Fast Tensor Decompositions for Big Data Processing

(Invited Paper)

Viet-Dung Nguyen*, Karim Abed-Meraim* and Nguyen Linh-Trung†
*PRISME Laboratory, University of Orléans, France
†University of Engineering and Technology, Vietnam National University, Hanoi, Vietnam
{viet-dung.nguyen,karim.abed-meraim}@univ-orleans.fr, linhtrung@vnu.edu.vn

***Abstract*—Tensors, as a natural extension of matrices, and their decompositions provide important tools in many disciplines such as psychometrics, signal processing, data communication, computer vision, and machine learning. The main objective of this paper is to briefly review several recent state-of-the-art approaches for large-scale tensor data which is a crucial part of big data. Moreover, we also introduce our own contributions on this topic.**

***Index Terms*—Tensor decomposition; PARAFAC/CANDECOMP; Tucker; Large-scale processing; Big data; Batch decomposition; Adaptive decomposition.**

I. Introduction

Large volumes of data are being generated at any given time, especially from transactional databases, multimedia content, social media, and applications of sensors in the Internet of Things. When the size of datasets is beyond the ability of typical database software tools to capture, store, manage, and analyze, we face the phenomenon of big data for which new and smarter data analytic tools are required. Big data provides opportunities for new form of data analytics, resulting in substantial productivity.

For datasets collected in a multi-dimensional form, they can be naturally represented by multi-way arrays, which are called tensors[1]. If we consider a vector as a first-order tensor, a matrix as a second-order tensor, we will work with higher-order tensors (of order larger than two) for multiway arrays. In other words, while a matrix is indexed with two indices, a higher-order tensor is a data structure with more than two indices. For example, a three-way tensor can be used to easily store the time-frequency representation of an EEG dataset. The first way is for the "spatial" dimension, storing the locations of the electrodes (a.k.a., channels) that measure the electricity of the brain. The two other dimensions are the time and the frequency which store the time-varying frequency contents of the EEG signal. These tensor decompositions, which reveal different structures/components hidden in the underlying tensors, thereby provide efficient tools to analyze, compress and understand data.

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.02-2015.32.

[1]More precisely, tensors are introduced in linear algebra as multilinear forms which are naturally represented, for a given basis of the considered Euclidean space, by multi-way array.

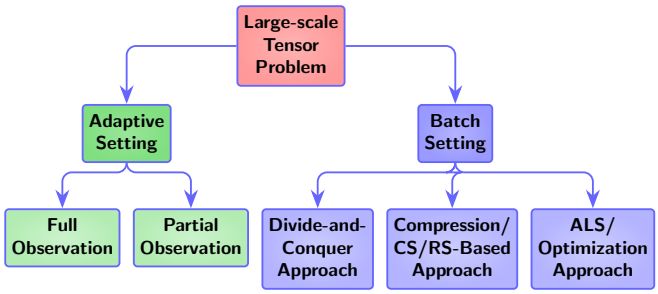


Fig. 1. Taxonomy of large-scale tensor problem.

Two widely-used tensor decompositions are: (i) parallel factor analysis (PARAFAC) [1], also known as canonical decompostion (CANDECOMP) [2] used for latent parameter estimation, and (ii) Tucker decomposition [3] often used for compression and subspace estimation. While matrix decompositions, such as singular value decomposition (SVD) and non-negative matrix decomposition, are used as powerful tools to analyze two dimensional data, tensor decompositions are more versatile because they enjoy the following main advantages for multi-dimensional data:

1) Tensors are a natural generalization of matrices;
2) The PARAFAC decomposition possesses the uniqueness property under mild conditions [4]. Note that additional constraints, such as non-negativity and sparseness, imposed on the tensor model, when possible, can improve the uniqueness property and/or interpretation [5];
3) The Tucker decomposition takes into account the multi-way structure of data and captures multiple interactions instead of pairwise interactions, which will be destroyed if applying matrix decomposition to collapse some of the modes of data [6];
4) Tensor decompositions outperform matrix decompositions in some practical applications as shown in [7].

Tensor decomposition is encountered in diverse applications, such as: psychometrics, chemistry, signal processing, linear and multilinear algebra, data communication, data mining, computer vision, machine learning, to name a few. Thus, a comprehensive survey that covers all those disciplines is difficult. We list here several important surveys arranged in chronological order: basics of tensor decomposition and its

978-1-5090-2710-1/16/$31.00 ©2016 IEEE

applications [4], unsupervised multiway data analysis [8], multi-linear subspace learning for tensor data [9], tensor factorization and decomposition in data mining [10], low-rank tensor approximation [11] and tensor decomposition for signal processing [7], [12]. This list is by no means exhaustive and for more details, we refer to them and the references therein.

Our short survey, complementary to the mentioned surveys, focuses on the problem of large-scale tensors and the potential approaches to solve it. In this paper, moreover, we also introduce our own contributions on this topic. Due to limited space, we only consider the two most popular tensor decomposition models– PARAFAC and Tucker–, and disregard other important models such as block tensor decomposition [13]–[15], tensor networks including tensor trains [16] and hierarchical Tucker [17], coupled matrix/tensor-tensor decomposition [18], [19].

