|^2 + \lambda \left\| \boldsymbol{Y}^\top\boldsymbol{Q} - \boldsymbol{I}_R \right\|^2. \end{aligned} \tag{5}$$

The latter two terms are the penalty terms enforcing the existence of $\boldsymbol{P}$ and $\boldsymbol{Q}$; $\lambda > 0$ is the strength of the penalties.

Finally, we introduce a compression procedure. Let $\boldsymbol{U}$, $\boldsymbol{V}$ be the $I \times R$ and $J \times R$ matrices whose column vectors are the R left singular vectors of $\sum_{k=1}^{K} \boldsymbol{T}_k\boldsymbol{T}_k^\top$ and $\sum_{k=1}^{K} \boldsymbol{T}_k^\top\boldsymbol{T}_k$ respectively. Then we consider the compressed expressions using $\boldsymbol{U}$ and $\boldsymbol{V}$: $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{A}$, $\boldsymbol{Y} = \boldsymbol{V}\boldsymbol{B}$, $\boldsymbol{P} = \boldsymbol{U}\boldsymbol{G}$, $\boldsymbol{Q} = \boldsymbol{V}\boldsymbol{H}$, and $\tilde{\boldsymbol{T}}_k = \boldsymbol{U}^\top\boldsymbol{T}_k\boldsymbol{V}$. There, $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{G}, \boldsymbol{H}$, and $\tilde{\boldsymbol{T}}_k$ are $R \times R$ matrices, which are much smaller than $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{P}, \boldsymbol{Q}$, and $\boldsymbol{T}_k$ (recall the assumption $R < I, J$). Note that such matrices do not necessarily exist, and the "compression" is an approximation. Substituting the compressed expressions into (5), we obtain a new objective function, L_{SD}, involving only smaller matrices:

$$\begin{aligned} L \approx L_{\mathrm{SD}} = & \sum_{k=1}^{K} \|\boldsymbol{G}^\top\tilde{\boldsymbol{T}}_k\boldsymbol{H} - \operatorname{diag}(\boldsymbol{z}_{(k)})\|^2 \\ & + \lambda\|\boldsymbol{G}^\top\boldsymbol{A} - \boldsymbol{I}_R\|^2 + \lambda\|\boldsymbol{B}^\top\boldsymbol{H} - \boldsymbol{I}_R\|^2. \end{aligned}$$

ASD updates $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{Z}, \boldsymbol{G}$, and $\boldsymbol{H}$, employing L_{SD} instead of L. Due to the penalty terms, the updates are not as easy as ALS; it instead considers the derivatives of L_{SD}. For example, the update of $\boldsymbol{A}$ is realized based on the condition $\partial L_{\mathrm{SD}}/\partial\boldsymbol{A} = 0$. In this sense, ASD does not necessarily strictly minimize the objective function L_{SD}, and accordingly, the original error L. In fact, in some preliminary numerical experiments, we observed that the value of L_{SD} could sometimes go up, and also that the behaviors of L_{SD} and L did not necessarily agree. The reason of the occasional instability of ASD can be attributed to these natures.

3. Proposed Method

Based on the above observation, we here propose a simpler approach. Our strategy is summarized as follows.

I. We start with the same matrix form (4) (thus we limit ourselves to third-order tensors).

II. We employ a similar compression procedure to ASD.

III. But we try to keep the objective function as close as possible to the original L.

The resulting algorithm, which we call Reduced ALS, is expected to be as fast as ASD, and to have a better stability due to the third requirement.

We commence by substituting the compressed expressions into (4) directly. Note that $\boldsymbol{T}_k$ is approximated as $\boldsymbol{T}_k \approx \boldsymbol{U}\tilde{\boldsymbol{T}}_k\boldsymbol{V}^\top = \boldsymbol{U}\boldsymbol{U}^\top\boldsymbol{T}_k\boldsymbol{V}\boldsymbol{V}^\top$. In this way, the approximation of L, denoted as L_{Re}, is expressed as

$$L \approx L_{\mathrm{Re}} = \sum_{k=1}^{K} \left\| \boldsymbol{U}\tilde{\boldsymbol{T}}_k\boldsymbol{V}^\top - \boldsymbol{U}\boldsymbol{A}\operatorname{diag}(\boldsymbol{z}_{(k)})\boldsymbol{B}^\top\boldsymbol{V}^\top \right\|^2. \tag{6}$$

Next, we remove $\boldsymbol{U}$ and $\boldsymbol{V}^\top$ from (6) in order to reduce our algorithm to ALS. It is easy to see that for an $R \times M$ matrix $\boldsymbol{M} = (\boldsymbol{m}_1, \dots, \boldsymbol{m}_M)$, it holds

$$\|\boldsymbol{U}\boldsymbol{M}\|^2 = \sum_{l=1}^{M} \left\|(\boldsymbol{U}\boldsymbol{m}_l)^\top \boldsymbol{U}\boldsymbol{m}_l\right\|^2 = \|\boldsymbol{M}\|^2. \qquad (7)$$

