On Optimal Low Rank Tucker Approximation for Tensors: The Case for an Adjustable Core Size

Bilian CHEN * Zhening LI † Shuzhong ZHANG ‡

August 7, 2014

Abstract

Approximating high order tensors by low Tucker-rank tensors have applications in psychometrics, chemometrics, computer vision, biomedical informatics, among others. Traditionally, solution methods for finding a low Tucker-rank approximation presume that the size of the core tensor is specified in advance, which may not be a realistic assumption in many applications. In this paper we propose a new computational model where the configuration and the size of the core become a part of the decisions to be optimized. Our approach is based on the so-called maximum block improvement (MBI) method for non-convex block optimization. Numerical tests on various real data sets from gene expression analysis and image compression are reported, which show promising performances of the proposed algorithms.

Keywords: multiway array, Tucker decomposition, low-rank approximation, maximum block improvement.

Mathematics Subject Classification: 15A69, 90C26, 90C59, 49M27, 65F99.

*Department of Automation, Xiamen University, Xiamen 361005, China. Email: blchen@xmu.edu.cn.

†Department of Mathematics, University of Portsmouth, Portsmouth PO1 3HF, United Kingdom. Email: zheningli@gmail.com.

‡Department of Industrial and Systems Engineering, University of Minnesota, Minneapolis, MN 55455, USA. Email: zhangs@umn.edu.

1 Introduction

The models and solution methods for high order tensor decompositions were initially motivated and developed in psychometrics (see e.g. [37, 38, 39]) and chemometrics (see e.g. [2]) in response to the need of analysis for multiway data. Since then, tensor decomposition has become an active field of research in mathematics due partly to the intricate algebraic structures of tensors. In addition, high order (tensor) statistics and independent component analysis (ICA) have been developed by engineers and statisticians primarily because of their wide practical applications. There are two major tensor decompositions that can be considered as higher order extensions of the matrix singular value decomposition (SVD): one is known as the CANDECOMP/PARAFAC (CP) decomposition, which attempts to present the tensor as a sum of rank-one tensors, and the other is known as the Tucker decomposition, which is a higher order generalization of the principal component analysis (PCA). The Tucker decomposition can be viewed as a generalization of the CP decomposition which is a Tucker model with equal number of components in each mode. It was first introduced by Tucker [37] in 1963, and later redefined in Levin [28] and Tucker [38, 39]. There are extensive studies in the literature on the topic of finding the CP decomposition and/or the Tucker decomposition for tensors; see e.g. [14, 15, 16, 23, 26, 27, 32, 36, 41, 44]. On the computational side, the existing popular algorithms are based on the *alternating least square* (ALS) method proposed originally by Carroll and Chang [9], and Harshman [18]. However, the ALS method is not guaranteed to converge to the global optimum or any stationary point, but only to a solution where the objective function ceases to decrease. If the ALS method converges under certain non-degeneracy assumption, then it actually has a local linear convergence rate [40]. Specifically for the Tucker decomposition, Tucker [39] proposed three methods to find the Tucker decomposition for three-way tensors in 1966, among which the first one is referred to as the Tucker1 method, and is now better known as the higher order singular value decomposition (HOSVD); see [14]. Analogous to the ALS method developed for computing CP decomposition, researchers also derived similar methods for solving the Tucker decomposition, e.g., TUCKALS3 [26] and its extension [21], higher order orthogonal iteration (HOOI) method [15] and its improvement [1]. As mentioned earlier, the ALS method has no convergence guarantee in general. Alternatively, there are methods for the Tucker decomposition with a convergence guarantee; see e.g., the Newton-Grassmann method [17] and the differential-geometric Newton method [20]. For more discussions on the Tucker decomposition and its variants, one is referred to the survey paper by Kolda and Bader [25].

The goal of the Tucker decomposition is to decompose a tensor into a core tensor multiplied by a matrix along each mode. It is related to the best rank-$(r_1, r_2, \ldots, r_d)$ approximation of a d-th order tensor (cf. [15]). Typically, a rank-$(r_1, r_2, \ldots, r_d)$ Tucker decomposition (sometimes called the best multilinear rank-$(r_1, r_2, \ldots, r_d)$ approximation) means that the size of the core tensor is $r_1 \times r_2 \times \cdots \times r_d$. Traditionally, the rank-$(r_1, r_2, \ldots, r_d)$ (namely the size of the core) is considered a set of predetermined parameters. This requirement, however, is restrictive in some applications. For example, the problem of choosing a *good* number of clusters for co-clustering of gene expression data comes up in the area of bioinformatics. In fact, the number of suitable co-clusters is not known in advance (see Zhang et al. [42]). In the same vein, the problem of determining the approximate ranks of a Tucker model (cf. [10, 11, 19, 22, 35]) is challenging, since the problem is already NP-hard even if the rank-$(r_1, r_2, \ldots, r_d)$ is given in advance. The similar issue of selecting an appropriate rank has also been considered in the CP decomposition context, e.g. a consistency diagnostic known as CORCONDIA in [8]. Timmerman and Kiers [35] proposed the so-called DIFFIT procedure based on optimal fit to choose the numbers of components in the Tucker decomposition of a three-way array. Kiers and Der Kinderen [22] revised the procedure of computing the DIFFIT of the approximate fit to save computational effort. The DIFFIT procedure, however, is rather time-

consuming, since its key step needs to compute all the fit values of the Tucker decomposition for all possible combinations (r_1, r_2, r_3) from (1, 1, 1) to a certain combination such that the fit equals to 100%. Moreover, Mørup and Hanson [31] proposed a Bayesian approach called automatic relevance detection (ARD) method for the Tucker decomposition of multi-way data, where the numbers of components for each mode of the core tensor should be chosen large enough at the beginning to encompass all potential models. The ARD method would sequentially remove the excessive components and simplify the core structure (by shrinking the size of the core) at the computational cost of fitting the traditional Tucker decomposition. However, once a component is removed, it will not be brought back again. The authors reported that the ARD method does not always do a good job in identifying the correct core size; also, the higher the signal to noise ratio (SNR) may cause the ARD approach to fail completely. Despite all the drawbacks, it is shown in [31] that the ARD method still outperforms DIFFIT approach in general.

For the problem considered in this paper, we shall resort to a recent block coordinate descent type search method known as *maximum block improvement* (MBI) proposed by Chen et al. [13], for solving nonlinear optimization with a separable block structure, which suits well with the tensor decomposition problems. They proved that the sequence produced by the MBI method converges to a stationary point. This method performs very well numerically. The MBI method has been applied successfully to the problem of finding the best rank-one approximation of tensors [13], and in solving problems arising from bioinformatics [42, 43] and the radar systems [3]. Local linear convergence of the MBI method for certain types of problems were established by Li et al. [29]. Recent treatise on this topic can be found in the Ph.D. thesis of Chen [12]. In this paper, we study the Tucker decomposition problem with the characteristic that the size of its core is unspecified. In particular, we propose a new scheme to choose some *good* ranks for the Tucker decomposition, subject to that the summation of the ranks $\sum_{i=1}^{d} r_i$ is fixed. We shall demonstrate how the MBI method can be applied to solve such problems.

The remainder of the paper is organized as follows. In Section 2, we introduce the notations and frequently used tensor operations throughout the paper. Then we apply the MBI method for solving rank-$(r_1, r_2, \ldots, r_d)$ Tucker decomposition in Section 3. In Section 4, a new model for Tucker decomposition with unspecified core size is proposed and solved based on the MBI method and the penalty method. A heuristic approach is also developed to compute the new model in this section. Finally, numerical results on testing the new model and algorithms are presented in Section 5.

2 Notations

Throughout this paper, we uniformly use non-bold lowercase letters, boldface lowercase letters, capital letters, and calligraphic letters to denote scalars, vectors, matrices, and tensors, respectively; e.g.: scalar i, vector $\mathbf{y}$, matrix A, and tensor $\mathcal{F}$. We use the subscripts to denote the component of a vector, a matrix, or a tensor; e.g.: y_i being the i-th entry of the vector $\mathbf{y}$, A_{ij} being the (i, j)-th entry of the matrix A, and $\mathcal{F}_{ijk}$ being the (i, j, k)-th entry of the tensor $\mathcal{F}$. Let us first introduce some important tensor operations frequently appeared in this paper, which are largely in line with that in [25, 14]. For an overview of tensor operations and properties, we refer to the survey paper [25].

A tensor is a multidimensional array, and the order of a tensor is its dimension, also known as the ways or the modes of a tensor. In particular, a vector is a tensor of order one, and a matrix is a tensor of order two. Consider $\mathcal{A} = (\mathcal{A}_{i_1 i_2 \ldots i_d}) \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, a standard tensor of order d, where $d \geq 3$. A usual way to handle a tensor is to reorder its elements into a matrix; the process is

called *matricization*, also known as *unfolding* or *flattening*. $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ has d modes, namely, mode-1, mode-2, ..., mode-d. Denote the *mode-k matricization* of tensor $\mathcal{A}$ to be $A_{(k)}$, then the $(i_1, i_2, \ldots, i_d)$-th entry of tensor $\mathcal{A}$ is mapped to the (i_k, j)-th entry of matrix $A_{(k)} \in \mathbb{R}^{n_k \times \prod_{\ell \neq k} n_\ell}$, where

$$j = 1 + \sum_{1 \leq \ell \leq d, \ell \neq k} (i_\ell - 1) \prod_{1 \leq t \leq \ell - 1, t \neq k} n_t.$$

The k-rank of tensor $\mathcal{A}$, denoted by $\operatorname{rank}_k(\mathcal{A})$, is the column rank of mode-k unfolding $\mathcal{A}_{(k)}$, i.e., $\operatorname{rank}_k(\mathcal{A}) = \operatorname{rank}(\mathcal{A}_{(k)})$. A d-th order tensor whose $\operatorname{rank}_k(\mathcal{A}) = r_k$ for $k = 1, 2, \ldots, d$, is briefly called a rank-$(r_1, r_2, \ldots, r_d)$ tensor.

Analogous to the Frobenius norm of a matrix, the *Frobenius norm* of tensor $\mathcal{A}$ is the usual 2-norm, defined by

$$\|\mathcal{A}\| := \sqrt{\sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_d=1}^{n_d} \mathcal{A}_{i_1 i_2 \ldots i_d}{}^2}.$$

In this paper we uniformly denote the 2-norm for vectors, and the Frobenius norm for matrices and tensors, all by notation $\|\cdot\|$. The *inner product* of two same-sized tensors $\mathcal{A}, \mathcal{B}$ is given as

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_d=1}^{n_d} \mathcal{A}_{i_1 i_2 \ldots i_d} \mathcal{B}_{i_1 i_2 \ldots i_d}.$$

Hence, it is clear that $\langle \mathcal{A}, \mathcal{A} \rangle = \|\mathcal{A}\|^2$.