Notations: We follow the notations used in [4]. Calligraphic letters are used for tensors ($\mathcal{A}, \mathcal{B}, \dots$). Matrices, vectors, and scalars are denoted by boldface uppercase, boldface lowercase, and lowercase respectively; for example $\mathbf{A}, \mathbf{a}$, and a. Element (i, j, k) of a tensor $\mathcal{A} \in \mathbb{R}^{I\times J\times K}$ is symbolized as a_{ijk}, element (i, j) of a matrix $\mathbf{A} \in \mathbb{R}^{I\times J}$ as a_{ij}, and i-th entry of a vector $\mathbf{a} \in \mathbb{R}^{I}$ as a_i. Moreover, $\mathbf{A} \otimes \mathbf{B}$ defines the Kronecker product of $\mathbf{A}$ and $\mathbf{B}$, $\mathbf{A} \odot \mathbf{B}$ the Khatri-Rao (column-wise Kronecker) product and $\mathbf{A} * \mathbf{B}$ the Hadamard product which is the element-wise matrix product, $\mathbf{a} \circ \mathbf{b}$ the outer product of $\mathbf{a}$ and $\mathbf{b}$.

II. From Matrix Decomposition to Tensor Decomposition

Before starting with basic operators and models of tensor decomposition, we provide "the bridge" between matrix and tensor decompositions through a comparison of their uniqueness. In its general form, matrix decomposition can be written as

$$\mathbf{X} = \mathbf{P}\mathbf{Q}^T, \tag{1}$$

which is non-unique. It means that there always exists a non-singular matrix $\mathbf{S}$ such that

$$\mathbf{X} = \mathbf{P}\mathbf{Q}^T = (\mathbf{P}\mathbf{S}^{-1})(\mathbf{S}\mathbf{Q}^T) = \hat{\mathbf{P}}\hat{\mathbf{Q}}^T \tag{2}$$

where $\hat{\mathbf{P}} = \mathbf{P}\mathbf{S}^{-1}$ and $\hat{\mathbf{Q}} = \mathbf{Q}\mathbf{S}^T$. To be unique, additional constraints must be imposed. Among various constraints, the most popular ones are orthogonality, sparseness and non-negativity. For example, SVD of $\mathbf{X}$ is given by

$$\mathbf{P} = \mathbf{U}\mathbf{E} \text{ and } \mathbf{Q} = \mathbf{V}, \tag{3}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal and $\mathbf{E}$ is a diagonal matrix with non-negative real singular values. By convention, singular values are arranged in descending order. As a result, the uniqueness of SVD, up to a sign, is due to the orthogonal constraint on $\mathbf{U}$ and $\mathbf{V}$, and the ordered diagonal matrix $\mathbf{E}$.

Let us consider, for example, the PARAFAC tensor decomposition. The PARAFAC model in matrix form (unfolded tensor) can be represented as (see next section for more details)

$$\mathbf{X} = \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T. \tag{4}$$

If we decompose $\mathbf{X}$ by SVD, we will obtain

$$\mathbf{P} = \mathbf{A}\hat{\mathbf{S}}^{-1}, \tag{5}$$

$$\mathbf{Q} = (\mathbf{C} \odot \mathbf{B})\hat{\mathbf{S}}^T, \tag{6}$$

where $\hat{\mathbf{S}}$ is a non-singular matrix. However, different from the matrix case, we know that $\mathbf{Q}$ has a Khatri-Rao product structure. Without any additional constraints, by "restoring" the Khatri-Rao structure of $\mathbf{Q}$, we can recover matrix $\mathbf{B}$ and $\mathbf{C}$, then $\mathbf{A}$ uniquely (up to scale and permutation). Thus, the PARAFAC model can be seen as matrix decomposition of the unfolded tensor with the Khatri-Rao product structure being imposed on matrix $\mathbf{Q}$. The point here is that, using tensor decomposition, if possible, often provides rich structures which can be efficiently exploited to improve the performance and convergence rate of certain algorithms.

III. Basic Tensor Operations and ¡odels

In this section, we present basic tensor operators which are often used in developing algorithms for tensor decomposition. We also present intuitive ideas behind PARAFAC and Tucker models and their uniqueness properties. This section is a summary of rich literature which can be found in the above-listed surveys. For simplicity, we will present most results in the case of 3-way tensors.

A. Basic tensor operations

1) Tensor unfolding: Tensor unfolding, also known as matricization and flattening, is an operation which reorders a tensor into a matrix. Using tensor unfolding allows exploitation of well-defined properties developed in linear algebra for vectors and matrices and provides a convenient way to process tensors. We note that tensor operators can be implemented without tensor unfolding.

Mode-n unfoldings of a tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ are defined as

$$\mathbf{X}_{(1)} : [\mathbf{X}_{(1)}]_{i,j+(k-1)J} = x_{ijk},$$
$$\mathbf{X}_{(2)} : [\mathbf{X}_{(2)}]_{j,i+(k-1)I} = x_{ijk},$$
$$\mathbf{X}_{(3)} : [\mathbf{X}_{(3)}]_{k,i+(j-1)I} = x_{ijk}.$$

There are different ways to choose ordering of columns. In the literature, the following three ways are considered: forward cyclic [20], backward cyclic [21] and ascending order [4]. The mode-n-unfolding here corresponds to the ascending case. Using different tensor unfoldings leads to slightly different formula of tensor models. However, it does not affect the final results (i.e., recovered factors) as long as it is chosen consistently.

