A MULTI-RESOLUTION LOW-RANK TENSOR DECOMPOSITION

Sergio Rozada and Antonio G. Marques

Dept. of Signal Theory and Communications, King Juan Carlos University, Madrid, Spain

ABSTRACT

The (efficient and parsimonious) decomposition of higher-order tensors is a fundamental problem with numerous applications in a variety of fields. Several methods have been proposed in the literature to that end, with the Tucker and PARAFAC decompositions being the most prominent ones. Inspired by the latter, in this work we propose a multi-resolution low-rank tensor decomposition to describe (approximate) a tensor in a hierarchical fashion. The central idea of the decomposition is to recast the tensor into *multiple* lower-dimensional tensors to exploit the structure at different levels of resolution. The method is first explained, an alternating least squares algorithm is discussed, and preliminary simulations illustrating the potential practical relevance are provided.

Index Terms— Tensor decomposition, Low-rank approximation, Kronecker decomposition, multi-resolution approximation.

1. INTRODUCTION

We live in a digital age where common-life devices, from smartphones to cars, generate massive amounts of data that provide researchers and practitioners a range of opportunities. Processing contemporary information comes, however, at a cost, since data sources are messy and heterogeneous. In this context, parsimonious models emerge as an ideal tool to enhance efficiency when processing such vast amounts of information. This can be done by leveraging the structure of the data, as is the case of information living in multiple (possibly many) dimensions. Multi-dimensional data are prevalent in numerous fields, with representative examples including chemometrics, bioengineering, communications, hyper-spectral imaging, or psychometrics [1, 2]. Traditionally, matrices were used to model those datasets, but tensor-representation models have been recently breaking through. Multi-dimensional arrays, or tensors, are data structures that generalize the concept of vectors and matrices to highly-dimensional domains. In recent years, tensors have also been applied to address numerous data science and machine learning tasks, from simple interpolation to supervised classification [3].

In this data-science context, a problem of particular interest is that of tensor decomposition, which tries to estimate a set of latent factors that summarize the tensor. Many tensor decompositions were developed as the generalization of well-known matrix-decomposition methods to high-dimensional domains [4, 5]. This was the case of the PARAFAC tensor decomposition [6] and its generalization, the Tucker tensor decomposition [7], which can be both understood as higher-order generalizations of the SVD decomposition of a matrix. More specifically, these decompositions aim at describing (approximating) the tensor as a sum of rank-1 tensors, decomposing it as a sum of outer products of vectors (called factors). The PARAFAC decomposition is conceptually simple and its representation complexity scales gracefully (the number of parameters grows linearly with the rank). The Tucker decomposition enjoys additional degrees of freedom at the cost of greater complexity (exponential dependence of the number of parameters with respect to the rank). Hierarchical tensor decompositions, such as the Tensor Train (TT) decomposition [8] or a hierarchical Tucker (hTucker) decomposition [9], try to alleviate this problem. The former unwraps the tensor into a chain of three-dimensional tensors, and the latter generalizes the same idea by organizing the dimensions in a binary tree. Furthermore, in recent years significant effort has been devoted to modify existing decomposition algorithms to deal with factor constraints (e.g., non-negativeness), promote certain priors (e.g., factor sparsity), or be robust to imperfections [10] [11] [12].

However, little to no work has been carried out to study the tensor decomposition from a multi-resolution perspective. This can be specially interesting for tensor signals such as videos, where 2-, 3-, and 4-dimensional components are mixed in a single tensor. In this work, we postulate a simple but novel multi-resolution low-rank decomposition method. More specifically, this paper:

- Introduces a new multi-resolution tensor decomposition to exploit the low-rank structure of a tensor at different resolutions.
- Proposes an algorithm to implement the decomposition.
- Tests the benefits of the model via numerical simulations.

Regarding the first contribution, rather than postulating a low-rank decomposition of the tensor using the original multidimensional representation, we 1) consider a *collection* of lower-order multidimensional representations of the tensor (where several of the original modes of the tensor are combined into a single one); 2) postulate a low-rank decomposition for each of the lower-dimensional representations; 3) map each of the representations back to the original tensor domain; and 4) model the original tensor as the sum of such low-rank representations. As illustrated in detail in the manuscript, this results in an efficient decomposition method capable of combining low-rank structures present at different resolutions.

Section 2 introduces notation and tensor preliminaries. Section 3 presents our decomposition method. A simple algorithmic approach to address the decomposition is described in Section 4. Illustrative numerical experiments are provided in Section 5.