One important tensor operation is the multiplication of a tensor by a matrix. The k-mode product of tensor $\mathcal{A}$ by a matrix $U \in \mathbb{R}^{m \times n_k}$, denoted by $\mathcal{A} \times_k U$, is a tensor in $\mathbb{R}^{n_1 \times n_2 \times \cdots \times n_{k-1} \times m \times n_{k+1} \times \cdots \times n_d}$, whose $(i_1, i_2, \ldots, i_{k-1}, \ell, i_{k+1}, \ldots, i_d)$-th entry is defined by

$$(\mathcal{A} \times_k U)_{i_1 i_2 \ldots i_{k-1} \ell \, i_{k+1} \ldots i_d} = \sum_{i_k=1}^{n_k} \mathcal{A}_{i_1 i_2 \ldots i_{k-1} i_k i_{k+1} \ldots i_d} U_{\ell i_k}.$$

The equation can also be written in terms of tensor unfolding as well, i.e.,

$$\mathcal{Y} = \mathcal{A} \times_k U \Longleftrightarrow Y_{(k)} = U A_{(k)}.$$

This multiplication in fact changes the dimension of tensor $\mathcal{A}$ in mode-k. In particular, if U is a vector in $\mathbb{R}^{n_k}$, the order of tensor $\mathcal{A} \times_k U$ is then reduced to $d-1$, whose size is $n_1 \times n_2 \times \cdots \times n_{k-1} \times n_{k+1} \times \cdots \times n_d$. The k-mode products can also be expressed by the matrix Kronecker product as follows:

$$\mathcal{Y} = \mathcal{A} \times_1 U^{(1)} \times_2 U^{(2)} \cdots \times_d U^{(d)} \Longleftrightarrow Y_{(k)} = U^{(k)} A_{(k)} \left(U^{(d)} \otimes \cdots \otimes U^{(k+1)} \otimes U^{(k-1)} \otimes \cdots \otimes U^{(1)} \right),$$

for any $k = 1, 2, \ldots, d$ with $U^{(k)} \in \mathbb{R}^{m_k \times n_k}$ for $k = 1, 2, \ldots, d$. The proof of this property can be found in [24].

Throughout this paper we uniformly use a subscript in the parentheses to denote the matricization of a tensor (e.g. $A_{(1)}$ being mode-1 matricization of tensor $\mathcal{A}$), and use a superscript in the parentheses to denote the matrix in the mode product of a tensor (e.g. $U^{(1)}$ in appropriate size showed in mode-1 product of a tensor).

3 Preliminaries

3.1 Traditional Tucker decomposition

Traditionally, Tucker decomposition attempts to find the best approximation for a large-sized tensor by a small-sized tensor with pre-specified dimensions (called the core tensor) multiplied by a matrix on each mode. The problem can be formulated as follows: Given a real tensor $\mathcal{F} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, find a core tensor $\mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_d}$ with pre-specified integers r_i with $1 \leq r_i \leq n_i$ for $i = 1, 2, \ldots, d$, that optimizes

$$\begin{array}{lll} (T_{\min}) & \min & \left\| \mathcal{F} - \mathcal{C} \times_1 A^{(1)} \times_2 A^{(2)} \cdots \times_d A^{(d)} \right\| \\ & \text{s.t.} & \mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_d}, \\ & & A^{(i)} \in \mathbb{R}^{n_i \times r_i},\ (A^{(i)})^{\mathrm{T}} A^{(i)} = I,\ i = 1, 2, \ldots, d. \end{array}$$

Here, matrices $A^{(i)}$'s are the factor matrices. Without loss of generality, these matrices are assumed to be columnwise orthogonal. The problem can be considered as a generalization of the best rank-one approximation problem, namely the case of $r_1 = r_2 = \cdots = r_d = 1$.

For any given matrices $A^{(i)}$'s, it is easy to optimize the objective function of $(T_{\min})$ over $\mathcal{C}$, which has a close-form solution. Therefore, one easily verifies that $(T_{\min})$ is equivalent to the following maximization model (see the discussion in [25, 15, 1], e.g., page 487 of [25]):

$$\begin{array}{lll} (T_{\max}) & \max & \left\| \mathcal{F} \times_1 (A^{(1)})^{\mathrm{T}} \times_2 (A^{(2)})^{\mathrm{T}} \cdots \times_d (A^{(d)})^{\mathrm{T}} \right\| \\ & \text{s.t.} & A^{(i)} \in \mathbb{R}^{n_i \times r_i},\ (A^{(i)})^{\mathrm{T}} A^{(i)} = I,\ i = 1, 2, \ldots, d. \end{array}$$

The workhorse for solving $(T_{\max})$ or $(T_{\min})$ has been traditionally the ALS method. However, the convergence of the ALS method is not guaranteed in general.

3.2 The MBI method

Chen et al. [13] proposed the so-called MBI method, a greedy-type search algorithm for optimization model with general separable structure

$$\begin{array}{lll} (G) & \max & f(x^1, x^2, \ldots, x^d) \\ & \text{s.t.} & x^i \in S^i \subseteq \mathbb{R}^{n_i},\ i = 1, 2, \ldots, d, \end{array}$$

where $f : \mathbb{R}^{n_1 + n_2 + \cdots + n_d} \to \mathbb{R}$ is a general continuous function. Assuming that for any fixed $d - 1$ blocks of variables x^i's, optimization over one block of variables is easy, the MBI method is guaranteed to converge to a stationary point under the mild condition that S^i is compact for $i = 1, 2, \ldots, d$. We notice that $(T_{\max})$ is a particular instance of (G). Moreover, by fixing any $d-1$ blocks $A^{(i)}$'s in $(T_{\max})$, the optimization subroutine required by the MBI method can be easily solved by the singular value decomposition (SVD).

To be specific, the subproblem of $(T_{\max})$ required by the MBI method is

$$\begin{array}{lll} (T^i_{\max}) & \max & \left\| \mathcal{F} \times_1 (A^{(1)})^{\mathrm{T}} \times_2 (A^{(2)})^{\mathrm{T}} \cdots \times_d (A^{(d)})^{\mathrm{T}} \right\| \\ & \text{s.t.} & A^{(i)} \in \mathbb{R}^{n_i \times r_i},\ (A^{(i)})^{\mathrm{T}} A^{(i)} = I, \end{array}$$

where the matrices $A^{(1)}, A^{(2)}, \ldots, A^{(i-1)}, A^{(i+1)}, \ldots, A^{(d)}$ are given. The objective function of $(T^i_{\max})$ can be written in the matrix form as follows:

$$\left\| (A^{(i)})^{\mathrm{T}} F_{(i)} \left(A^{(d)} \otimes \cdots \otimes A^{(i+1)} \otimes A^{(i-1)} \otimes \cdots \otimes A^{(1)} \right) \right\|.$$

Therefore, the optimal solution for $(T^i_{\max})$ is the r_i leading left singular vectors of the matrix $F_{(i)} \left(A^{(d)} \otimes \cdots \otimes A^{(i+1)} \otimes A^{(i-1)} \otimes \cdots \otimes A^{(1)} \right)$. Let us now present our algorithm for solving $(T_{\max})$.

Algorithm 1 *The MBI method for Tucker decomposition*

Input *Tensor* $\mathcal{F} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ *and scalars* $r_1, r_2, \ldots, r_d$.

Output *Core tensor* $\mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_d}$ *and matrices* $A^{(i)} \in \mathbb{R}^{n_i \times r_i}$ *for* $i = 1, 2, \ldots, d$.

0 *Choose an initial feasible solution* $(A_0^{(1)}, A_0^{(2)}, \ldots, A_0^{(d)})$ *and compute initial objective value* $v_0 := \left\| \mathcal{F} \times_1 (A_0^{(1)})^{\mathrm{T}} \times_2 (A_0^{(2)})^{\mathrm{T}} \cdots \times_d (A_0^{(d)})^{\mathrm{T}} \right\|$. *Set* $k := 0$.

1 *For each* $i = 1, 2, \ldots, d$, *compute* $B_{k+1}^{(i)}$ *consisting of the* r_i *leading left singular vectors of* $F_{(i)} \left(A_k^{(d)} \otimes \cdots \otimes A_k^{(i+1)} \otimes A_k^{(i-1)} \otimes \cdots \otimes A_k^{(1)} \right)$, *and let*

$$w_{k+1}^i := \left\| \mathcal{F} \times_1 (A_k^{(1)})^{\mathrm{T}} \cdots \times_{i-1} (A_k^{(i-1)})^{\mathrm{T}} \times_i (B_{k+1}^{(i)})^{\mathrm{T}} \times_{i+1} (A_k^{(i+1)})^{\mathrm{T}} \cdots \times_d (A_k^{(d)})^{\mathrm{T}} \right\|.$$

2 *Let* $v_{k+1} := \max_{1 \le i \le d} w_{k+1}^i$ *and choose one* $i^* = \arg\max_{1 \le i \le d} w_{k+1}^i$, *and further denote*

$$A_{k+1}^{(i)} := \begin{cases} A_k^{(i)} & i \in \{1, 2, \ldots, d\} \backslash \{i^*\}, \\ B_{k+1}^{(i)} & i = i^*. \end{cases}$$

3 *If* $|v_{k+1} - v_k| \le \epsilon$, *stop and output the core tensor*

$$\mathcal{C} := \mathcal{F} \times_1 (A_{k+1}^{(1)})^{\mathrm{T}} \times_2 (A_{k+1}^{(2)})^{\mathrm{T}} \cdots \times_d (A_{k+1}^{(d)})^{\mathrm{T}}$$

and matrices $A^{(i)} = A_{k+1}^{(i)}$ *for* $i = 1, 2, \ldots, d$; *otherwise, set* $k := k + 1$ *and go to Step 1.*

According to the convergence property of the MBI method established in [13], Algorithm 1 converges to a stationary point for Tucker decomposition. Though computing all possible improvements in Step 1 may be costly, the efforts get paid off well in terms of the total number of iterations required. Moreover, a parallel computation scheme is possible, which is shown to significantly shorten the overall computational time; see [29].

4 Tucker decomposition with unspecified size of the core

In this section, we propose a new model for Tucker decomposition without pre-specifying the size of the core tensor. The dimension of each mode for the core is no longer a constant. Rather it is a variable that also needs to be optimized, which is a key ingredient in our model. Several algorithms are proposed to solve the new model as well.