This is also true for $\boldsymbol{V}$. Based on these observations, we see, by appropriately considering transposes,

$$\begin{aligned} L_{\mathrm{Re}} &= \sum_{k=1}^{K} \left\|\boldsymbol{U}\tilde{\boldsymbol{T}}_k\boldsymbol{V}^\top - \boldsymbol{U}\boldsymbol{A}\mathrm{diag}(\boldsymbol{z}_{(k)})\boldsymbol{B}^\top\boldsymbol{V}^\top\right\|^2 \\ &= \sum_{k=1}^{K} \left\|\boldsymbol{V}\tilde{\boldsymbol{T}}_k - \boldsymbol{V}\boldsymbol{B}\mathrm{diag}(\boldsymbol{z}_{(k)})\boldsymbol{A}^\top\right\|^2 \\ &= \sum_{k=1}^{K} \left\|\tilde{\boldsymbol{T}}_k - \boldsymbol{A}\mathrm{diag}(\boldsymbol{z}_{(k)})\boldsymbol{B}^\top\right\|^2. \end{aligned}$$

Then L_{Re} can be rewritten back to the original form

$$\sum_{k=1}^{K} \left\|\tilde{\boldsymbol{T}}_k - \boldsymbol{A}\mathrm{diag}(\boldsymbol{z}_{(k)})\boldsymbol{B}^\top\right\|^2 = \left\|\tilde{\mathcal{T}} - \sum_{r=1}^{R} \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{z}_r\right\|^2$$

This implies that we have reduced the size of the original problem, and we can simply apply the standard ALS to L_{Re}. There, ALS decomposes $\tilde{\mathcal{T}}$ (compressed $\mathcal{T}$) into $\boldsymbol{A}$ (compressed $\boldsymbol{X}$), $\boldsymbol{B}$ (compressed $\boldsymbol{Y}$), and $\boldsymbol{Z}$. Then we can obtain the desired decomposition in terms of $\mathcal{T}$ by $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{A}$, and so on.

4. Numerical experiments

Before starting experiments, we further simplify the calculation of $\boldsymbol{U}$ and $\boldsymbol{V}$ to reduce the overall computational complexity; we calculate $\sum_{k=1}^{K} \boldsymbol{T}_k$ and let $\boldsymbol{U}(\boldsymbol{V})$ be the matrix whose column vectors are its R left (right) singular vectors. If the common singular matrices $\boldsymbol{U}_{\mathrm{full}}$ and $\boldsymbol{V}_{\mathrm{full}}$ exist for $\boldsymbol{T}_1, \dots, \boldsymbol{T}_K$ simultaneously (i.e., $\boldsymbol{T}_k = \boldsymbol{U}_{\mathrm{full}}\boldsymbol{\Sigma}_k\boldsymbol{V}_{\mathrm{full}}^\top, k = 1, \dots, K$), $\boldsymbol{U}$ and $\boldsymbol{V}$ defined above coincide with the original $\boldsymbol{U}$ and $\boldsymbol{V}$ in Section 2.2.

In order to set each stopping criterion, we considered $L/\|\mathcal{T}\|^2$ for ALS, $L_{\mathrm{SD}}/\|\tilde{\mathcal{T}}\|^2$ for ASD, and $L_{\mathrm{Re}}/\|\tilde{\mathcal{T}}\|^2$ for Reduced ALS. Then ALS, ASD, and Reduced ALS were stopped if the update of each value fell below 10^{-3}.

Initial matrices of $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{Z}$ were generated by uniform random numbers on $[0,1]$. After $\boldsymbol{U}$ and $\boldsymbol{V}$ were calculated using a given tensor $\mathcal{T}$, we set $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{A}$, $\boldsymbol{Y} = \boldsymbol{V}\boldsymbol{B}$, $\boldsymbol{G} = (\boldsymbol{A}^\top)^{-1}$, and $\boldsymbol{H} = (\boldsymbol{B}^\top)^{-1}$. The strength λ in ASD was set to 10^{-3} following [5,7,11].

Every experiment was conducted on a laptop PC with a quad-core 2.4GHz Intel Core i7-3630QM processor and 16GB memory. The operating system was Windows 7 Home Premium 64bit, and MATLAB 8.1.0.604 (R2013a) was used. We employed Matlab tensor toolbox 2.5 produced by [13], and ASD was implemented following [5].