2. NOTATION AND TENSOR PRELIMINARIES

The entries of a (column) vector $\mathbf{x}$, a matrix $\mathbf{X}$ and a tensor $\underline{\mathbf{X}}$ are denoted by $[\mathbf{x}]_n$, $[\mathbf{X}]_{n_1,n_2}$ and $[\underline{\mathbf{X}}]_{n_1,n_2,\ldots,n_I}$, respectively, with I denoting the order of tensor $\underline{\mathbf{X}}$. Moreover, the nth column of matrix $\mathbf{X}$ is denoted by $[\mathbf{X}]_n$. Sets are represented by calligraphic capital letters. The cardinality of a set $\mathcal{S}$ is denoted by $|\mathcal{S}|$. When a set $\mathcal{S}$ is *ordered*, we use the notation $\mathcal{S}(i)$ with $1 \leq i \leq |\mathcal{S}|$ to denote the ith element of the set. The vertical concatenation of the columns of matrix $\mathbf{X}$ is denoted by $\text{vec}(\mathbf{X})$. $\|\mathbf{X}\|_F$ is the Frobenious norm of matrix $\mathbf{X}$, which can be equivalently written as $\|\text{vec}(\mathbf{X})\|_2$.

Work supported by the Spanish NSF Grant SPGraph (PID2019-105032GB-I00), and by the Grants F661-MAPPING-UCI, F663-AAGNCS and F730-DSSP funded by the Comunidad de Madrid (CAM) and King Juan Carlos University (URJC).

2.1. Tensor to matrix unfolding

Given a tensor $\underline{\mathbf{X}}$ of order I and size $N_1 \times ... \times N_I$, there are many ways to unfold the entries of the tensor into a matrix $\mathbf{X}$. In this section, we are interested in unfoldings where the columns of matrix $\mathbf{X}$ represent one of the original modes of $\underline{\mathbf{X}}$ and the rows of $\mathbf{X}$ represent all the other modes of the tensor. Mathematically, we define the matrix unfolding operator as

$$\begin{aligned} \mathbf{X} =& \mathrm{mat}_p(\underline{\mathbf{X}}) \in \mathbb{R}^{(N_1 \dots N_{p-1} N_{p+1} \dots N_I) \times N_p} \text{ where} \\ & [\mathbf{X}]_{k,n_p} = [\underline{\mathbf{X}}]_{n_1,\dots,n_I} \text{ and} \\ & k = n_1 + \sum_{i=2, i\neq p}^{I} (n_i - 1) \prod_{j=2, j\neq p}^{i-1} N_j. \end{aligned} \tag{1}$$

where $p \leq I$ and, to simplify exposition, we have assumed that $p > 1$.

2.2. Tensor to lower-order tensor unfolding

Consider a tensor $\underline{\mathbf{X}}$, of order I, and let $\mathcal{I} := \{1, 2, ..., I\}$ denote the set containing the indexes of all the modes of $\underline{\mathbf{X}}$.

Definition 1 *The ordered set $\mathcal{P} = \{\mathcal{P}_1, ..., \mathcal{P}_P\}$ is a partition of the set $\mathcal{I}$ if it holds that: $\mathcal{P}_p \neq \emptyset$ for all p, $\mathcal{P}_p \cap \mathcal{P}_{p'} = \emptyset$ for all $p' \neq p$, and $\bigcup_{p=1}^{P} \mathcal{P}_p = \mathcal{I}$.*

We are interested in reshaping the entries of the Ith order tensor $\underline{\mathbf{X}}$ of size $N_1 \times ... \times N_I$ to generate a lower-order tensor $\check{\underline{\mathbf{X}}}$, with order $P < I$ and according to a given partition $\mathcal{P} = \{\mathcal{P}_1, ..., \mathcal{P}_P\}$ as specified next

$$\begin{aligned} \check{\underline{\mathbf{X}}} &= \mathrm{ten}_{\mathcal{P}}(\underline{\mathbf{X}}) \in \mathbb{R}^{\prod_{j=1}^{|\mathcal{P}_1|} |\mathcal{P}_1(j)| \times \dots \times \prod_{j=1}^{|\mathcal{P}_P|} |\mathcal{P}_P(j)|} \\ & [\check{\underline{\mathbf{X}}}]_{k_1,\dots,k_{|\mathcal{P}|}} = [\underline{\mathbf{X}}]_{n_1,\dots,n_I} \text{ and} \\ & k_p = n_{\mathcal{P}_p(1)} \text{ if } |\mathcal{P}_p| = 1 \\ & k_p = n_{\mathcal{P}_p(1)} + \sum_{i=2}^{|\mathcal{P}_p|} (n_{\mathcal{P}_p(i)} - 1) \prod_{j=1}^{i-1} N_{\mathcal{P}_p(j)} \text{ if } |\mathcal{P}_p| > 1 \end{aligned} \tag{2}$$