2) Tensor multiplication: Entries of mode-1 product of a tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ and a matrix $\mathbf{A} \in \mathbb{R}^{N\times I}$, denoted by $(\mathcal{X} \times_n \mathbf{A})$, is given by

$$(\mathcal{X} \times_1 \mathbf{A})_{njk} = \sum_{i=1}^{I} x_{ijk}a_{ni}.$$

More generally, entries of *mode-n product* of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \cdots \times I_n \times \cdots \times I_N}$ and a matrix $\mathbf{A} \in \mathbb{R}^{J \times I_n}$, denoted by $(\mathcal{X} \times_n \mathbf{A})$, is defined as

$$(\mathcal{X} \times_n \mathbf{A})_{i_1 \cdots i_{n-1} j i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 \cdots i_N} a_{j i_n}.$$

Let $\mathcal{Y} = \mathcal{X} \times_n \mathbf{A}$. We can write an equivalent expression in unfolded tensor form as follows:

$$\mathbf{Y}_{(n)} = \mathbf{A}\mathbf{X}_{(n)}.$$

We also have the following properties:

$$\mathcal{X} \times_n \mathbf{A} \times_m \mathbf{B} = \mathcal{X} \times_m \mathbf{B} \times_n \mathbf{A}, \tag{7}$$
$$\mathcal{X} \times_n \mathbf{A} \times_n \mathbf{B} = \mathcal{X} \times_n (\mathbf{B}\mathbf{A}). \tag{8}$$

The **inner product** of two same-size tensors $\mathcal{X}$, $\mathcal{Y} \in \mathbb{R}^{I \times J \times K}$ is defined as

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} x_{ijk} y_{ijk}.$$

As a consequence, we have $\langle \mathcal{X}, \mathcal{X} \rangle = \|\mathcal{X}\|^2$.

3) Useful matrix properties: Several useful matrix properties are summarized here

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{A}\mathbf{C}) \otimes (\mathbf{B}\mathbf{D})$$
$$(\mathbf{A} \odot \mathbf{B})^T (\mathbf{A} \odot \mathbf{B}) = (\mathbf{A}^T\mathbf{A}) * (\mathbf{B}^T\mathbf{B})$$
$$\mathrm{vec}(\mathbf{A}\mathbf{B}\mathbf{C}^T) = (\mathbf{C} \otimes \mathbf{A})\,\mathrm{vec}(\mathbf{B}),$$
$$(\mathbf{A} \otimes \mathbf{B})^{\#} = ((\mathbf{A}^T\mathbf{A}) * (\mathbf{B}^T\mathbf{B}))^{\#} (\mathbf{A} \otimes \mathbf{B})^T$$
$$(\mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C})(\mathbf{D} \odot \mathbf{E} \odot \mathbf{F}) = (\mathbf{A}\mathbf{D}) \odot (\mathbf{B}\mathbf{E}) \odot (\mathbf{C}\mathbf{F})$$

where $\mathrm{vec}(\cdot)$ performs vectorization of a matrix or a tensor that stacks the columns of the matrix or the tensor into a vector. That is, given $\mathbf{B} \in \mathbb{R}^{I \times J}$, $\mathrm{vec}(\mathbf{B}) = [\mathbf{b}_1^T, \cdots, \mathbf{b}_J^T]^T$. The superscript $(\cdot)^{\#}$ is the pseudo-inverse operator. Dimensions of each matrix are assumed to match.

B. PARAFAC model

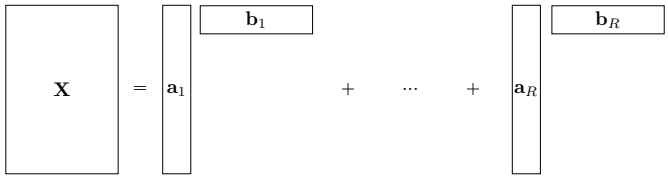
The intuitive idea behind PARAFAC can be captured through Figure 2. While SVD for matrices can be written as sum of R rank one matrices (Figure 2 (a)), the PARAFAC decomposition of $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ can be defined as

$$\mathcal{X} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \equiv \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r, \tag{9}$$

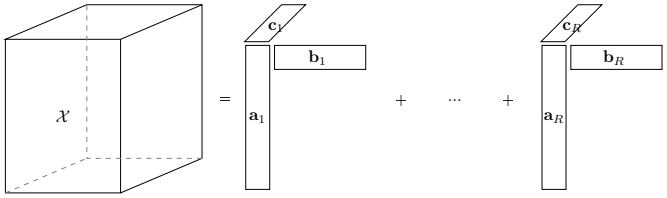
or equivalently

$$\mathbf{x}_{ijk} = \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr}, \tag{10}$$

which is the sum of R rank-one tensors (Figure 2 (b)), with R being the tensor rank. The set of vectors, $\{\mathbf{a}_r\}, \{\mathbf{b}_r\}, \{\mathbf{c}_r\}$ can be grouped into the so-called loading matrices $\mathbf{A} = [\mathbf{a}_1 \dots \mathbf{a}_R] \in \mathbb{R}^{I \times R}$, $\mathbf{B} = [\mathbf{b}_1 \dots \mathbf{b}_R] \in \mathbb{R}^{J \times R}$, and $\mathbf{C} = [\mathbf{c}_1 \dots \mathbf{c}_R] \in \mathbb{R}^{K \times R}$.



(a) SVD of a matrix as sum of R rank-1 matrices.



(b) PARAFAC of a tensor as sum of R rank-1 tensors.

Fig. 2. PARAFAC can be seen as a generalization of SVD.