4.1 Generated data

We used the generated random tensors proposed in [7] (also used in [8,9]). These tensors are generated using the 3 parameters: l_1 and l_2 which control the levels of two kinds of noise ($l_1, l_2 \geq 1$ and 1 implies no noise), and c controlling the collinearity (see [7,8]). Parameters were set to $I = J = K = 500, R = 10, l_1 = 5$, and $l_2 = 10$. Ten trials were run changing both a random tensor $\mathcal{T}$ and initial matrices, for $c = 0.5$, and $c = 0.9$. On every trial, we measured the time to converge, the number of iterations, and the relative error in terms of L in the obtained CPD (i.e. $\left\|\mathcal{T} - \sum_{r=1}^{R} \boldsymbol{x}_r \circ \boldsymbol{y}_r \circ \boldsymbol{z}_r\right\|^2 / \|\mathcal{T}\|^2$).

Table 1. The results for generated data.

$c = 0.5$	time (s)	relative errors	iterations
ALS	42.7 ± 8.47	0.153 ± 0.0109	13.5 ± 2.76
ASD	3.9 ± 0.20	0.296 ± 0.2149	6.9 ± 2.13
Re ALS	4.0 ± 0.20	0.220 ± 0.0396	11.7 ± 3.16

$c = 0.9$	time (s)	relative errors	iterations
ALS	9.9 ± 0.16	0.146 ± 0.0001	3.0 ± 0.00
ASD	3.8 ± 0.17	0.171 ± 0.0505	5.2 ± 1.99
Re ALS	3.6 ± 0.06	0.169 ± 0.0506	2.0 ± 0.00

Table 1 shows the results. Each average and standard deviation on 10 trials are described. First, we see that ASD and Reduced ALS ran faster than ALS in both cases ($c = 0.5, 0.9$); in particular, Reduced ALS ran as fast as ASD as expected. The compression procedure commonly employed in ASD and Reduced ALS certainly improved the efficiency. Second, ALS achieved the best accuracy. The other two algorithms provided worse accuracies, which should be attributed to the introduction of the compression procedure. The difference was, however, not quite big and acceptable. We also observe that Reduced ALS was slightly superior to ASD; there is no clear theoretical explanation for this, but it is in some sense likely since in Reduced ALS, many approximation processes in ASD are dropped. In the case of $c = 0.5$, Reduced ALS gave smaller deviation than ASD. This confirms the instability of ASD mentioned above.

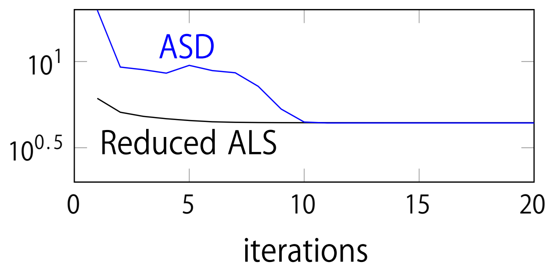

Fig. 1. Convergence profiles of L for Reduced ALS and ASD. Parameters were set to $I = J = K = 50, R = 10, l_1 = l_2 = 1$, and $c = 0.5$ in this experiment.

We also compared convergence profiles of L for Reduced ALS and ASD using a small and noiseless tensor. Fig. 1 shows that L of Reduced ALS is monotonically decreased, while that of ASD showed irregular behavior.

4.2 Real data

Next we consider more realistic data; we employ the ORL face database [14]. First, we prepared two $112 \times 92 \times 10$ tensors consisting of 10 pictures in the database (the set in the folders "s1" and "s11"). Fig. 2 shows the

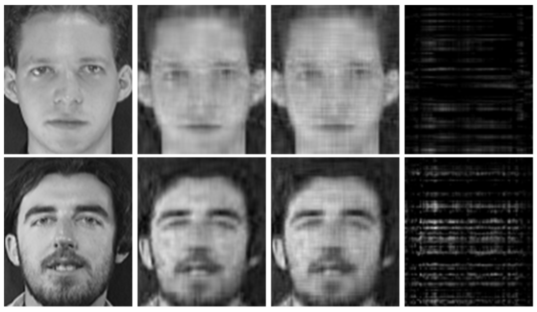
Fig. 2. The results for the picture data ($K = 10$). From left: the original, ALS, Reduced ALS, and ASD results. The first row corresponds to one of the dataset 1, and the second to one of the dataset 2.