Note that, according to definition of the $\mathrm{ten}_{\mathcal{P}}(\cdot)$ operator, the indexes along the pth mode of $\check{\underline{\mathbf{X}}}$ represent tuples $(m_{\mathcal{P}_p(1)}, ..., m_{\mathcal{P}_p(|\mathcal{P}_p|)})$ of indexes of the original tensor $\underline{\mathbf{X}}$.

Clearly, if $\mathcal{P} = \{\mathcal{I}\}$, so that $P = |\mathcal{P}| = 1$ and $|\mathcal{P}_1| = I$, we have that $\mathrm{ten}_{\mathcal{P}}(\underline{\mathbf{X}}) = \mathrm{vec}(\underline{\mathbf{X}})$. On the other hand, if $\mathcal{P} = \{\{1\}, \{2\}, ..., \{I\}\}$, so that $P = |\mathcal{P}| = I$ and $|\mathcal{P}_p| = 1$ for all p, we have that $\mathrm{ten}_{\mathcal{P}}(\underline{\mathbf{X}}) = \underline{\mathbf{X}}$.

Finally, the inverse operator of (2), which recovers the original tensor $\underline{\mathbf{X}}$ using as input the reshaped $\check{\underline{\mathbf{X}}} = \mathrm{ten}_{\mathcal{P}}(\underline{\mathbf{X}})$, is denoted by $\mathrm{unten}_{\mathcal{P}}(\check{\underline{\mathbf{X}}}) = \underline{\mathbf{X}}$. Since the definition of $\mathrm{unten}_{\mathcal{P}}(\cdot)$ starting from (2) is straightforward, it is omitted for conciseness.

2.3. Low-rank PARAFAC tensor decomposition

Consider the Ith order tensor $\underline{\mathbf{X}}$ along with the matrices $\mathbf{F}_i \in \mathbb{R}^{N_i \times R}$ for $i = 1, ..., I$. Then, $\underline{\mathbf{X}}$ is said to have rank R if it can be written as

$$\underline{\mathbf{X}} = \sum_{r=1}^{R} [\mathbf{F}_1]_r \circledcirc [\mathbf{F}_2]_r \circledcirc ... \circledcirc [\mathbf{F}_I]_r \tag{3}$$

where $\circledcirc$ is the generalization of the outer product for more than two vectors. That is, if $\mathbf{x} \in \mathbb{R}^{N_1}$, $\mathbf{y} \in \mathbb{R}^{N_2}$, $\mathbf{z} \in \mathbb{R}^{N_3}$ are three generic vectors, then $\mathbf{x} \circledcirc \mathbf{y} \circledcirc \mathbf{z}$ is a tensor of order $I = 3$ satisfying $[\mathbf{x} \circledcirc \mathbf{y} \circledcirc \mathbf{z}]_{n_1,n_2,n_3} = [\mathbf{x}]_{n_1} [\mathbf{y}]_{n_2} [\mathbf{z}]_{n_3} \in \mathbb{R}$.

The decomposition in (3) is oftentimes referred to as canonical polyadic decomposition or PARAFAC decomposition, with matrices $\mathbf{F}_i$ being referred to as factors. As in the case of matrices, moderate values of R induce a parsimonious description of the tensor, since the $\prod_{i=1}^{I} N_i$ values in $\underline{\mathbf{X}}$ can be equivalently represented by the $\sum_{i=1}^{I} RN_i$ entries in $\{\mathbf{F}_i\}_{i=1}^{I}$.