Equation (9) can also be formulated in matrix and vector form using mode-n unfolding as

$$\mathbf{X}_{(1)} = \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T \tag{11}$$
$$\mathbf{X}_{(2)} = \mathbf{B}(\mathbf{C} \odot \mathbf{A})^T \tag{12}$$
$$\mathbf{X}_{(3)} = \mathbf{C}(\mathbf{B} \odot \mathbf{A})^T \tag{13}$$
$$\mathbf{x} = (\mathbf{C} \odot \mathbf{B} \odot \mathbf{A})\mathbf{1} \tag{14}$$

where $\mathbf{x} = \mathrm{vec}(\mathbf{X}_{(1)})$ and $\mathbf{1} \in \mathbb{R}^{R \times 1}$ whose all entries are one. It is straightforward to see from Figure 2(b) and Equation (9) that the sum is unchangeable if we reorder and re-scale rank-one tensors (hence, order of vectors in loading matrices). Thus, we have

$$\mathcal{X} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] = [\![\mathbf{A}\mathbf{\Pi}\mathbf{\Lambda}_1, \mathbf{B}\mathbf{\Pi}\mathbf{\Lambda}_2, \mathbf{C}\mathbf{\Pi}\mathbf{\Lambda}_3]\!], \tag{15}$$

where $\mathbf{\Pi}$ is a permutation matrix and $\mathbf{\Lambda}_i, i = 1, \cdots, 3$, are scale diagonal matrices satisfying $\mathbf{\Lambda}_1\mathbf{\Lambda}_2\mathbf{\Lambda}_3 = \mathbf{I}$. The PARAFAC decomposition is generically unique (up to scales and permutation) if the following condition is satisfied [22]:

$$2R(R-1) \leqslant I(I-1)K(K-1), \qquad R \leqslant J.$$

This condition is developed based on Kruskal's result [23].

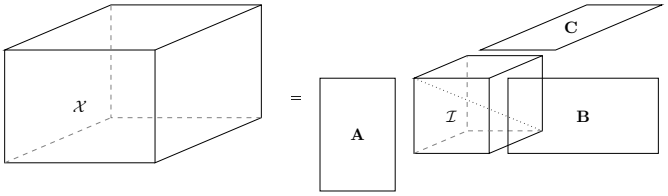
C. Tucker model

We note that PARAFAC can also be illustrated as Figure 3 (a) with an identity tensor[2] $\mathcal{I}$. If we relax this constraint (i.e., $\mathcal{I}$ now can be sparse or dense tensor), we form the Tucker decomposition of $\mathcal{Y} \in \mathbb{R}^{I \times J \times K}$ which can be written as follows:

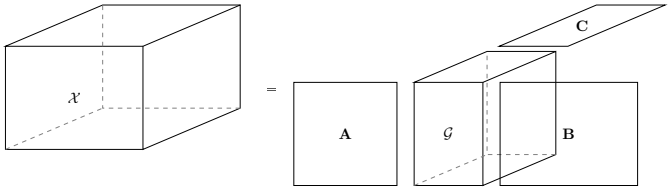
$$\mathcal{Y} = [\![\mathcal{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \equiv \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} \mathbf{a}_p \circ \mathbf{b}_q \circ \mathbf{c}_r, \tag{16}$$

where $\mathbf{A} = [\mathbf{a}_1 \dots \mathbf{a}_P] \in \mathbb{R}^{I \times P}$, $\mathbf{B} = [\mathbf{b}_1 \dots \mathbf{b}_Q] \in \mathbb{R}^{J \times Q}$ and $\mathbf{C} = [\mathbf{c}_1 \dots \mathbf{c}_R] \in \mathbb{R}^{K \times R}$ are the factor matrices and

[2]An identity tensor is a cubical tensor with ones along the superdiagonal.



(a) Representation of PARAFAC using loading matrices and identity core.



(b) Tucker model with a non-identity core tensor $\mathcal{G}$.

Fig. 3. PARAFAC can be seen as a special case of Tucker.

$\mathcal{G} \in \mathbb{R}^{P \times Q \times R}$ is called the core tensor. In general, the factor matrices of the Tucker model are not necessarily orthogonal. However, in practice, a column-wise orthogonal constraint is, in most cases, imposed.

A special case of orthogonal Tucker decomposition is Higher-Order SVD (HOSVD) [21] where the core tensor has the all-orthogonal property besides orthogonal factors. All-orthogonal property means that by considering a 3-way core tensor, the matrices, extracted by fixing one index and releasing two the others, are mutually orthogonal.

Matrix and vector forms of (16) can be presented as

$$\mathbf{Y}_{(1)} = \mathbf{A}\mathbf{G}_{(1)}(\mathbf{C} \otimes \mathbf{B})^T \tag{17}$$

$$\mathbf{Y}_{(2)} = \mathbf{B}\mathbf{G}_{(2)}(\mathbf{C} \otimes \mathbf{A})^T \tag{18}$$

$$\mathbf{Y}_{(3)} = \mathbf{C}\mathbf{G}_{(3)}(\mathbf{B} \otimes \mathbf{A})^T \tag{19}$$

$$\mathbf{y} = (\mathbf{C} \otimes \mathbf{B} \otimes \mathbf{A})\mathbf{g}, \tag{20}$$

where $\mathbf{y} = \mathrm{vec}(\mathcal{Y})$ and $\mathbf{g} = \mathrm{vec}(\mathcal{G})$.

In contrast to the PARAFAC model, the Tucker model is non-unique since

$$\mathcal{Y} = [\![\mathcal{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!] = [\![\mathcal{G} \times_1 \mathbf{P} \times_2 \mathbf{Q} \times_3 \mathbf{R}; \mathbf{A}\mathbf{P}^{-1}, \mathbf{B}\mathbf{Q}^{-1}, \mathbf{C}\mathbf{R}^{-1}]\!]. \tag{21}$$

Uniqueness can be achieved only if specific constraints are added, for example both sparsity and non-negativity constraints.