4.1 The formulation

Given a tensor $\mathcal{F} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, the goal is to find a small-sized (low rank) core tensor $\mathcal{C}$ and d slim matrices $A^{(i)}$'s to express $\mathcal{F}$, as close as possible. We are interested in determining the rank of each mode of $\mathcal{C}$ as well as the best approximation of $\mathcal{F}$. Let the i-rank of $\mathcal{C}$ be r_i for $i = 1, 2, \ldots, d$. Clearly, we have $1 \le r_i \le n_i$ for $i = 1, 2, \ldots, d$. Unlike the general Tucker decomposition, r_i's are now decision variables which need to be determined. Denote c to be a given constant for the

summation of all i-rank variables, i.e., $\sum_{i=1}^{d} r_i = c$, which in general prevents r_i from being too large. The idea behind this constraint is that we would like to have a Tucker approximation which is overall "low ranked", but the specific allocation of the ranks is allowed to be flexible. To determine the allocation for r_i's in the total number c, the new model is

$$\begin{array}{lll} (NT_{\min}) & \min & \left\|\mathcal{F} - \mathcal{C} \times_1 A^{(1)} \times_2 A^{(2)} \cdots \times_d A^{(d)}\right\| \\ & \text{s.t.} & \mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_d}, \\ & & A^{(i)} \in \mathbb{R}^{n_i \times r_i},\ i = 1, 2, \ldots, d, \\ & & r_i \in \mathbb{Z},\ 1 \le r_i \le n_i,\ i = 1, 2, \ldots, d, \\ & & \sum_{i=1}^{d} r_i = c. \end{array}$$

It is difficult to solve $(NT_{\min})$ directly, since the first two constraints of $(NT_{\min})$ combine the block variables $A^{(i)}$ and i-rank variable r_i together. A straightforward method is to separate these variables, and we introduce d more block variables $Y^{(i)} \in \mathbb{R}^{m_i \times m_i}$ where $m_i := \min\{n_i, c\}$ for $i = 1, 2, \ldots, d$ and

$$Y^{(i)} = \operatorname{diag}(\mathbf{y}^{(i)}),\ \mathbf{y}^{(i)} \in \{0,1\}^{m_i},\ \text{and}\ \sum_{j=1}^{m_i} y_j^{(i)} = r_i \text{ for } i = 1, 2, \ldots, d.$$

By the equivalence between Tucker decomposition models $(T_{\min})$ and $(T_{\max})$, we may reformulate $(NT_{\min})$ as follows:

$$\begin{array}{lll} (NT_{\max}) & \max & \left\|\mathcal{F} \times_1 \left(A^{(1)}Y^{(1)}\right)^{\mathrm{T}} \times_2 \left(A^{(2)}Y^{(2)}\right)^{\mathrm{T}} \cdots \times_d \left(A^{(d)}Y^{(d)}\right)^{\mathrm{T}}\right\| \\ & \text{s.t.} & A^{(i)} \in \mathbb{R}^{n_i \times m_i},\ (A^{(i)})^{\mathrm{T}} A^{(i)} = I,\ i = 1, 2, \ldots, d, \\ & & \mathbf{y}^{(i)} \in \{0,1\}^{m_i},\ i = 1, 2, \ldots, d, \\ & & \sum_{j=1}^{m_i} y_j^{(i)} \ge 1,\ i = 1, 2, \ldots, d, \\ & & \sum_{i=1}^{d} \sum_{j=1}^{m_i} y_j^{(i)} = c. \end{array}$$

Throughout this paper, $Y^{(i)}$ denotes $\operatorname{diag}(\mathbf{y}^{(i)})$. Note that r_i in $(NT_{\min})$ is already replaced by the number of nonzero entries of $\mathbf{y}^{(i)}$ in $(NT_{\max})$. Let $\mathcal{X} = \mathcal{F} \times_1 \left(A^{(1)}Y^{(1)}\right)^{\mathrm{T}} \times_2 \left(A^{(2)}Y^{(2)}\right)^{\mathrm{T}} \cdots \times_d \left(A^{(d)}Y^{(d)}\right)^{\mathrm{T}} \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_d}$. If feasible block variables $Y^{(i)}$'s satisfy

$$Y^{(i)} = \operatorname{diag}(\underbrace{1, 1, \ldots, 1}_{r_i}, \underbrace{0, 0, \ldots, 0}_{m_i - r_i}),\ i = 1, 2, \ldots, d, \tag{1}$$

then the size of the core tensor $\mathcal{X}$ can be reduced to $r_1 \times r_2 \times \cdots \times r_d$ by deleting the tails of zero entries in all modes, and the rank of $\mathcal{X}$ in each mode is equal to $r_1, r_2, \ldots, r_d$, respectively. This observation, however, establishes the equivalence between $(NT_{\min})$ and $(NT_{\max})$. As we shall see in the next subsection, we may without loss of generality assume $Y^{(i)}$ to be in the form of (1). This allows us to construct a core tensor with rank-$(r_1, r_2, \ldots, r_d)$ easily, which is exactly the same size of the core tensor $\mathcal{C}$ to be optimized in $(NT_{\min})$. Besides, we would like to remark that the third constraint $\sum_{j=1}^{m_i} y_j^{(i)} \ge 1$ for $i = 1, 2, \ldots, d$ in $(NT_{\max})$ is actually redundant. The reason is that if $\sum_{j=1}^{m_i} y_j^{(i)} = 0$ for some i, then clearly the objective is zero, which can never be optimal. However, we still keep it in $(NT_{\max})$ for consistency purpose in implementing the algorithms discussed in the following subsections.

4.2 The penalty method

Let us focus on the model for Tucker decomposition with unspecified size of the core in the maximization form ($NT_{\max}$). Recall that the separable structure of (G) is required to implement the MBI method. Therefore, we move the nonseparable constraint $\sum_{i=1}^{d}\sum_{j=1}^{m_i} y_j^{(i)} = c$ to the objective function, i.e., the following penalty function

$$\begin{aligned} & p\left(\lambda, A^{(1)}, A^{(2)}, \ldots, A^{(d)}, Y^{(1)}, Y^{(2)}, \ldots, Y^{(d)}\right) \\ := & \left\| \mathcal{F} \times_1 \left(A^{(1)}Y^{(1)}\right)^{\mathrm{T}} \times_2 \left(A^{(2)}Y^{(2)}\right)^{\mathrm{T}} \times_3 \cdots \times_d \left(A^{(d)}Y^{(d)}\right)^{\mathrm{T}} \right\|^2 - \lambda \left(\sum_{i=1}^{d}\sum_{j=1}^{m_i} y_j^{(i)} - c \right)^2, \end{aligned}$$

where $\lambda > 0$ is a penalty parameter. The penalty model for ($NT_{\max}$) is then

$$\begin{array}{lll} (PT) & \max & p\left(\lambda, A^{(1)}, A^{(2)}, \ldots, A^{(d)}, Y^{(1)}, Y^{(2)}, \ldots, Y^{(d)}\right) \\ & \text{s.t.} & A^{(i)} \in \mathbb{R}^{n_i \times m_i},\ (A^{(i)})^{\mathrm{T}} A^{(i)} = I,\ i = 1, 2, \ldots, d, \\ & & \mathbf{y}^{(i)} \in \{0,1\}^{m_i},\ i = 1, 2, \ldots, d, \\ & & \sum_{j=1}^{m_i} y_j^{(i)} \geq 1,\ i = 1, 2, \ldots, d. \end{array}$$

We are ready to apply the MBI method to solve (PT) since the block constraints are now separable. Before presenting the formal algorithm, let us first discuss the subproblems in implementing the MBI method, which have to be solved globally as a requirement to guarantee convergence. Without loss of generality, we wish to optimize $(A^{(1)}, Y^{(1)})$ while all other block variables $(A^{(i)}, Y^{(i)})$ for $i = 2, 3, \ldots, d$ are fixed, i.e.,

$$\begin{array}{lll} (PT^1) & \max & \left\| \left(A^{(1)}Y^{(1)}\right)^{\mathrm{T}} W^{(1)} \right\|^2 - \lambda \left(\sum_{j=1}^{m_1} y_j^{(1)} + \bar{c}_1 \right)^2 \\ & \text{s.t.} & A^{(1)} \in \mathbb{R}^{n_1 \times m_1},\ (A^{(1)})^{\mathrm{T}} A^{(1)} = I, \\ & & \mathbf{y}^{(1)} \in \{0,1\}^{m_1}, \\ & & \sum_{j=1}^{m_1} y_j^{(1)} \geq 1, \end{array}$$

where $W^{(1)} := F_{(1)}\left(A^{(d)}Y^{(d)} \otimes \cdots \otimes A^{(2)}Y^{(2)}\right)$ and $\bar{c}_1 := \sum_{i=2}^{d}\sum_{j=1}^{m_i} y_j^{(i)} - c$.

This subproblem indeed can be solved easily. First, as the optimization of $A^{(1)}$ is irrelevant to the penalty term, the optimal $A^{(1)}$ is the m_1 leading left singular vectors of matrix $W^{(1)}$ by applying SVD. Next, we search for the optimal $Y^{(1)}$ for given optimal $A^{(1)}$. Denote $\mathbf{v}_j$ to be the j-th row vector of matrix $(A^{(1)})^{\mathrm{T}} W^{(1)}$ for $j = 1, 2, \ldots, m_1$, and we have

$$\left\| \left(A^{(1)}Y^{(1)}\right)^{\mathrm{T}} W^{(1)} \right\|^2 = \sum_{j=1}^{m_1} y_j^{(1)} \|\mathbf{v}_j\|^2.$$

Therefore, the optimization of $Y^{(1)}$ is then the following problem:

$$\begin{array}{ll} \max & -\lambda \left(\sum_{j=1}^{m_1} y_j^{(1)} \right)^2 + \sum_{j=1}^{m_1} (\|\mathbf{v}_j\|^2 - 2\lambda\bar{c}_1) y_j^{(1)} - \lambda(\bar{c}_1)^2 \\ \text{s.t.} & \mathbf{y}^{(1)} \in \{0,1\}^{m_1},\ \sum_{j=1}^{m_1} y_j^{(1)} \geq 1. \end{array}$$

Although the above model appears to be combinatorial, it is solvable in polynomial-time. This is because all the possible values for $\sum_{j=1}^{m_1} y_j^{(1)}$ are $\{1, 2, \ldots, m_1\}$, and for any fixed $\sum_{j=1}^{m_1} y_j^{(1)}$, the

optimal allocation for $\mathbf{y}^{(1)}$ can be assigned greedily as the objective function is now linear. Thus we only need to try m_1 different values for $\sum_{j=1}^{m_1} y_j^{(1)}$ and pick the best solution.

In the above discussion, we know that the optimal $Y^{(1)}$ automatically satisfies the formation (1). This is because $A^{(1)}$ is the m_1 leading left singular vectors of matrix $W^{(1)}$, leading to the non-increasing order for $\|\mathbf{v}_j\|$'s. Therefore we can update $A^{(1)}$ and $Y^{(1)}$ simultaneously in solving the subproblem of the MBI method for given penalty parameter λ. To summarize, the whole procedure for solving $(NT_{\max})$ using the MBI method and penalty function method (cf. [5, 34]) is as follows.

Algorithm 2 *The MBI method for the penalty model*

Input *Tensor* $\mathcal{F} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ *and integer* $c \geq d$.