Using the Khatri-Rao product, denoted as $\odot$, and the different unfolding operators introduced in the previous sections, we have that

$$\begin{aligned} \mathrm{mat}_p(\underline{\mathbf{X}}) =& \sum_{r=1}^{R} \mathrm{mat}_p([\mathbf{F}_1]_r \circledcirc [\mathbf{F}_2]_r \circledcirc ... \circledcirc [\mathbf{F}_I]_r) \\ =& (\mathbf{F}_I \odot ... \odot \mathbf{F}_{p+1} \odot \mathbf{F}_{p-1} \odot ... \odot \mathbf{F}_1)(\mathbf{F}_p)^T \end{aligned} \tag{4}$$

$$\begin{aligned} \mathrm{ten}_{\mathcal{P}}(\underline{\mathbf{X}}) &= \sum_{r=1}^{R} \mathrm{ten}_{\mathcal{P}}([\mathbf{F}_1]_r \circledcirc [\mathbf{F}_2]_r \circledcirc ... \circledcirc [\mathbf{F}_I]_r) \\ &= \sum_{r=1}^{R} ([\check{\mathbf{F}}_1]_r \circledcirc [\check{\mathbf{F}}_2]_r \circledcirc ... \circledcirc [\check{\mathbf{F}}_P]_r) \\ \text{with} \quad & \check{\mathbf{F}}_p = \mathbf{F}_{\mathcal{P}_p(|\mathcal{P}_p|)} \odot ... \odot \mathbf{F}_{\mathcal{P}_p(2)} \odot \mathbf{F}_{\mathcal{P}_p(1)}. \end{aligned} \tag{5}$$

These expressions will be leveraged in the next section.

3. MULTI-RESOLUTION LOW-RANK DECOMPOSITION

Consider a collection of partitions $\mathcal{P}^{(1)}$,...,$\mathcal{P}^{(L)}$, with $|\mathcal{P}^{(l)}| \leq |\mathcal{P}^{(l')}|$ for $l < l'$. Given the Ith order tensor $\underline{\mathbf{X}}$ and the collection of partitions $\mathcal{P}^{(1)}$,...,$\mathcal{P}^{(L)}$, we propose the following decomposition for the tensor at hand

$$\underline{\mathbf{X}} = \sum_{l=1}^{L} \underline{\mathbf{Z}}_l, \text{ with } \mathrm{rank}\big(\mathrm{ten}_{\mathcal{P}^{(l)}}(\underline{\mathbf{Z}}_l)\big) \leq R_l, \tag{6}$$

which can be equivalently written as

$$\underline{\mathbf{X}} = \sum_{l=1}^{L} \mathrm{unten}_{\mathcal{P}^{(l)}}(\check{\underline{\mathbf{Z}}}_l), \text{ with } \mathrm{rank}\big(\check{\underline{\mathbf{Z}}}_l\big) \leq R_l. \tag{7}$$

where R_l is the rank of the tensor associated to the l partition.

Number of parameters: As already explained, one of the most meaningful implications of low-rank tensor models is the fact that they provide a parsimonious description of the tensor, reducing its implicit number of degrees of freedom. The same is true for the decomposition in (6). To be concrete, the tensor $\check{\underline{\mathbf{Z}}}_l = \mathrm{ten}_{\mathcal{P}^{(l)}}(\underline{\mathbf{Z}}_l)$ has order $P^{(l)} = |\mathcal{P}^{(l)}|$, with the dimension of the pth mode being $\prod_{j=1}^{|\mathcal{P}_p^{(l)}|} |\mathcal{P}_p^{(l)}(j)|$. As a result, $\check{\underline{\mathbf{Z}}}_l$ having rank R_l implies that

$$R_l \sum_{p=1}^{P^{(l)}} \prod_{j=1}^{|\mathcal{P}_p^{(l)}|} |\mathcal{P}_p^{(l)}(j)|$$

parameters suffice to fully describe the $\prod_{i=1}^{I} N_i$ entries in $\check{\underline{\mathbf{Z}}}_l$. Summing across the different L factors implies that

$$\sum_{l=1}^{L} R_l \sum_{p=1}^{P^{(l)}} \prod_{j=1}^{|\mathcal{P}_p^{(l)}|} |\mathcal{P}_p^{(l)}(j)|$$

parameters suffice to fully describe the $\prod_{i=1}^{I} N_i$ entries in $\underline{\mathbf{Z}}$.