IV. Batch Setting

In batch setting, we categorize the existing algorithms based on their tensor decomposition approaches. For each one, we first describe the main idea which is shared by all algorithms and then present specific solutions and their differences.

A. Divide-and-Conquer approach

The main idea of this approach (Figure 4) is to divide a big data tensor into a number of smaller data tensors; then run specific tensor decomposition algorithms on those small tensors (possibly in a parallel scheme) before joining "local" results into "global" one. The difference resides in the way each algorithm handle the data (i.e., decentralized or distributed) and on the optimization techniques in use.

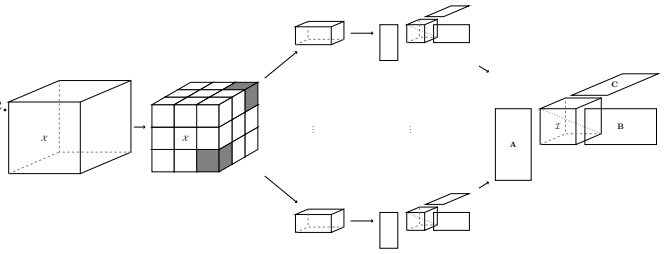


Fig. 4. Divide-and-Conquer approach for large-scale tensors (An example of the PARAFAC model.)

1) PARAFAC model: In [24], the authors proposed to use this approach combined with a fast Alternating Least-Squares (ALS) algorithm. To speed up the joining procedure, they also proposed a multi-stage reconstruction step where local factors are merged from results of neighbouring sub-tensors. The algorithm works under the assumption that the decomposition of each sub-tensor is strictly unique.

In [25], we also used a decentralized approach but the way the structure of PARAFAC is used is different from [24]. In fact, our algorithm is an extension in spirit of the Generalized Minimum Noise Subspace (GMNS) method [26], [27] which permits the application of (stable) SVD at small-scale to subspace estimation. Our algorithm also use the same uniqueness condition assumption as in [24].

To relax the uniqueness condition on sub-tensors, in [28], [29], the authors developed a distributed ALS algorithm. The main idea is to permit collaboration across the three modes of the tensor.

2) Tucker model: A distributed memory Tucker decomposition for data compression was proposed in [30]. While each block data is fixed in each processor, factor matrices are exchanged between processor grid. Parallel implementation of Higher Order Orthogonal Iteration (HOOI) [31] using Sequentially-Truncated HOSVD [32] as an initialization, was taken into account. Those algorithms can also be implemented efficiently by using level-3 Basic Linear Algebra Subprograms (BLAS) routines[3].

B. Compression/compressive sensing/random sampling based approaches

The spirit of these approaches is to process a reduced-size or a sparse representation that essentially keeps the same or approximately the same information as the original form. An illustration of this approach is given in Figure 5.

In tensor decomposition, the *compression* approach-based algorithms always use the Tucker decomposition or HOSVD to compress data. Then, depending on applications, a desired decomposition (e.g., PARAFAC) is applied on the core tensor

[3]Level-3 BLAS aims to implement matrix-matrix operations and supports block-partitioned algorithms

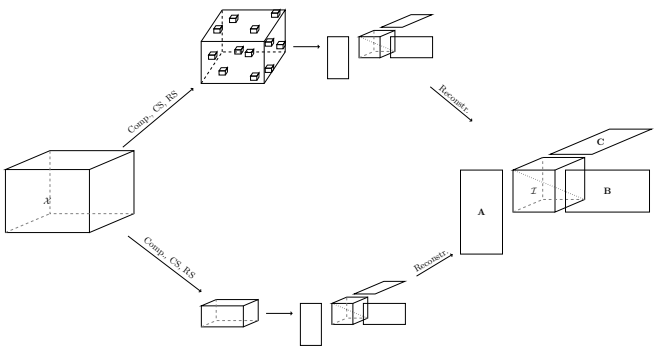


Fig. 5. Compression/compressive sensing/random sampling approaches for large-scale tensors (PARAFAC model).

to extract loading factors. To recover the factors of the original form, the extracted factors are simply multiplied with the corresponding factors of Tucker decomposition. As a consequence, it allows to avoid running desired decomposition on large dimensional tensors (i.e., avoid running iterative algorithm[4] in large-scale data). Since both Tucker and HOSVD algorithms require SVD or EVD computation of large dimensional matrices, this approach would be appropriate only for small or moderate size tensor decomposition. For more details, we refer the reader to [33].

The *random sampling* (RS) approach represents data in smaller size or sparse form while approximately preserving essential information (see [34] for an introduction and review). Depending on data form (e.g., matrix or tensor) and algorithms, there are different kinds of sampling strategies such as element-wise, row/column, slide and block. Moreover, those strategies can be chosen following several specific probability distributions (e.g., uniform, non-uniform, data-dependent).

For the Tucker model, in [35], along each mode, a column sampling strategy, which can be one-pass or multiple-pass, is first applied to the unfolded tensor. Then the factor matrices are computed as principal singular vectors of the sampled matrix. A similar method but using element-wise sampling is developed in [36]. We note that element-wise sampling yields sparse representation instead of reduced-size form.

For the PARAFAC model, the Parcube algorithm [37] uses data-dependent-based sampling to determine the important part of the tensor (i.e., marginal sum of the tensor for each mode), runs PARAFAC for each sampled tensor and then merges results. This algorithm only works for sparse tensors and offers no identifiablity guarantee.