Output *Core tensor* $\mathcal{C} \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_d}$ *and matrices* $A^{(i)} \in \mathbb{R}^{n_i \times m_i}$ *for* $i = 1, 2, \ldots, d$.

0 *Choose parameters* $\lambda_0 > 0$, $\sigma > 1$ *and an initial solution* $(A_0^{(1)}, A_0^{(2)}, \ldots, A_0^{(d)}, Y_0^{(1)}, Y_0^{(2)}, \ldots, Y_0^{(d)})$, *and compute initial objective value* $v_0 := p\Big(\lambda_0, A_0^{(1)}, A_0^{(2)}, \ldots, A_0^{(d)}, Y_0^{(1)}, Y_0^{(2)}, \ldots, Y_0^{(d)}\Big)$. *Set* $k := 0$, $\ell := 0$ *and* $\lambda := \lambda_0$.

1 *For each* $i = 1, 2, \ldots, d$, *solve* (PT^i) *and get its optimal solution* $\left(B_{k+1}^{(i)}, Z_{k+1}^{(i)}\right)$ *with optimal value* w_{k+1}^i, *where* (PT^i) *is defined similar as* (PT^1) *by replacing block 1 by block* i.

2 *Let* $v_{k+1} := \max_{1 \leq i \leq d} w_{k+1}^i$ *and choose one* $i^* = \arg\max_{1 \leq i \leq d} w_{k+1}^i$, *and further denote*

$$\left(A_{k+1}^{(i)}, Y_{k+1}^{(i)}\right) := \begin{cases} \left(A_k^{(i)}, Y_k^{(i)}\right) & i \in \{1, 2, \ldots, d\} \backslash \{i^*\}, \\ \left(B_{k+1}^{(i)}, Z_{k+1}^{(i)}\right) & i = i^*. \end{cases}$$

3 *If* $|v_{k+1} - v_k| \leq \epsilon$, *go to Step 4; otherwise, set* $k := k + 1$ *and go to Step 1.*

4 *If* $\lambda \left(\sum_{i=1}^d \sum_{j=1}^{m_i} (y_{k+1}^{(i)})_j - c\right)^2 \leq \epsilon_0$, *stop and output the core tensor*

$$\mathcal{C} := \mathcal{F} \times_1 \left(A_{k+1}^{(1)} Y_{k+1}^{(1)}\right)^{\mathrm{T}} \times_2 \left(A_{k+1}^{(2)} Y_{k+1}^{(2)}\right)^{\mathrm{T}} \times_3 \cdots \times_d \left(A_{k+1}^{(d)} Y_{k+1}^{(d)}\right)^{\mathrm{T}}$$

and matrices $A^{(i)} = A_{k+1}^{(i)} Y_{k+1}^{(i)}$ *for* $i = 1, 2, \ldots, d$; *otherwise, set* $\ell := \ell + 1$, $\lambda := \lambda_0 \sigma^\ell$ *and* $k := 0$, *and go to Step 1.*

We remark that when Algorithm 2 stops, we can shrink the size of the core tensor to $r_1 \times r_2 \times \cdots \times r_d$ by deleting the tails of zero entries in all modes, where r_i is the number of nonzero entries of $\mathbf{y}^{(i)}$. As an MBI method, Algorithm 2 is guaranteed to converge to a stationary point of (PT), as claimed in [13].

4.3 A heuristic method

To further save the computational efforts, in this subsection we present a heuristic approach to solve Tucker decomposition model with unspecified size of the core, i.e., $(NT_{\max})$. We know that if each i-rank $(i = 1, 2, \ldots, d)$ of the core tensor $\mathcal{C}$ is given, then the problem becomes the traditional Tucker decomposition discussed in Section 3, which can be solved by the MBI method (Algorithm 1). From the discussion in Section 4, we notice that the key issue of the model is to allocate the constant number c to each i-rank of the core tensor. Therefore, the optimal value of $(T_{\max})$ can be considered as a function of $(r_1, r_2, \ldots, r_d)$, denoted by $g(r_1, r_2, \ldots, r_d)$. Our heuristic approach tries to find a distribution of the constant c, by adapting the idea of the MBI method. Specifically, we may start with lower i-ranks, e.g. $r_1^0 = r_2^0 = \cdots = r_d^0 = 1$, and compute Tucker decomposition using Algorithm 1; then we increase one of the i-ranks $(1 \le i \le d)$ by one, by choosing i as the best Tucker decomposition among d possible increment of the i-rank, i.e.,

$$(r_1^{k+1}, r_2^{k+1}, \ldots, r_d^{k+1}) = \arg \max_{1 \le i \le d} g(r_1^k, r_2^k, \ldots, r_{i-1}^k, r_i^k + 1, r_{i+1}^k, \ldots, r_d^k).$$

This procedure is continued until $\sum_{i=1}^d r_i^k = c$ for some k $(= c - d)$.

If the function $g(r_1, r_2, \ldots, r_d)$ for Tucker decomposition can be globally solved for any given $(r_1, r_2, \ldots, r_d)$ (though it is NP-hard in general), then the above approach is a dynamic program and the optimality of $(NT_{\max})$ is guaranteed when the method stops. Though in general Algorithm 1 can only find the stationary solution of $g(r_1, r_2, \ldots, r_d)$, this heuristic approach is in fact very effective; see the numerical tests in Section 5. Apart from the rank increasing approach, rank decreasing strategy can be the other alternative, i.e., starting from large number r_i's and decreasing them until $\sum_{i=1}^d r_i = c$. We conclude this subsection by presenting the heuristic algorithm for the rank decreasing approach below, which is actually very easy to implement.

Algorithm 3 *The rank decreasing method*

Input *Tensor* $\mathcal{F} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ *and integer* $c \ge d$.

Output *Core tensor* $\mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_d}$ *and matrices* $A^{(i)} \in \mathbb{R}^{n_i \times r_i}$ *for* $i = 1, 2, \ldots, d$.

0 *Choose initial ranks* $(r_1, r_2, \ldots, r_d)$ *with* $\sum_{i=1}^d r_i > c$.

1 *Apply Algorithm 1 to solve* $(T_{\max})$ *with input* $(r_1, r_2, \ldots, r_d)$, *and output matrices* $A^{(i)}$ *for* $i = 1, 2, \ldots, d$.

2 *For each* $i = 1, 2, \ldots, d$, *let* $B^{(i)} \in \mathbb{R}^{n_i \times (r_i - 1)}$ *be the matrix by deleting the* r_i*-th column of* $A^{(i)}$, *and compute*

$$i^* := \arg \max_{1 \le i \le d} \left\| \mathcal{F} \times_1 (A^{(1)})^{\mathrm{T}} \cdots \times_{i-1} (A^{(i-1)})^{\mathrm{T}} \times_i (B^{(i)})^{\mathrm{T}} \times_{i+1} (A^{(i+1)})^{\mathrm{T}} \cdots \times_d (A^{(d)})^{\mathrm{T}} \right\|.$$

Update $r_{i^*} := r_{i^*} - 1$ *and* $A^{(i^*)} := B^{(i^*)}$. *Repeat if necessary until* $\sum_{i=1}^d r_i = c$.

3 *Apply Algorithm 1 to solve* $(T_{\max})$ *with input* $(r_1, r_2, \ldots, r_d)$, *and output the core tensor* $\mathcal{C}$ *and matrices* $A^{(i)}$ *for* $i = 1, 2, \ldots, d$.

---

5 Numerical experiments

In this section, we apply the algorithms proposed in the previous sections to solve Tucker decomposition without specification of the core size. Four different types of data are tested to demonstrate the effectiveness and efficiency of our algorithms. All the numerical computations are conducted in an Intel Xeon CPU 3.40GHz computer with 8GB RAM. The supporting software is MATLAB 7.12.0 (R2011a) as a platform. We use MATLAB Tensor Toolbox Version 2.5 [4] whenever tensor operations are called, and also apply its embedded algorithm (the ALS method) to solve Tucker decomposition with given rank-$(r_1, r_2, \ldots, r_d)$. The termination precision for the ALS method is set to be 10^{-4}.

In implementing Algorithm 2, the penalty increment parameter is set to be $\sigma := 2$. The starting matrices $A_0^{(i)}$'s are randomly generated and then made to be orthonormal. The starting diagonal matrices $Y_0^{(i)}$'s are all set to be

$$Y_0^{(i)} := \mathrm{diag}\,(1, \underbrace{0, 0, \ldots, 0}_{m_i - 1}) \text{ for } i = 1, 2, \ldots, d.$$

The termination precision for Algorithm 2 is also set to be 10^{-4}.

For Algorithm 3, the initial ranks are set to be $r_i := \min(n_i, c)$ for $i = 1, 2, \ldots, d$. When applying Algorithm 1 in Step 1, the starting matrices $A_0^{(i)}$'s for solving $(T_{\max})$ are randomly generated, and the termination precision is set to be 10^{-2}. While applying Algorithm 1 in Step 3, the starting matrices $A_0^{(i)}$'s for solving $(T_{\max})$ are obtained from Step 2, and the termination precision is set to be 10^{-4}.

For the given data tensor $\mathcal{F}$ in the following tests, and the rank-$(r_1, r_2, \ldots, r_d)$ approximation tensor $\hat{\mathcal{F}}$ computed by the algorithms, the relative square error of the approximation is normally defined as $\|\mathcal{F} - \hat{\mathcal{F}}\|/\|\mathcal{F}\|$. To measure how close it is to the original tensor $\mathcal{F}$, the term *fit* is defined as

$$\text{fit} := 1 - \frac{\|\mathcal{F} - \hat{\mathcal{F}}\|}{\|\mathcal{F}\|}.$$

The larger the fit, the better the performance.

5.1 Noisy tensor decomposition

First we present some preliminary test results on two synthetic data sets. We use Tensor Toolbox to randomly generated a noise-free tensor $\mathcal{Y} \in \mathbb{R}^{50\times 50\times 30}$ with rank-$(4, 4, 2)$, where entries of the core tensor follow standard normal distributions and entries of the orthogonal factor matrices follow uniform distributions. A noise tensor $\mathcal{N} \in \mathbb{R}^{50\times 50\times 30}$ in the same size is randomly generated, whose entries follow standard normal distributions. The noisy tensor $\mathcal{Z}$ to be tested is then constructed as follows:

$$\mathcal{Z} = \mathcal{Y} + \eta \frac{\|\mathcal{Y}\|}{\|\mathcal{N}\|}\mathcal{N},$$

where η is the noise parameter, more formally, the perturbed ratio. Our task is to find the real i-ranks of the core tensor from the corrupted tensor $\mathcal{Z}$.

In this set of tests, the initial penalty parameter for Algorithm 2 is set to be $\lambda_0 = 0.005\|\mathcal{Z}\|$, and summation of the ranks is set to be $c = 10$. Results of this experiment are summarized in Figure 1, representing the fit and the computational time of Algorithms 2 and 3 for different perturbed ratios. For comparison, we also call the ALS method for solving the traditional Tucker decomposition with given (r_1, r_2, r_3), which are randomly generated satisfying $r_1 + r_2 + r_3 = 10$.