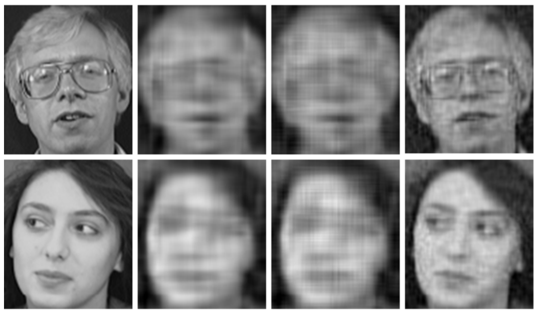
Fig. 3. The result for the picture data ($K = 110$). From left: the original, ALS ($R = 72$), Reduced ALS ($R = 72$), and ALS ($R = 500$). Only two pieces of 110 pictures are shown here.

pictures compressed with CPD created by ALS, ASD, and Reduced ALS (all $R = 72$). Reduced ALS gave clear pictures in the same level as ALS. The compression ratio of the data was $IJK/[(I + J + K)R] \approx 6.687$. On the other hand, ASD failed to catch a proper face picture; in fact, even after 10,000 iterations (i.e. even if we continue the iteration after the stopping criterion), the pictures by ASD were still blurred. This points out another qualitative instability of ASD.

Next, we considered a $112 \times 92 \times 110$ tensor consisting of 110 pictures (in the folders "s1", "s2", ..., and "s11"). Fig. 3 shows the results by ALS ($R = 72$), Reduced ALS ($R = 72$), and ALS ($R = 500$). If $R = 72$, the compression ratio was quite high, about 50.13. The pictures were, however, so blurred that only the shape of the faces could be identified. In our experiment, we observed that for example $R = 500$ was necessary to recover good pictures (see the ALS with $R = 500$ column). Even for this big R, the compression ratio was 7.21, and thus CPD can make sense. This example illustrates the limitation of Reduced ALS (and ASD); in some cases, we need big R in CPD. In the above example, $R = 500$ for $I = 112$ and $J = 92$. However, Reduced ALS (and ASD) intrinsically demands the assumption $R < I, J$ for the compression being able to work, and the algorithm ceases to work if the assumption is broken.

5. Concluding Remarks

We proposed Reduced ALS that utilizes the compression procedure of ASD more directly to ALS. Although the idea is quite simple, numerical experiments showed that Reduced ALS had the expected property: high efficiency and accuracy at the same time. It has the limitation, however, that it can work only for third-order tensors and its CPD with the small rank $R < I, J$.

Our future works include the application of Reduced ALS to various real data, and comparison with other CPD algorithms. It should be also interesting to investigate theoretically the mathematical structure of the optimization problems with respect to L and L_{Re}; we expect these two problems are not so far apart, but its theoretical explanation is hoped. Finally, at this moment, it is left open if the compression technique employed in ASD and Reduced ALS can be extended to fourth- or higher-order tensors.

Acknowledgments

We really appreciate Prof. Hans De Sterck, the author of [8], who provided us with some codes, and also many helpful advices. We are also grateful for the anonymous reviewer who gave us helpful comments.

References

[1] F. L. Hitchcock, The expression of a tensor or a polyadic as a sum of products, J. Math. Phys., **6** (1927), 164–189.

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

[3] J. D. Carroll and J. J. Chang, Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart–Young" decomposition, Psychometrika, **35** (1970), 283–319.

[4] R. A. Harshman, Foundations of the PARAFAC procedure: models and conditions for an "explanatory" multimodal factor analysis, UCLA Working Papers in Phonetics, **16** (1970), 1–84.

[5] N. M. Faber, R. Bro and P. K. Hopke, Recent developments in CANDECOMP/PARAFAC algorithms: a critical review, Chemometr. Intell. Lab. Syst., **65** (2003), 119–137.

[6] N. Li, S. Kindermann and C. Navasca, Some convergence results on the regularized alternating least-squares method for tensor decomposition, Linear Algebra Appl., **438** (2013), 796–812.

[7] G. Tomasi and R. Bro, A comparison of algorithms for fitting the PARAFAC model, Comput. Stat. Data Anal., **50** (2006), 1700–1734.

[8] H. De Sterck, A nonlinear GMRES optimization algorithm for canonical tensor decomposition, SIAM J. Sci. Comput., **34** (2012), A1351–A1379.

[9] E. Acar, D. M. Dunlavy and T. G. Kolda, A scalable optimization approach for fitting canonical tensor decompositions, J. Chemometrics, **25** (2011), 67–86.

[10] I. Domanov and L. De Lathauwer, Canonical polyadic decomposition of third-order tensors: reduction to generalized eigenvalue decomposition, SIAM J. Matrix Anal. Appl., **35** (2014), 636–660.

[11] J. Jiang, H. Wu, Y. Li and R. Yu, Three-way data resolution by alternating slice-wise diagonalization (ASD) method, J. Chemometrics, **14** (2000), 15–36.

[12] J. Håstad, Tensor rank is NP-complete, J. Algorithms, **11** (1990), 644–654.

[13] B. W. Bader and T. G. Kolda, Matlab tensor toolbox version 2.5, `http://www.sandia.gov/~tgkolda/TensorToolbox/index-2.5.html`.

[14] Olivetti Research Laboratory, The ORL face database, `http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`.