In [38], authors further developed the compression approach by assuming that the big tensor has an underlying low-rank structure (i.e., the PARAFAC model) and the factor matrices are sparse. While the low-rank structure allows the design of a special compression matrix which has a Kronecker product structure, the sparse factors help to guarantee identifiability (i.e., uniqueness of the recovered factors from results of the compressed tensor). Here, the Kronecker product structure of the compression matrix keeps the compressed tensor having a low-rank as the original one. Thus, authors claimed that this approach can be considered as a generalization of *compressive sensing* (CS) idea for the multilinear case. A combination of this approach and divide-and-conquer strategy can be found in [39].

[4]Most PARAFAC decomposition algorithms are iterative.

C. Alternating least-square/Optimization approach

For tensor decomposition, *alternating least-squares* (ALS)is considered the workhorse algorithm for a long time. The idea of the alternating approach is simple; at each step, we optimize a factor while keeping the others fixed. At the beginning, there were several efforts to accelerate convergence and overcome degeneracy[5] using a line-search approach [40]. The direct implementation of ALS is difficult to handle for large-scale data because (i) the size of intermediate computation results between the unfolded tensor and all-but-one factors is much larger than that of the interested loading matrices (ii) multiple accesses of original tensor data in different orders are necessary. Several techniques to tackle those problems have been addressed in [41] and [42].

The *general optimization* approach casts the tensor decomposition problem into a nonlinear equation problem and then solves it using standard optimization tools, such as gradient-based methods. In some difficult situations such as degeneracy or over-factoring case[6], gradient-based algorithms can outperform ALS in terms of accuracy and performance. Within this class, various algorithms have been developed, including the non-linear conjugate gradient [43], also see [44] for the case of missing data, [45] for the case of sparse and nonnegative PARAFAC and Tucker, the fast damped Gauss-Newton (dGN) algorithms [46] and the Alternating Direction Method of Multipliers (ADMM) [47].

V. Adaptive Setting

In adaptive setting. also known as *online* or *incremental* setting, we classify the existing algorithms with respect to the characteristic of streaming data: full observation (i.e., without missing data) and partial observation (i.e., with missing data). An illustration of the adaptive tensor setting is given in Figure 6.

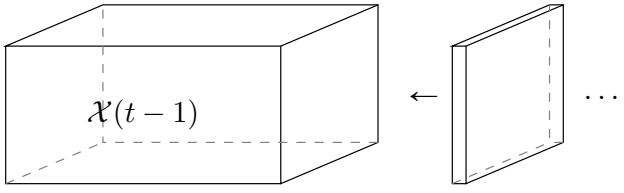


Fig. 6. Adaptive tensor setting: at time instant t, tensor $\mathcal{X}(t)$ captures a new data slice.

[5]Degeneracy refers to a problem when collinearity of two or more components in the factor matrices exists.

[6]Over-factoring means that the chosen tensor rank is larger than the true one.

A. Full observation

An adaptive PARAFAC model for third-order tensors having one dimension growing with time has been introduced in [48]. Two algorithms using recursive least-squares tracking (PARAFAC-RLST) and simultaneous diagonalization tracking (PARAFAC-SDT) have been proposed. While the former uses first-order methods (i.e., using gradients) to optimize an exponentially-weighted least-squares cost function, the latter exploits an SVD tracking algorithm combined with a recursive simultaneous diagonalization step. The computational complexity of both algorithms is quadratic in the tensor rank.

To deal with computational complexity of [48], we have proposed a linear complexity adaptive PARAFAC algorithm [49] which generalizes the Orthonormal Projection Approximation Subspace Tracking (OPAST) approach [50]. This algorithm, named 3DOPAST, uses a special interpretation of the Khatri-Rao product as collinear vectors inside each column. Its performance is equal or even superior to PARAFAC-RLST and PARAFAC-SDT while keeping the computational complexity linear with respect to the tensor rank. An improved version of 3DOPAST, named Second-Order Optimization based Adaptive PARAFAC Decomposition (SOAP), using a second-order stochastic gradient as well as preserving the Khatri-Rao product structure is also proposed in [51]. Moreover, we also adapt SOAP to handle the adaptive non-negative PARAFAC model. It is shown that SOAP is stable for very long time run. For more details, we invite the reader to our papers [49], [51].

Adaptive Tucker decomposition has several different names in the literature, for examples, dynamic tensor analysis [52], incremental tensor subspace learning [53], tensor subspace tracking [54]. Streaming tensor analysis was proposed in [52]. In this work, the eigensubspace is updated via tracking a projection matrix for all modes. Dynamic tensor analysis, which is a more general model for streaming tensor analysis, allows a changeable amount of data to come at each time instant. The main idea of the proposed algorithm for this kind of models is to track the covariance matrix and the eigensubspace of unfolded tensors for each mode. With the same spirit, the incremental tensor subspace learning applies an incremental SVD algorithm [55] to three unfolded tensors. In tensor subspace tracking, based on Kronecker-structured projection, the tensor subspace is tracked by using the matrix-based subspace tracking PAST algorithm [56].

B. Partial observation

In another recent work [57], authors have proposed an adaptive PARAFAC for incomplete streaming data. They proposed to use a first-order method to minimize an exponentially-weighted least-squares cost function with regularization terms. Second-order methods have been considered independently in [58] as well as in our work [59].