Also, 10 random rank samples are generated for the ALS method, and its maximum fit, average fit and minimum fit are presented in the left of Figure 1, corresponding to three pentagon dots from the top to the bottom in each line segment, respectively. The computational time for the ALS method is then the summation of these 10 rank samples. We observe that Algorithms 2 and 3 can easily and quickly find the exact i-ranks $(4, 4, 2)$ of the core tensor for different perturbed ratios, and can still approximate the original tensor $\mathcal{Y}$ accurately even for large noise. Figure 1 shows that Algorithms 2 and 3 outperform the ALS method significantly, both in fit and in computational time, implying the importance of selecting the core with a right configuration.

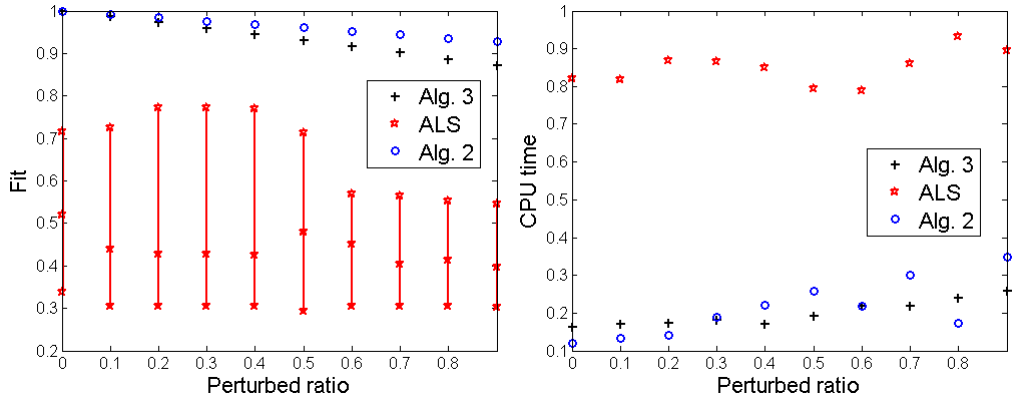

Figure 1: Approximating a noisy tensor of size $50 \times 50 \times 30$ with original rank-(4,4,2).

Figure 2 shows another synthetic experiment whose data is generated similarly as that in Figure 1. In this set of data, $\mathcal{Y} \in \mathbb{R}^{100\times100\times50}$ is a rank-$(5, 5, 4)$ tensor. The summation of the ranks in testing Algorithms 2 and 3 are set as $c = 14$. Figure 2 again shows that our algorithms perform quite well in anti-interference and are far superior to the ALS method, which again illustrates that selecting a suitable core size is necessary. Furthermore, both Algorithms 2 and 3 can find the exact i-ranks $(5, 5, 4)$ of the core tensor for various perturbed ratios.

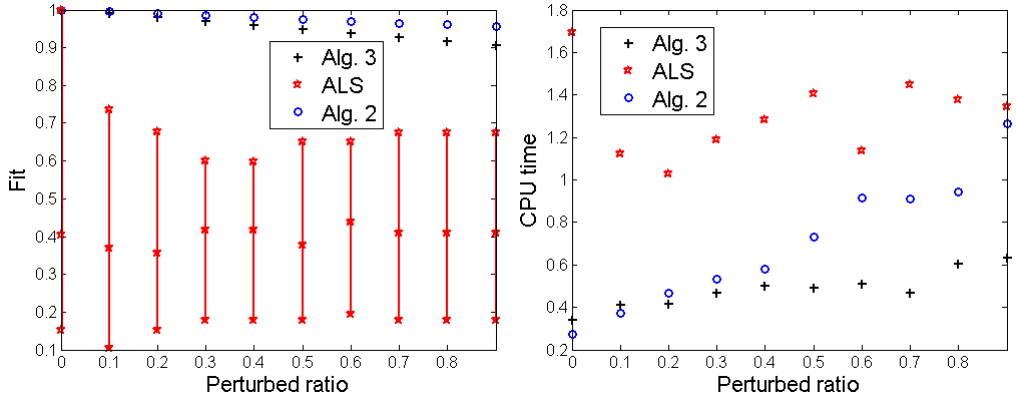

Figure 2: Approximating a noisy tensor of size $100 \times 100 \times 50$ with original rank-(5,5,4).

5.2 Amino acid fluorescence data

This data set was originally generated and measured by Claus Andersson and was later published and tested by Bro [6, 7]. It consists of five laboratory-made samples. Each sample contains different

amounts of tyrosine, tryptophan and phenylalanine dissolved in phosphate buffered water. The size of the array $\mathcal{A}$ to be decomposed is $5 \times 201 \times 61$, which corresponds to samples, emission wavelength (250–450nm) and excitation wavelength (240–300nm), respectively. This array is actually a rank-$(5, 201, 61)$ tensor. Ideally the array should be describable with three PARAFAC components, where its fit is equal to 97.44% obtained by Tensor Toolbox. This data can also be approximated well in the sense of Tucker decomposition. In implementing Algorithm 2, the initial penalty parameter is set to be $\lambda_0 = 0.01\|\mathcal{A}\|$.

The numerical results for the three methods (ALS, Algorithms 2 and 3) are presented in Figure 3. The maximum fit and the average fit by running the ALS method 10 times with randomly generated rank-(r_1, r_2, r_3) satisfying $r_1 + r_2 + r_3 = c$ are plotted on the left part of Figure 3, corresponding to the two endpoints of each line segment. The CPU time for ALS is the summation of these 10 random samples. Algorithms 2 and 3 perform better than the ALS method. In particular for Algorithm 2, it can get a convincing fit even for lower c. Furthermore, Algorithm 2 decomposes this data set as good as PARAFAC does without knowing the exact i-ranks, e.g. the fit reaches 97.55% when $c = 9$ which finds rank-$(3, 3, 3)$ when it stops.

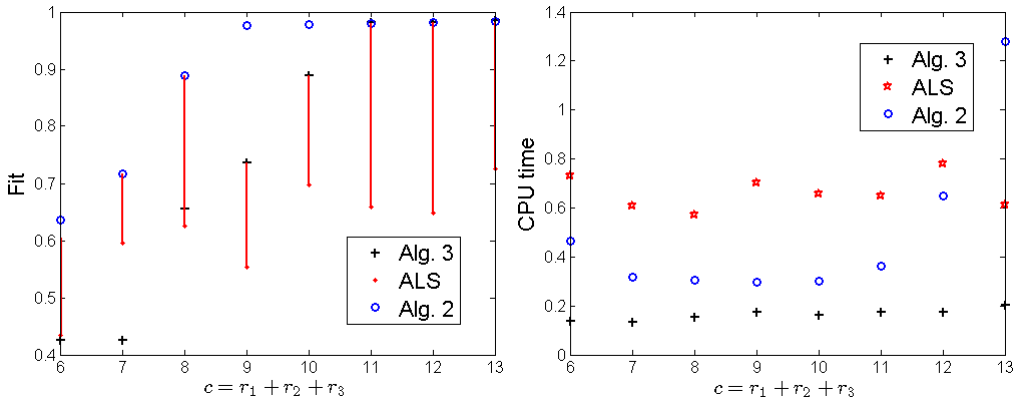

Figure 3: Approximating the amino acid fluorescence data with original rank-$(5, 201, 61)$ for different $c = r_1 + r_2 + r_3$.

To further justify the importance of Tucker decomposition with unspecified size of the core as well as Algorithm 2, here we investigate the ALS method with all possible pre-specified i-ranks to test this data set. For given summation of i-ranks $c = 9$, there are a total of 25 possible combinations of (r_1, r_2, r_3), and their corresponding fits are listed in Figure 4. It shows that the Tucker decomposition of rank-$(3, 3, 3)$ outperforms all other combinations, which is exactly the i-ranks found by Algorithm 2.

5.3 Gene expression data

We further test one real three-way tensor $\mathcal{F} \in \mathbb{R}^{2395 \times 6 \times 9}$, which is from 3D Arabidopsis gene expression data[1]. The i-ranks of the data tensor are $(54, 6, 9)$. Essentially this is a set of gene expression data with 2395 genes, measured at 6 time points and 9 different conditions.

We again apply the three methods to test this data set. The initial penalty parameter for Algorithm 2 is set to be $\lambda_0 = 0.01\|\mathcal{F}\|$. Results of our experiments are summarized in Table 1. For the fit in the last three columns by the ALS method, fit-2 and fit-3 denote the fit by using the ALS method with the ranks obtained from Algorithms 2 and 3, respectively, while the fit in the

[1]We thank Professor Xiuzhen Huang of Arkansas State University for providing us this set of data.

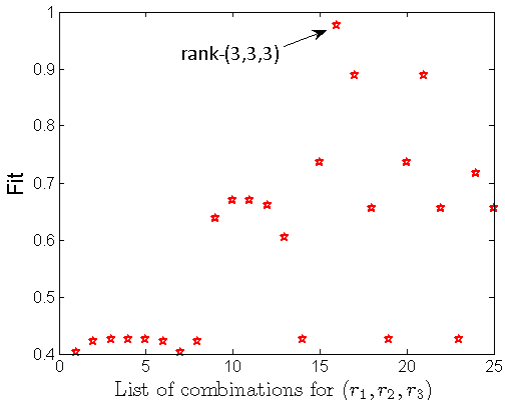

Figure 4: The ALS method on the amino acid fluorescence data for $c = 9$.

Table 1: Approximating the gene expression data with original rank-$(54, 6, 9)$.

c	Algorithm 2			Algorithm 3			ALS		
	(r_1, r_2, r_3)	CPU	Fit(%)	(r_1, r_2, r_3)	CPU	Fit(%)	Fit-2(%)	Fit-3(%)	Fit(%)
3	(1, 1, 1)	0.45	85.85	(1, 1, 1)	1.62	85.96	85.85	85.85	85.85
5	(2, 1, 2)	7.04	86.66	(2, 1, 2)	2.89	87.73	86.75	86.77	86.01
10	(4, 2, 4)	6.16	89.79	(1, 3, 6)	2.33	93.89	89.82	85.96	86.61
15	(7, 3, 5)	9.21	91.96	(5, 6, 4)	1.62	91.44	91.97	90.99	89.15
20	(10, 4, 6)	5.97	93.59	(5, 6, 9)	1.55	91.44	93.60	91.44	92.71

last column is the average fit by running the ALS method 10 times with randomly generated ranks satisfying $r_1 + r_2 + r_3 = c$. The following conclusions can be drawn from this set of experiments:

- Excellent performances of Algorithms 2 and 3 are validated.
- Computing the i-rank information is important, as the ranks produced by Algorithms 2 and 3 are better than the randomly generated ranks for the ALS method in terms of the fit.
- With the combination of the ALS method, Algorithm 2 works better than Algorithm 3 in terms of the fit, while its CPU time is longer than that of Algorithm 3.