4. ALGORITHMIC IMPLEMENTATION

The decomposition introduced in (6) can be obtained by solving the following minimization problem:

$$\min_{\underline{\mathbf{Z}}_1 \dots \underline{\mathbf{Z}}_L} \Big\| \underline{\mathbf{X}} - \sum_{l=1}^{L} \underline{\mathbf{Z}}_l \Big\|_F \tag{8}$$
$$\text{s. t. } \operatorname{rank}\big(\operatorname{ten}_{\mathcal{P}^{(l)}}(\underline{\mathbf{Z}}_l)\big) \leq R_l.$$

The approach proposed in this section is to estimate each of the L tensors sequentially, so that when optimizing with respect to $\underline{\mathbf{Z}}_i$ the remaining tensors $\underline{\mathbf{Z}}_l$ with $l \neq i$ are kept fixed. As a result, the minimization problem to be solved in the ith step is:

$$\min_{\underline{\mathbf{Z}}_i} \Big\| \underline{\mathbf{X}} - \sum_{l \neq i}^{L} \underline{\mathbf{Z}}_l - \underline{\mathbf{Z}}_i \Big\|_F \tag{9}$$
$$\text{s. t. } \operatorname{rank}\big(\operatorname{ten}_{\mathcal{P}^{(i)}}(\underline{\mathbf{Z}}_i)\big) \leq R_i$$

for $i = 1, ..., L$. The constraint in (9) can be handled using a PARAFAC decomposition

$$\underline{\mathbf{Z}}_i = \sum_{j=1}^{R_i} [\mathbf{H}_1^i]_j \circledcirc ... \circledcirc [\mathbf{H}_{J_i}^i]_j, \tag{10}$$

so that (9) can be equivalently formulated as:

$$\min_{\mathbf{H}_1^i, \dots, \mathbf{H}_{J_i}^i} \Big\| \underline{\mathbf{X}} - \sum_{l \neq i}^{L} \underline{\mathbf{Z}}_l - \sum_{j=1}^{R_i} [\mathbf{H}_1^i]_j \circledcirc ... \circledcirc [\mathbf{H}_{J_i}^i]_j \Big\|_F. \tag{11}$$

The above problem is non-convex, but fixing all but one of the factors (say the jth one), it becomes linear in $\mathbf{H}_j^i$. Under this approach and unfolding the tensor into a matrix $\hat{\mathbf{X}}^i = \operatorname{mat}_p(\underline{\mathbf{X}} - \sum_{l \neq i}^{L} \underline{\mathbf{Z}}_l)$, we have the following update rule to constructing an Alternating Least Squares (ALS) algorithm:

$$\min_{\mathbf{H}_j^i} ||\hat{\mathbf{X}}^i - (\mathbf{H}_{J_i}^i \odot ... \odot \mathbf{H}_{j+1}^i \odot \mathbf{H}_{j-1}^i \odot ... \odot \mathbf{H}_1^i)(\mathbf{H}_j^i)^T||_F, \tag{12}$$

for all $j = 1, ..., J_i$. Once the J_i factors $\{\mathbf{H}_j^i\}_{j=1}^{J_i}$ have been obtained, then a) the ith tensor $\underline{\mathbf{Z}}_i$ is found using (10) and b) the problem in (9) is solved for the next i, with $i = 1, ..., L$. As a result, $\sum_{i=1}^{L} J_i$ instances of (12) need to be run. Note that, when solving (8) via (9)-(12), the order matters. The first $\underline{\mathbf{Z}}_l$ to be estimated provides the main (coarser) approximation, while the subsequent ones try to fit the residual error between the main tensor $\underline{\mathbf{X}}$ and the sum of the previously estimated components $\underline{\mathbf{Z}}_l$, providing a finer approximation. Due to the structure $\mathcal{P}^{(l)}$, which carries over $\underline{\mathbf{Z}}_l$, the order in which the tensors $\{\underline{\mathbf{Z}}_l\}_{l=1}^{L}$ are approximated is expected to generate variations in the results.

4.1. Constructing the partitions

The algorithm in the previous section assumes that the partitions $\mathcal{P}^{(1)}$,...,$\mathcal{P}^{(L)}$ are given. A simple generic approach to design $\mathcal{P}^{(1)}$,...,$\mathcal{P}^{(L)}$ is to rely on a *regular* multiresolution construction that splits the index set $\mathcal{I} = \{1, 2, ..., I\}$ into smaller sets with the *same cardinality*. More specifically, one can implement a sequential design with $L = I - 1$ steps for which, at step $l \in \{1, ..., L\}$ we split $\mathcal{I}$ into $l + 1$ index sets with (approximately) the same number of elements. The collection of $L = I - 1$ partitions $\mathcal{P}^{(1)}$,...,$\mathcal{P}^{(L)}$ is then naturally given by grouping together the sets obtained in each of those steps. To be more clear, let $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ be the floor and ceil operators and consider the collection of partitions $\mathcal{P}^{(1)}$,...,$\mathcal{P}^{(L)}$ with $L = I - 1$ and where the lth element is given by

$$\mathcal{P}^{(l)} = \{\mathcal{P}_n^{(l)}\}_{n=1}^{l+1}, \text{ with} \tag{13}$$
$$\mathcal{P}_n^{(l)} = \{\lceil (n-1)I/(l+1) \rceil, ..., \lfloor nI/(l+1) \rfloor\}.$$