VI. Conclusion

This paper provides a brief overview of decomposition methods for large-dimensional tensors. Three classes of approaches are highlighted that handle the large-scale problem by reducing the data size (i.e., compression, compressive sensing, random sampling), by using parallel computing (i.e., divide-and-conquer) or by alternating optimization (e.g., by reducing the size of the parameters with respect to which the optimization is achieved). In the case of streaming data, our fast adaptive algorithms were presented including the 3DOPAST and SOAP methods that have the advantage of linear complexity.

References

[1] R. A. Harshman, "Foundations of the PARAFAC procedure: Models and conditions for an" explanatory" multi-modal factor analysis," *UCLA Working Papers in Phonetics*, , no. 1, pp. 84, 1970.

[2] J. D. Carroll and J.-J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition," *Psychometrika*, pp. 283–319, 1970.

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

[4] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, pp. 455–500, 2009.

[5] A. Cichocki, R. Zdunek, A. H. Phan, and S. Amari, *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*, John Wiley & Sons, 2009.

[6] A Cichocki, "Tensors decompositions: New concepts for brain data analysis?," *Journal of Control Measurement, and System Integration*, pp. 507–517, 2011.

[7] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *Signal Processing Magazine, IEEE*, pp. 145–163, 2015.

[8] E. Acar and B. Yener, "Unsupervised multiway data analysis: A literature survey," *IEEE transactions on knowledge and data engineering*, pp. 6–20, 2009.

[9] H. Lu, K. N Plataniotis, and A. N. Venetsanopoulos, "A survey of multilinear subspace learning for tensor data," *Pattern Recognition*, pp. 1540–1551, 2011.

[10] M. Mørup, "Applications of tensor (multiway array) factorizations and decompositions in data mining," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, pp. 24–40, 2011.

[11] L. Grasedyck, D. Kressner, and C. Tobler, "A literature survey of low-rank tensor approximation techniques," *GAMM-Mitteilungen*, pp. 53–78, 2013.

[12] P. Comon, "Tensors: a brief introduction," *IEEE Signal Processing Magazine*, pp. 44–53, 2014.

[13] L. De Lathauwer, "Decompositions of a higher-order tensor in block terms-part i: Lemmas for partitioned matrices," *SIAM Journal on Matrix Analysis and Applications*, pp. 1022–1032, 2008.

[14] L. De Lathauwer, "Decompositions of a higher-order tensor in block terms-part ii: Definitions and uniqueness," *SIAM Journal on Matrix Analysis and Applications*, pp. 1033–1066, 2008.

[15] L. De Lathauwer and D. Nion, "Decompositions of a higher-order tensor in block terms-part iii: Alternating least squares algorithms," *SIAM journal on Matrix Analysis and Applications*, pp. 1067–1083, 2008.

[16] I. V Oseledets, "Tensor-train decomposition," *SIAM Journal on Scientific Computing*, pp. 2295–2317, 2011.

[17] W. Hackbusch and S. Kühn, "A new scheme for the tensor representation," *Journal of Fourier Analysis and Applications*, pp. 706–722, 2009.

[18] E. Acar, R. Bro, and A. K. Smilde, "Data fusion in metabolomics using coupled matrix and tensor factorizations," *Proceedings of the IEEE*, pp. 1602–1620, Sept 2015.

[19] R. Cabral Farias, J. Cohen, and P. Comon, "Exploring multimodal data fusion through joint decompositions with flexible couplings," *IEEE-T-SP*, pp. 1–1, 2016.

[20] H. AL Kiers, "Towards a standardized notation and terminology in multiway analysis," *Journal of chemometrics*, pp. 105–122, 2000.

[21] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM journal on Matrix Analysis and Applications*, pp. 1253–1278, 2000.

[22] L. De Lathauwer, "A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization," *SIAM journal on Matrix Analysis and Applications*, pp. 642–666, 2006.

[23] J. B Kruskal, "Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics," *Linear algebra and its applications*, pp. 95–138, 1977.

[24] A.-H Phan and A. Cichocki, "PARAFAC algorithms for large-scale problems," *Neurocomputing*, pp. 1970–1984, 2011.

[25] V.-D. Nguyen, K. Abed-Meraim, and N. Linh-Trung, "Parallelizable PARAFAC decomposition of 3-way tensors," in *Proc. ICASSP*. IEEE, 2015, pp. 5505–5509.

[26] Viet-Dung Nguyen, Karim Abed-Meraim, Nguyen Linh-Trung, and Rodolphe Weber, "Generalized MNS method for parallel minor and principal subspace analysis," in *Proc. EUSIPCO*. IEEE, 2014, pp. 2265–2269.

[27] V.-D. Nguyen, K. Abed-Meraim, N. Linh-Trung, and R. Weber, "Array processing using generalized minimum noise subspace," *HAL-https://hal.inria.fr/hal-01295030*, 2016.

[28] A. LF de Almeida and A. Y Kibangou, "Distributed computation of tensor decompositions in collaborative networks," in *Proc. CAMSAP*. IEEE, 2013, pp. 232–235.

[29] A.L.F. De Almeida and A. Y. Kibangou, "Distributed large-scale tensor decomposition," in *Proc. ICASSP*. IEEE, 2014, pp. 26–30.

[30] W. Austin, G. Ballard, and T. G. Kolda, "Parallel tensor compression for large-scale scientific data," in *Proc. IPDPS*, May 2016.

[31] L. De Lathauwer, B. De Moor, and J. Vandewalle, "On the best rank-1 and rank-(r 1, r 2,..., rn) approximation of higher-order tensors," *SIAM Journal on Matrix Analysis and Applications*, pp. 1324–1342, 2000.