5.4 Tensor compression of image data

In this part, we focus on the experiments of the models and algorithms for the image data. Here two sets of faces are experimented, with each in one subsection.

5.4.1 The ORL database of faces

The first set of images is from the ORL database of faces [33] in AT&T Laboratories Cambridge. In this database, there are 10 different images for each of the 40 distinct subjects, and the size of each image is 92×112 pixels. Here, we draw one single distinct subject, and then construct a tensor $\mathcal{T} \in \mathbb{R}^{92\times112\times10}$, whose i-ranks are $(92, 112, 10)$. When applying Algorithm 2, the initial penalty parameter is set as $\lambda_0 = 0.005\|\mathcal{T}\|$.

Table 2: Tensor compression for the ORL database of faces with original rank-$(92, 112, 10)$.

c	Methods	(r_1, r_2, r_3)	CR	CP(%)	RMSE	Fit(%)	CPU
50	Algorithm 3	(25, 22, 3)	62.4	98.4	143.98	83.99	0.71
	ALS (best)	(16, 25, 9)	28.6	96.5	75.25	87.64	1.15
	ALS (worst)	(43, 6, 1)	399.4	99.8	556.85	75.52	
	Algorithm 2	(21, 19, 10)	25.8	96.1	67.89	88.26	5.80
70	Algorithm 3	(35, 31, 4)	23.7	95.8	72.92	86.85	1.36
	ALS (best)	(34, 29, 7)	14.9	93.3	45.78	89.59	1.20
	ALS (worst)	(11, 58, 1)	161.5	99.4	345.26	76.10	
	Algorithm 2	(30, 30, 10)	11.4	91.3	34.13	91.14	12.35

Figure 5 displays the images compressed and recovered by the three methods when $c = 50$ and 70 respectively. The detailed numerical values for the two cases are listed in Table 2. Some standard abbreviations from image science are adopted, namely, CP for compression, CR for compression ratio, and RMSE for root mean squared error. For the ALS method, its CPU time is computed by the summation of 10 random generated i-ranks satisfying $r_1 + r_2 + r_3 = c$, and its best and worst compressions in terms of fit are reported for comparison.

Figure 5: Recovered images for the ORL database of faces (rows from the top to the bottom: 1. Original, 2. Algorithm 3, 3. ALS (the best), 4. ALS (the worst), 5. Algorithm 2).

Observation from Figure 5 indicates that only the compressed images by the ALS method (the best one) and Algorithm 2 keep all the facial expressions. However Algorithm 2 indeed outperforms the ALS method (the best one) in terms of fit and RMSE as shown in Table 2. It is worth mentioning that most of computational effort of Algorithm 2 is to find a suitable combination of ranks (r_1, r_2, r_3), while this issue for the ALS method is pre-specified. This computational effort in selecting better initial ranks is worthwhile, as we noticed that the ALS method for a large c may be worse than Algorithm 2 for a smaller c, e.g. the 4th row in the right of Figure 5 is not clearer

Table 3: Tensor compression for the JAFFE database with original rank-$(256, 256, 7)$.

c	Methods	(r_1, r_2, r_3)	CR	CP(%)	RMSE	Fit(%)	CPU
80	Algorithm 3	(40, 39, 1)	294.1	99.7	480.15	80.11	2.50
	ALS (best)	(39, 38, 3)	103.2	99.0	183.30	87.18	2.81
	ALS (worst)	(75, 2, 3)	1.0e+03	99.9	1350.50	69.95	
	Algorithm 2	(39, 34, 7)	49.4	98.0	75.17	92.40	35.67
110	Algorithm 3	(53, 55, 2)	78.7	98.7	185.49	85.14	9.00
	ALS (best)	(54, 52, 4)	40.8	97.6	94.95	89.44	2.50
	ALS (worst)	(1, 105, 4)	1.1e+03	99.9	1958.61	57.89	
	Algorithm 2	(53, 50, 7)	24.7	96.0	43.35	93.81	169.27

than the 2rd row in the left of Figure 5, which is also confirmed by Table 2.

Figure 6 presents the performance of Algorithm 2 for varying c. Clearly, the larger c is, the larger the fit, and the lower the RMSE. This figure also suggests a guideline for choosing a suitable c depending on the quality of the compressed images required by the user.

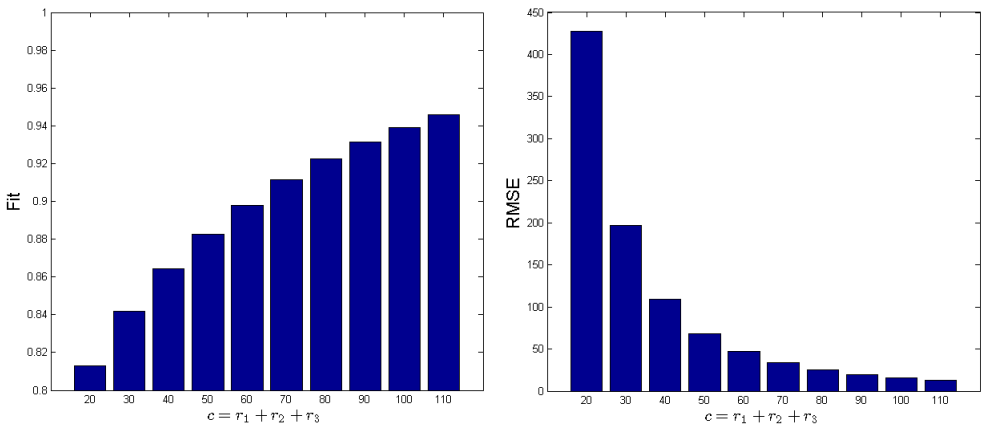

Figure 6: Performance of Algorithm 2 for the ORL database of faces.

5.4.2 The Japanese female facial expression (JAFFE) database

The other tests are on the facial database [30] from the Psychology Department in Kyushu University, which contains 213 images of 7 different emotional facial expressions (sadness, happiness, surprise, anger, disgust, fear and neutral) posed by 10 Japanese female models. Each image has 256×256 pixels. Here we draw 7 different facial expressions of one female model and construct a tensor $\mathcal{T}$ of size $256 \times 256 \times 7$, whose i-ranks are $(256, 256, 7)$. A similar set of tests as in Section 5.4.1 are conducted. The results are presented in Figures 7 and 8, Table 3, and Figure 9, which again confirm the observation from the results for the ORL database of faces. In particular, by comparing the best two set of images (the best ALS method and Algorithm 2) in Figures 7 and 8, Algorithm 2 is clearly better in details, e.g. the eyebrows.

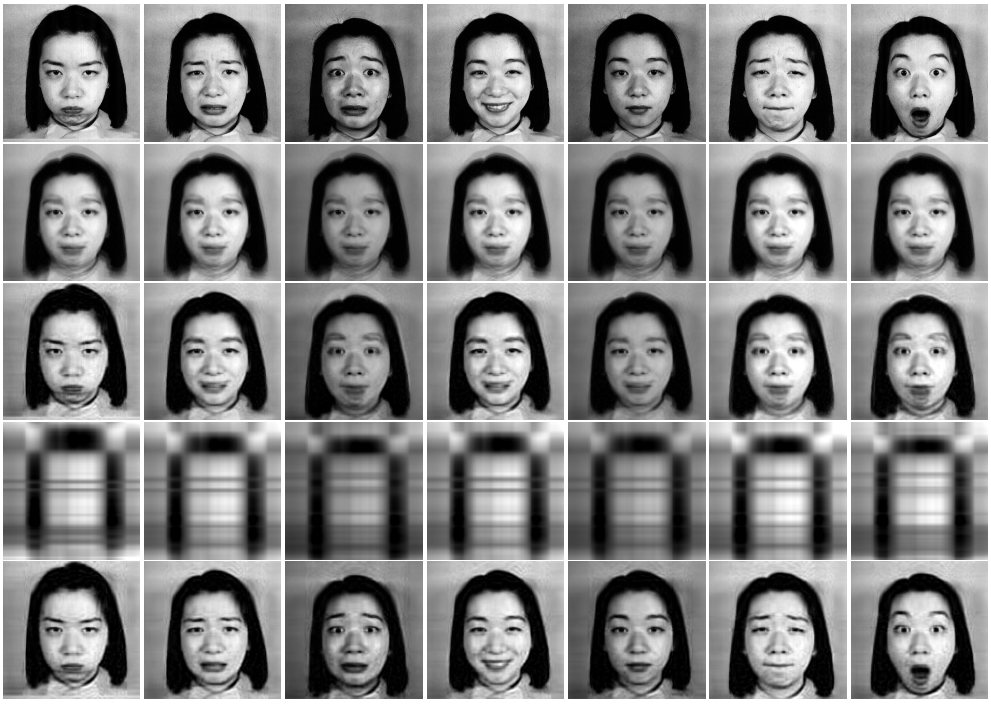

Figure 7: Recovered images for the JAFFE database when $c = 80$ (rows from the top to the bottom: 1. Original, 2. Algorithm 3, 3. ALS (the best), 4. ALS (the worst), 5. Algorithm 2).

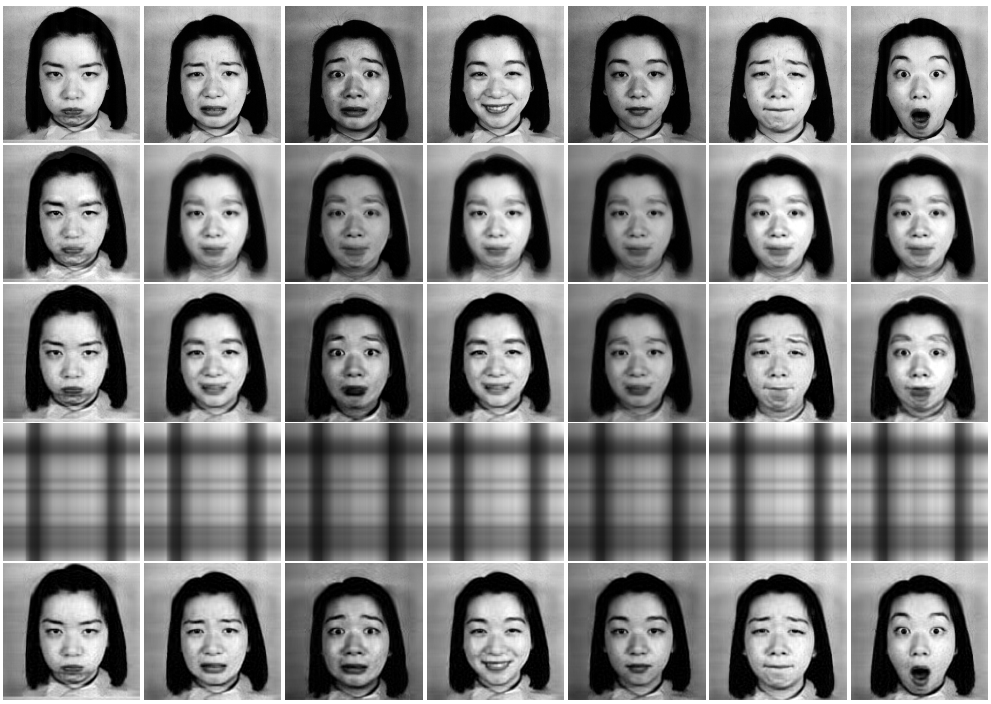

Figure 8: Recovered images for the JAFFE database when $c = 110$ (rows from the top to the bottom: 1. Original, 2. Algorithm 3, 3. ALS (the best), 4. ALS (the worst), 5. Algorithm 2).