In the above definition we have adopted the convention that, if x is a whole positive number, $\lfloor x \rfloor = x$ and $\lceil x \rceil = x + 1$. Clearly, the partition design in (13) is regular in the sense that it achieves $|\mathcal{P}^{(l)}| = l+1$ for all l and $|\mathcal{P}^{(l)}(n)| \approx I/(l+1)$ for $n = 1, ..., l+1$.

To gain insights, suppose for simplicity that our tensor $\underline{\mathbf{X}}$ of order I has size $\eta \times ... \times \eta$, i.e., that the value of N_i is the same across modes, then the number of parameters required to represent $\underline{\mathbf{X}}$ using the model in (6) and the partitions in (13) is approximately

$$\sum_{l=1}^{I-1} R_l (l+1) \eta^{I/(l+l)}, \tag{14}$$

which contrasts with the $\prod_{i=1}^{I} N_i = \eta^I$ entries in $\underline{\mathbf{X}}$.

Clearly, alternative ways to build the partitions $\mathcal{P}^{(1)}$,...,$\mathcal{P}^{(L)}$ are possible. This is especially relevant when prior knowledge exists and one can leverage it to group indexes based on known (di-)similarities among the underlying dimensions. Due to space limitations discussing such alternative partition techniques is out of the scope of this manuscript, but it is part of our ongoing work.

5. NUMERICAL EXPERIMENTS

The multi-resolution low-rank (MRLR) tensor decomposition scheme is numerically tested in three different scenarios: the first dealing with an amino acids dataset [13], the second one with a video signal [14], and the third one to approximate a multivariate function. The amino acids dataset is a three-mode tensor of size $5 \times 201 \times 61$. The video signal is composed of 173 frames of 1080×720 pixels each and three channels (R, G, and B). To reduce the computational and memory complexity requirements of the problem, the frames have been sub-sampled and the resolution has been lowered, resulting in a final four-mode tensor of size $9 \times 36 \times 54 \times 3$. Finally, the multidimensional function in the last scenario has $\mathbb{R}^3$ as its domain, with each of the three dimensions being discretized using 100 points, so that a tensor with 10^6 entries is obtained. The Tensorly Python package is used to benchmark the MRLR tensor decomposition against other tensor decomposition algorithms [15].

The amino acids tensor $\underline{\mathbf{X}}$ is approximated using a hierarchical structure of a matrix plus a three-mode tensor. The matrix can be build by unfolding the $5 \times 201 \times 61$ tensor in different ways. Here, two reshapes have been studied, a 201×305 unfolding (res-1), and a 1005×61 unfolding (aka res-2). The structure of the algorithm resembles that of a gradient-boosting-like approach [16]. First, the initial tensor is approximated by a low-rank structure. Then, the residual is approximated by a low-rank structure too. Subsequent residuals are also approximated if necessary. This sequential process can be started from the coarser unfolding, the matrix, or the other way around (reverse). In this experiment, both alternatives have been tested. The rank of the matrix unfolding is fixed while the rank of the three-mode tensor is gradually increased.

The performance of the algorithms has been measured in terms of Normalized Frobenius Error (NFE) between the true tensor $\underline{\mathbf{X}}$ and the approximation $\hat{\underline{\mathbf{X}}}$, which is given by

$$\text{NFE} = ||\underline{\mathbf{X}} - \hat{\underline{\mathbf{X}}}||_F / ||\underline{\mathbf{X}}||_F. \tag{15}$$

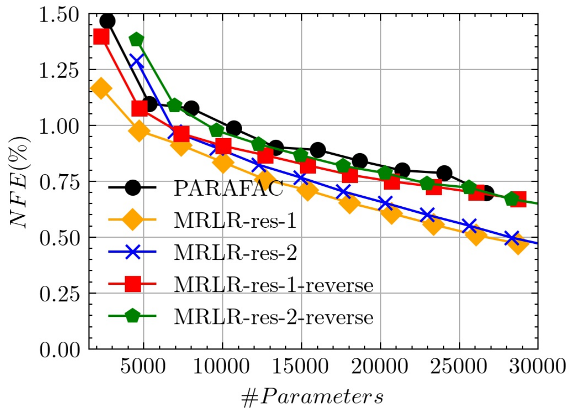

Fig. 1. Normalized Squared Frobenius Error (15) between the original $5 \times 201 \times 61$ amino acids tensor and its approximation obtained via the MRLR and the PARAFAC tensor decompositions when the number of parameters (tensor rank) is increased.