[32] N. Vannieuwenhoven, R. Vandebril, and K. Meerbergen, "A new truncation strategy for the higher-order singular value decomposition," *SIAM Journal on Scientific Computing*, pp. A1027–A1052, 2012.

[33] R. Bro and C. A Andersson, "Improving the speed of multiway algorithms: Part ii: Compression," *Chemometrics and intelligent laboratory systems*, pp. 105–113, 1998.

[34] P. Drineas and M. W. Mahoney, "Randnla: Randomized numerical linear algebra," *Communications of the ACM*, pp. Pages 80–90, 2016.

[35] P. Drineas and M. W Mahoney, "A randomized algorithm for a tensor-based generalization of the singular value decomposition," *Linear algebra and its applications*, pp. 553–571, 2007.

[36] C. E Tsourakakis, "MACH: Fast randomized tensor decompositions.," SIAM.

[37] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos, "Parcube: Sparse parallelizable tensor decompositions," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2012, pp. 521–536.

[38] N. D. Sidiropoulos and A. Kyrillidis, "Multi-way compressed sensing for sparse low-rank tensors," *IEEE Signal Processing Letters*, pp. 757–760, 2012.

[39] N. Sidiropoulos, E. Papalexakis, and C. Faloutsos, "Parallel randomly compressed cubes: A scalable distributed architecture for big tensor decomposition," *Signal Processing Magazine, IEEE*, pp. 57–70, 2014.

[40] M. Rajih, P. Comon, and R. A Harshman, "Enhanced line search: A novel method to accelerate parafac," *SIAM journal on matrix analysis and applications*, pp. 1128–1147, 2008.

[41] A.-H. Phan, P. Tichavskỳ, and A. Cichocki, "Fast alternating ls algorithms for high order candecomp/parafac tensor factorizations," *IEEE-T-SP*, pp. 4834–4846, 2013.

[42] N. Ravindran, N. D Sidiropoulos, S. Smith, and G. Karypis, "Memory-efficient parallel computation of tensor and matrix products for big tensor decomposition," in *Proc. Asilomar*. IEEE, 2014, pp. 581–585.

[43] Acar E., Dunlavy D. M., and Kolda T. G., "A scalable optimization approach for fitting canonical tensor decompositions," *Journal of Chemometrics*, pp. 67–86, February 2011.

[44] E. Acar, D. M Dunlavy, T. G Kolda, and M. Mørup, "Scalable tensor factorizations for incomplete data," *Chemometrics and Intelligent Laboratory Systems*, pp. 41–56, 2011.

[45] V.-D. Nguyen, K. Abed-Meraim, and N. Linh-Trung, "New robust algorithms for sparse non-negative three-way tensor decompositions," in *Proc. EUSIPCO*, 2016.

[46] A.-H Phan, P. Tichavsky, and A. Cichocki, "Low complexity damped gauss–newton algorithms for candecomp/parafac," *SIAM Journal on Matrix Analysis and Applications*, pp. 126–147, 2013.

[47] A. P Liavas and N. D Sidiropoulos, "Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers," *IEEE-T-SP*, pp. 5450–5463, 2015.

[48] D. Nion and N. D. Sidiropoulos, "Adaptive algorithms to track the PARAFAC decomposition of a third-order tensor," *IEEE-T-SP*, pp. 2299–2310, 2009.

[49] V.-D Nguyen, K. Abed-Meraim, and N. Linh-Trung, "Fast adaptive PARAFAC decomposition algorithm with linear complexity," in *Proc. ICASSP*, 2016.

[50] K Abed-Meraim, A Chkeif, and Y Hua, "Fast orthonormal PAST algorithm," *Signal Processing Letters, IEEE*, pp. 60–62, 2000.

[51] V.-D. Nguyen, K. Abed-Meraim, and N. Linh-Trung, "On adaptive PARAFAC decomposition of three-way tensors," *HAL-https://hal.inria.fr/hal-01295020*, 2016.

[52] J. Sun, D. Tao, S. Papadimitriou, P. S. Yu, and C. Faloutsos, "Incremental tensor analysis: Theory and applications," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, p. 11, 2008.

[53] W. Hu, X. Li, X. Zhang, X. Shi, S. Maybank, and Z. Zhang, "Incremental tensor subspace learning and its applications to foreground segmentation and tracking," *International Journal of Computer Vision*, pp. 303–327, 2011.

[54] Y. Cheng, F. Roemer, O. Khatib, and M. Haardt, "Tensor subspace tracking via Kronecker structured projections (TeTraKron) for time-varying multidimensional harmonic retrieval," *EURASIP Journal on Advances in Signal Processing*, pp. 1–14, 2014.

[55] D. A Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *International Journal of Computer Vision*, pp. 125–141, 2008.

[56] B. Yang, "Projection approximation subspace tracking," *IEEE-T-SP*, pp. 95–107, 1995.

[57] M. Mardani, G. Mateos, and G. B Giannakis, "Subspace learning and imputation for streaming big data matrices and tensors," *IEEE-T-SP*, pp. 2663–2677, 2015.

[58] H. Kasai, "Online low-rank tensor subspace tracking from incomplete data by CP decomposition using recursive least squares," in *Proc. ICASSP*, March 2016, pp. 2519–2523.

[59] T. Minh-Chinh, V.-D. Nguyen, N. Linh-Trung, and K. Abed-Meraim, "Adaptive parafac decomposition for third-order tensor completion," in *Proc. ICCE*, 2016.