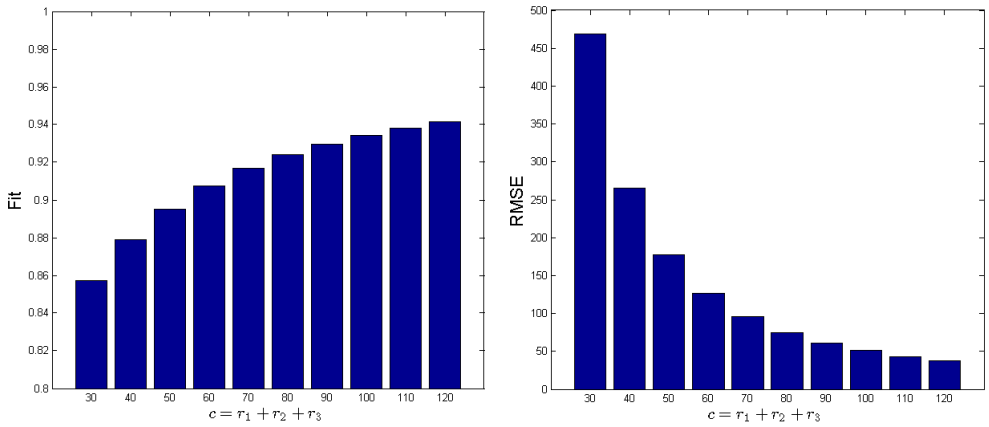

Figure 9: Performance of Algorithm 2 for the JAFFE database.

Table 4: Analysis of the DIFFIT approach for amino acid fluorescence data.

c	(r_1, r_2, r_3)	ExpVar(%)	DIF(%)	DIFFIT	Fit(%)
3	$(1,1,1)$	64.3900	64.3900	3.2278	40.33
5	$(2,2,1)$	84.3384	19.9484	8.1922	60.43
6	$(2,2,2)$	86.7735	2.4351	0.4657	63.63
7	$(3,2,2)$	92.0022	5.2287	0.7746	71.72
8	$(3,3,2)$	98.7522	6.7500	5.6819	88.83
9	$(3,3,3)$	99.9402	1.1880	**103.8363**	97.55
10	$(4,3,3)$	99.9516	0.0114	1.2007	97.80
11	$(4,4,3)$	99.9611	0.0095	1.7685	98.03
12	$(4,4,4)$	99.9665	0.0054	0.9857	98.17
13	$(4,5,4)$	99.9720	0.0055	0.9466	98.33
14	$(5,5,4)$	99.9778	0.0058	1.4935	98.51
15	$(5,5,5)$	99.9816	0.0039	–	98.64

5.5 Comparisons with the DIFFIT and the ARD methods

In our final set of tests, we compare our approaches with those methods that are capable of choosing the rank of the core tensor in the Tucker decomposition, in particular, the DIFFIT procedure [35][2] and the ARD method [31][3]. They are both useful tools in analyzing the data with a low-rank structure. The parameters in the ARD algorithm are set as default. We use two data sets in previous subsections for comparison, the amino acid fluorescence data in Section 5.2 and the noisy tensor of size $50 \times 50 \times 30$ with original rank-(4,4,2) in Section 5.1. Numerical results of the DIFFIT approach are listed in Tables 4 and 5, and that of the ARD method are listed in Tables 6 and 7.

Table 4 presents the results to estimate the core size of Tucker model based on the DIFFIT procedure. Its stopping criterion is based on the explained variance, i.e.,

$$\text{ExpVar} = 1 - \frac{\|\mathcal{F} - \hat{\mathcal{F}}\|^2}{\|\mathcal{F}\|^2}.$$

[2]We thank Professor Marieke Timmerman for providing us some basic codes of the DIFFIT approach.
[3]A Matlab implementation of the ARD method is available at http://www.erpwavelab.org.

Table 5: Comparison with DIFFIT approach for the noisy tensor of size $50 \times 50 \times 30$ with original rank-(4,4,2).

η	DIFFIT						Algorithm 2	
	(n_1, n_2, n_3)	ExpVar(%)	(r_1, r_2, r_3)	DIFFIT	Fit(%)	CPU	Fit(%)	CPU
0.01	$(4, 4, 4)$	99.99	$(4, 4, 2)$	$5.79 \cdot 10^5$	99.00	0.53	99.92	0.09
	$(5, 5, 5)$	99.99	$(4, 4, 2)$	$5.79 \cdot 10^5$	99.00	1.01		
	$(10, 10, 6)$	99.99	$(4, 4, 2)$	$5.79 \cdot 10^5$	99.00	19.22		
0.1	$(4, 4, 4)$	99.02	$(4, 4, 2)$	$5.79 \cdot 10^3$	90.08	0.35	99.20	0.13
	$(5, 5, 5)$	99.02	$(4, 4, 2)$	$5.80 \cdot 10^3$	90.08	0.92		
	$(10, 10, 6)$	99.05	$(4, 4, 2)$	$5.80 \cdot 10^3$	90.08	18.60		
0.2	$(4, 4, 4)$	96.19	$(4, 4, 2)$	$1.45 \cdot 10^3$	80.46	0.39	98.40	0.14
	$(5, 5, 5)$	96.21	$(4, 4, 2)$	$1.45 \cdot 10^3$	80.46	1.20		
	$(10, 10, 6)$	96.31	$(4, 4, 2)$	$1.45 \cdot 10^3$	80.46	20.10		

ExpVar is similar to the definition of *fit* in previous numerical experiments, which is also computed in the last column of Table 4 for reference. In implementing DIFFIT, one needs to increase the value c one by one until the algorithm finds the best combination of (r_1, r_2, r_3) with $r_1 + r_2 + r_3 = c$ and its corresponding ExpVar = 100%, which is time-consuming. For comparison with our method, we only evaluate all combinations up to rank-(5, 5, 5) Tucker decomposition. The best combinations of (r_1, r_2, r_3) satisfying $r_1 + r_2 + r_3 = c$ are listed in each row of Table 4. The DIFFIT procedure correctly identifies rank-$(3, 3, 3)$, and its corresponding fit is 97.55% and the total CPU time is 0.79 seconds. As shown in the tests in Section 5.2, Algorithm 2 could quickly find rank-$(3, 3, 3)$ provided that $c = 9$.

Table 5 presents the results on synthetic data tensors for both the DIFFIT approach and Algorithm 2. For each different perturbed ratio η in the noisy tensor, we evaluate all the combinations of the models up to the rank-(n_1, n_2, n_3) Tucker decomposition in the DIFFIT procedure. Three different trials on (n_1, n_2, n_3)'s are tested for DIFFIT, with its corresponding ExpVar value listed when the DIFFIT approach stops. For different sets of (n_1, n_2, n_3), DIFFIT always finds the best size of the core tensor, which is $(4, 4, 2)$. However, the corresponding fit values are worse than that of Algorithm 2. In order to run Algorithm 2, we need to provide the predefined information on $c = 10$ to find the best rank and its Tucker approximation. The numerical results show that DIFFIT naturally finds the best size of the core tensor from all the possible combinations, while our method needs a predefined sum of the dimensions for the core tensor.

Table 6 presents the analysis of amino acid fluorescence data based on the ARD method. Two types of priors for the parameters in ARD are used, one is the Laplacian priors, which is referred to sparse ARD Tucker analysis in Table 6, and the other is the Gaussian priors, which corresponds to ridge ARD Tucker analysis. In the test, we choose three different initial sizes (n_1, n_2, n_3) of the core tensor, and run the two analyzing approaches 10 times, respectively. All the estimated core sizes (r_1, r_2, r_3) are reported, together with the number of times (denoted by No.) that the algorithm reaches a specific (r_1, r_2, r_3) among all the 10 runs and the likelihood (denoted by Val.) of the best estimated (r_1, r_2, r_3) when the algorithm stops. The best model is given by the one with the largest 'Val.' indicated in bold face. Results in Table 6 indicate that the sparse ARD method performs better than the ridge ARD method in identifying the correct core size $(3, 3, 3)$. The fit value of the

Table 6: Analysis of the ARD method for amino acid fluorescence data.

(n_1, n_2, n_3)	(r_1, r_2, r_3)	Sparse ARD Tucker				Ridge ARD Tucker			
		No.	Val.(10^5)	Fit(%)	CPU	No.	Val.(10^5)	Fit(%)	CPU
$(4, 4, 4)$	$(3, 3, 3)$	5	**-2.9527**	97.31	10.06	4	**-2.9842**	97.33	10.81
	$(4, 3, 3)$	5	-2.9576	97.62	10.14	6	-2.9883	97.69	8.86
$(5, 5, 5)$	$(3, 3, 3)$	2	**-2.8883**	97.30	7.31	2	-2.9563	97.27	9.38
	$(4, 3, 3)$	5	-2.8937	97.52	11.63	6	-2.9526	97.46	11.13
	$(5, 3, 3)$	3	-2.8945	97.56	8.75	2	**-2.9485**	97.40	11.19
$(10, 10, 10)$	$(3, 3, 3)$	4	**-2.5547**	97.36	9.23	2	-2.7860	97.49	12.20
	$(4, 3, 3)$	3	-2.5570	97.49	8.97	6	**-2.7782**	97.51	13.27
	$(5, 3, 3)$	2	-2.5619	97.52	9.79	1	-2.7816	97.58	13.07
	$(6, 3, 3)$	0	–	–	–	1	-2.7888	97.72	14.06
	$(7, 3, 3)$	1	-2.5663	97.62	10.64	0	–	–	–

estimated rank-$(3, 3, 3)$ model is a little bit less than 97.55% of Algorithm 2. Moreover, the ARD method is quite time-consuming compared to Algorithm 2.