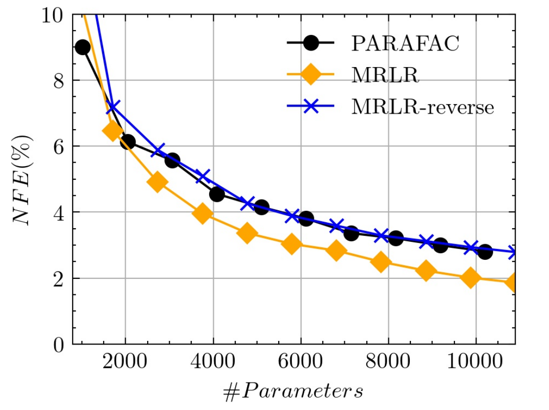

Fig. 2. Normalized Squared Frobenius Error (15) between the original $9 \times 36 \times 54 \times 3$ video signal tensor and its approximation obtained via the MRLR and the PARAFAC tensor decompositions when the number of parameters (tensor rank) is increased.

The results are reported in Fig. 1. The MRLR decomposition is compared to the PARAFAC decomposition. The res-1 unfolding of the matrix (square-like unfolding) seems to perform better than the res-2 unfolding (tall unfolding). Then, the approximation from the coarser to the finer arrangement beats the reverse one. Moreover, all the MRLR schemes outperform the PARAFAC one in terms of NFE for the same number of parameters. Indeed, the best-performing MRLR algorithm obtains roughly the same NFE as the PARAFAC decomposition using $10,000$ parameters less approximately.

In the second test case, the four-mode video tensor $\underline{\mathbf{X}}$ is unfolded into a 324×162 matrix and a $9 \times 36 \times 162$ three-mode tensor. The ranks of the matrix and the three-mode tensors have been fixed to 1. The rank of the four-mode tensor approximation is gradually increased. The results are provided in Fig. 2. Again, the coarser-to-finer arrangement outperforms both, the reverse (finer-to-coarser) arrangement, and the PARAFAC decomposition. It needs approximately $1,500$ parameters less to achieve the same NFE.

Finally, we tested the MRLR tensor decomposition in a third test case to approximate a multivariate function. Given a set of I input variables, with x_i denoting the ith input variable and $\mathcal{X}_i$ the set of all possible values of x_i, we are interested in functions that map any element $(x_1, ..., x_I) \in \mathcal{X}$ into a real value. When these functions are discrete, tensors can be used to model them efficiently. Continuous functions can be discretized/quantized. Tensor decomposition methods can then be leveraged for applications such as approximation, or denoising [17]. In such a context, we tested the MRLR tensor decomposition algorithm to model the following multivariate continuous function $f : \mathbb{R}^3 \mapsto \mathbb{R}$:

$$f(x_1, x_2, x_3) = \frac{x_1^2 + x_2^2}{e^{|x_2+x_3|}}. \tag{16}$$

Sampling a three dimensional grid of discrete values ranging from -5 to 5 with an step-size of 0.1 leads to a $100 \times 100 \times 100$ tensor $\underline{\mathbf{X}}$ that summarizes the multivariate function in (16). The tensor $\underline{\mathbf{X}}$ can be approximated using the MRLR tensor decomposition to leverage parsimony. The tensor $\underline{\mathbf{X}}$ is unfolded into a 10000×100 matrix, and the coarser-to-finer setup has been implemented. The performance of the MRLR tensor decomposition is again compared to that of the PARAFAC decomposition in terms of NFE for an increasing number of parameters. The results are shown in Fig. 3. As in previous scenarios, the MRLR decomposition outperforms the PARAFAC decomposition for the same number of parameters consistently. At some points, the difference between both algorithms is particularly high. For example, the MRLR tensor decomposition needs roughly $15,000$ parameters to achieve 1% of NFE, while the PARAFAC decomposition needs more than $30,000$ parameters.