For the noisy tensor data, similar test results for the sparse ARD method are presented in Table 7. Here we only report the best estimated rank-(r_1, r_2, r_3) by running the sparse ARD method 10 times. As shown in Table 7, when the size (n_1, n_2, n_3) increases, the sparse ARD method becomes increasingly harder to identify the correct core size. For comparison purpose, we use $c = n_1 + n_2 + n_3$ to test Algorithm 2. In particular when $c = 10$, Algorithm 2 can find the correct core size quickly. This also showed in Figure 1, even though the perturbed ratio η is large. Algorithm 2 is able to find rank-(4, 4, 2) quickly once $c = 10$ is given.

6 Conclusion

In this paper we study the problem of finding a proper Tucker approximation for a general tensor without a pre-specified size of the core tensor, which addresses an important practical issue for real applications. The size of the core tensor is assumed in almost all the existing methods for tensor Tucker decomposition. The approach that we propose is based on the so-called maximum block improvement (MBI) method. Our numerical experiments on a variety of instances taken from real applications suggest that the proposed methods perform robustly and effectively.

Acknowledgements

This work was partially supported by National Science Foundation of China (Grant 11301436 & 11371242), National Science Foundation of USA (Grant CMMI-1161242), Natural Science Foundation of Shanghai (Grant 12ZR1410100), and Ph.D. Programs Foundation of Chinese Ministry of Education (Grant 20123108120002). We would like to thank the anonymous referee for the insightful suggestions.

Table 7: Comparison with sparse ARD Tucker analysis for the noisy tensor of size $50 \times 50 \times 30$ with original rank-(4,4,2).

η	Sparse ARD Tucker						Algorithm 2			
	(n_1, n_2, n_3)	(r_1, r_2, r_3)	No.	Val.(10^5)	Fit(%)	CPU	c	(r_1, r_2, r_3)	Fit(%)	CPU
0	$(4, 4, 2)$	$(4, 4, 2)$	10	3.27	99.76	4.98	10	$(4, 4, 2)$	100.00	0.07
	$(5, 5, 5)$	$(4, 4, 2)$	4	3.32	99.78	6.71	15	$(4, 4, 7)$	100.00	0.09
	$(10, 10, 10)$	$(4, 5, 4)$	1	3.48	99.72	10.94	30	$(24, 4, 2)$	100.00	0.55
0.1	$(4, 4, 2)$	$(4, 4, 2)$	10	3.26	99.16	5.45	10	$(4, 4, 2)$	99.20	0.06
	$(5, 5, 5)$	$(4, 4, 2)$	2	3.31	99.17	6.96	15	$(9, 4, 2)$	99.05	0.09
	$(10, 10, 10)$	$(4, 4, 4)$	1	3.48	98.97	11.63	30	$(24, 4, 2)$	99.05	0.79
0.2	$(4, 4, 2)$	$(4, 4, 2)$	10	3.23	98.39	6.09	10	$(4, 4, 2)$	98.40	0.09
	$(5, 5, 5)$	$(4, 4, 2)$	5	3.28	98.39	8.62	15	$(4, 9, 2)$	98.13	0.13
	(10,10,10)	$(4, 5, 3)$	1	3.47	99.09	12.49	30	$(24, 4, 2)$	98.10	1.04

References

[1] Andersson, C.A., Bro, R.: Improving the speed of multi-way algorithms: Part I. Tucker3. Chemometr. Intell. Lab. 42, 93–103 (1998)

[2] Appellof, C.J., Davidson, E.R.: Strategies for analyzing data from video fluorometric monitoring of liquid chromatographic effluents. Anal. Chem. 53, 2053–2056 (1981)

[3] Aubry, A., De Maio, A., Jiang, B., Zhang, S.: Ambiguity Function Shaping for Cognitive Radar via Complex Quartic Optimization. IEEE Transaction on Signal Processing 61, 5603–5619, (2013)

[4] Bader, B.W., Kolda, T.G.: Matlab Tensor Toolbox, Version 2.5. http://csmr.ca.sandia.gov/~tgkolda/TensorToolbox (2012).

[5] Bertsekas, D.P.: Nonlinear Programming (2nd edition). Athena Scientific, Belmont, MA (1999)

[6] Bro, R.: PARAFAC: Tutorial and applications. Chemometr. Intell. Lab. 38, 149–171 (1997)

[7] Bro, R.: Multi-way Analysis in the Food Industry: Models, Algorithms, and Applications. Ph.D. Thesis, University of Amsterdam, Netherlands, and Royal Veterinary and Agricultural University, Denmark (1998)

[8] Bro, R., Kiers H.A.L.: A new efficient method for determining the number of components in PARAFAC models. J. Chemometr. 17, 274–286 (2003)

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

[10] Ceulemans, E., Kiers, H.A.L.: Selecting among three-way principal component models of different types and complexities: A numerical convex hull based method. Brit. J. Math. Stat. Psy. 59, 133–150 (2006)

[11] Ceulemans, E., Kiers, H.A.L.: Discriminating between strong and weak structures in three-mode principal component analysis. Brit. J. Math. Stat. Psy. 62, 601–620 (2009)

[12] Chen, B.: Optimization with Block Variables: Theory and Applications. Ph.D. Thesis, The Chinese Univesrity of Hong Kong, Hong Kong (2012)

[13] Chen, B., He, S., Li, Z., Zhang, S.: Maximum block improvement and polynomial optimization. SIAM J. Optim. 22, 87–107 (2012)

[14] De Lathauwer, L., De Moor, B., Vandewalle, J.: A multilinear singular value decomposition. SIAM J. Matrix Anal. Appl. 21, 1253–1278 (2000)

[15] De Lathauwer, L., De Moor, B., Vandewalle, J.: On the best rank-1 and rank-$(R_1, R_2, \ldots, R_N)$ approximation of higher-order tensors. SIAM J. Matrix Anal. Appl. 21, 1324–1342 (2000)

[16] De Lathauwer, L., Nion, D.: Decompositions of a higher-order tensor in block terms—Part III: Alternating least squares algorithms. SIAM J. Matrix Anal. Appl. 30, 1067–1083 (2008)

[17] Eldén, L., Savas, B.: A Newton-Grassmann method for computing the best multi-linear rank-$(r1, r2, r3)$ approximation of a tensor, SIAM J. Matrix Anal. Appl. 31, 248–271 (2009)

[18] Harshman, R.A.: Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis. UCLA Working Papers in Phonetics 16, 1–84 (1970). Also available online at http://publish.uwo.ca/~harshman/wpppfac0.pdf

[19] He, Z., Cichocki, A., Xie, S.: Efficient method for Tucker3 model selection. Electron. Lett. 45, 805–806 (2009)

[20] Ishteva, M., De Lathauwer, L., Absil, P.A., Van Huffel, S.: Differential-geometric Newton method for the best rank-(R_1, R_2, R_3) approximation of tensors. Numer. Algorithms 51, 179–194 (2009)

[21] Kapteyn, A., Neudecker, H., Wansbeek, T.: An approach to n-mode components analysis. Psychometrika 51, 269–275 (1986)

[22] Kiers, H.A.L., Der Kinderen, A.: A fast method for choosing the numbers of components in Tucker3 analysis. Brit. J. Math. Stat. Psy. 56, 119–125 (2003)

[23] Kofidis, E., Regalia, P.A.: On the best rank-1 approximation of higher order supersymmetric tensors. SIAM J. Matrix Anal. Appl. 23, 863–884 (2002)

[24] Kolda, T.G.: Multilinear operators for higher-order decompositions. Technical Report SAND2006-2081, Sandia National Laboratories, Albuquerque, NM (2006)

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

[26] Kroonenberg, P.M., De Leeuw, J.: Principal component analysis of three-mode data by means of alternating least squares algorithms. Psychometrika 45, 69–97 (1980)

[27] Leibovici, D., Sabatier, R.: A singular value decomposition of a k-way array for a principal component analysis of multiway data, PTA-k. Linear Algebra Appl. 269, 307–329 (1998)

[28] Levin, J.: Three-mode Factor Analysis. Ph.D. Thesis, University of Illinois, Urbana, IL (1963)

[29] Li, Z., Uschmajew, A., Zhang, S.: On convergence of the maximum block improvement method. Technical Report (2013)

[30] Lyons, M.J., Akamatsu, S., Kamachi, M., Gyoba, J.: Coding facial expressions with gabor wavelets. In: Proceedings of the 3rd IEEE International Conference on Automatic Face and Gesture Recognition, pp. 200–205 (1998)

[31] Mørup M., Hansen, L.K.: Automatic relevance determination for multi-way models. J. Chemometr. 23, 352–363 (2009)

[32] Qi, L.: The best rank-one approximation ratio of a tensor space. SIAM J. Matrix Anal. Appl. 32, 430–442 (2011)

[33] Samaria, F., Harter, A.: Parameterisation of a stochastic model for human face identification. In: Proceedings of 2nd IEEE Workshop on Applications of Computer Vision, pp. 138–142 (1994)

[34] Sun, W., Yuan, Y.: Optimization Theory and Methods: Nonlinear Programming. Springer Optimization and Its Applications, Volume 1, Springer, New York (2006)

[35] Timmerman, M.E., Kiers, H.A.L.: Three-mode principal components analysis: Choosing the numbers of components and sensitivity to local optima. Brit. J. Math. Stat. Psy. 53, 1–16 (2000)

[36] Tomioka, R., Suzuki, T., Hayashi, K., Kashima, H.: Statistical performance of convex tensor decomposition. In: Proceedings of the 25th Annual Conference on Neural Information Processing Systems, pp. 972–980 (2011)

[37] Tucker, L.R.: Implications of factor analysis of three-way matrices for measurement of change. In: Harris C.W. (eds.) Problems in Measuring Change, pp. 122–137. University of Wisconsin Press (1963)

[38] Tucker, L.R.: The extension of factor analysis to three-dimensional matrices. In: Gulliksen H., Frederiksen N. (eds.) Contributions to Mathematical Psychology. Holt, Rinehardt, & Winston, New York, NY (1964)

[39] Tucker, L.R.: Some mathematical notes on three-mode factor analysis. Psychometrika 31, 279–311 (1966)

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

[41] Wang, Y., Qi, L.: On the successive supersymmetric rank-1 decomposition of higher-order supersymmetric tensors. Numer. Linear Algebra Appl. 14, 503–519 (2007)

[42] Zhang, S., Wang, K., Ashby, C., Chen, B., Huang, X.: A unified adaptive co-identification framework for high-D expression data. In: Shibuya, T., et al. (eds.) In: Proceedings of the 7th IAPR International Conference on Pattern Recognition in Bioinformatics, Lecture Notes in Computer Science, vol. 7632, pp. 59–70. Springer, New York (2012)

[43] Zhang, S., Wang, K., Chen, B., Huang, X.: A new framework for co-clustering of gene expression data. In: Loog, M., et al. (eds.) In: Proceedings of the 6th IAPR International Conference on Pattern Recognition in Bioinformatics, Lecture Notes in Computer Science, vol. 7036, pp. 1–12. Springer, New York (2011)

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