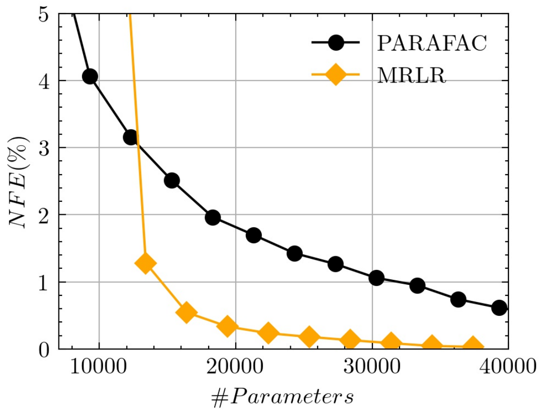

Fig. 3. Normalized Squared Frobenius Error (15) between the $100 \times 100 \times 100$ tensor sampled from the multivariate function in (16) and its approximation obtained via the MRLR and the PARAFAC tensor decompositions when the number of parameters (tensor rank) is increased.

6. CONCLUSIONS

This paper presented a parsimonious multi-resolution low-rank (MRLR) tensor decomposition to approximate a tensor as a sum of low-order tensor unfoldings. An Alternating Least Squares (ALS) algorithm was proposed to implement the MRLR tensor decomposition. Then, the MRLR tensor decomposition was compared against the PARAFAC decomposition in two real-case scenarios, and also in a multivariate function approximation problem. The MRLR tensor decomposition outperformed the PARAFAC decomposition for the same number of parameters, showing that it can efficiently leverage information defined at different dimensional orders.

7. REFERENCES

[1] R. Bro, “Parafac. tutorial and applications,” *Chemometrics and Intelligent Laboratory Systems*, vol. 38, no. 2, pp. 149–171, 1997.

[2] R. B. Cattell, “Parallel proportional profiles and other principles for determining the choice of factors by rotation,” *Psychometrika*, vol. 9, no. 4, pp. 267–283, 1944.

[3] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos, “Tensors for data mining and data fusion: Models, applications, and scalable algorithms,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, no. 2, pp. 1–44, 2016.

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

[5] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, “Tensor decomposition for signal processing and machine learning,” *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, 2017.

[6] R. A. Harshman, “Foundations of the parafac procedure: Models and conditions for an “explanatory” multimodal factor analysis,” *UCLA Working Papers Phonetics*, vol. 16, pp. 1–84, 1970.

[7] L. R. Tucker, “Some mathematical notes on three-mode factor analysis,” *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

[8] I. V. Oseledets, “Tensor-train decomposition,” *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011.

[9] L. Grasedyck, D. Kressner, and C. Tobler, “A literature survey of low-rank tensor approximation techniques,” *GAMM-Mitteilungen*, vol. 36, no. 1, pp. 53–78, 2013.

[10] D. Wang, F. Cong, and T. Ristaniemi, “Higher-order nonnegative candecomp/parafac tensor decomposition using proximal algorithm,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3457–3461.

[11] Q. Xie, Q. Zhao, D. Meng, and Z. Xu, “Kronecker-basis-representation based tensor sparsity and its applications to tensor recovery,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 8, pp. 1888–1902, 2017.

[12] O. Kaya and B. Uçar, “Parallel candecomp/parafac decomposition of sparse tensors using dimension trees,” *SIAM Journal on Scientific Computing*, vol. 40, no. 1, pp. C99–C130, 2018.

[13] R. Bro, “Multi-way analysis in the food industry-models, algorithms, and applications,” Ph.D. dissertation, University of Amsterdam (NL), 1998.

[14] S. Rozada, “Multi-resolution low-rank tensor decomposition,” https://github.com/sergiorozada12/multiresolution-tensor-decomposition, 2021.

[15] J. Kossaifi, Y. Panagakis, A. Anandkumar, and M. Pantic, “Tensorly: Tensor learning in python,” *arXiv preprint arXiv:1610.09555*, 2016.

[16] J. H. Friedman, “Greedy function approximation: a gradient boosting machine,” *Annals of Statistics*, pp. 1189–1232, 2001.

[17] N. Kargas and N. D. Sidiropoulos, “Supervised learning and canonical decomposition of multivariate functions,” *IEEE Transactions on Signal Processing*, vol. 69, pp. 1097–1107, 2021.