Krylov-Levenberg-Marquardt Algorithm for Structured Tucker Tensor Decompositions

Petr Tichavský, *Senior Member, IEEE*, Anh-Huy Phan, and Andrzej Cichocki, *Fellow, IEEE*

***Abstract*—Structured Tucker tensor decomposition models complete or incomplete multiway data sets (tensors), where the core tensor and the factor matrices can obey different constraints. The model includes block-term decomposition or canonical polyadic decomposition as special cases. We propose a very flexible optimization method for the structured Tucker decomposition problem, based on the second-order Levenberg-Marquardt optimization, using an approximation of the Hessian matrix by the Krylov subspace method. An algorithm with limited sensitivity of the decomposition is included. The proposed algorithm is shown to perform well in comparison to existing tensor decomposition methods.**

***Index Terms*—Alternating direction of multipliers, CANDECOMP, CANDELINC, Krylov subspace, Levenberg-Marquardt algorithm, PARAFAC, PARALIND, sensitivity, tensor chain, Tucker decomposition.**

I. Introduction

Tensor decompositions, especially the canonical polyadic (CP) tensor representations, have been increasingly popular since 70's and the number of papers and applications for tensor decomposition is increasing [1], [2].

The most popular tensor decompositions are the canonical polyadic decomposition (CPD) and the Tucker decomposition, and, more recently, tensor train (TT) [3], [4] and Tensor Chain (TC) [5], [6]. This paper is devoted to the structured Tucker tensor decomposition (STKD). A special case of STKD is a Block-Term Decomposition (BTD) [7] which has found applications in biomedical applications [8]. The BTD was implemented, to a certain extent, in a toolbox Tensorlab of De Lathauwer and his co-workers [9], [10]. A connection between STKD and TT/TC will also be shown in this paper.

We have recently proposed a Krylov-Levenberg-Marquardt (KLM) algorithm for computing the CP decomposition of tensors [11], [12]. The idea is to preserve the good numerical properties of the well-known second-order Levenberg-Marquardt (LM) algorithm but decrease its computational complexity by a low-rank approximation of the Hessian through the Krylov subspace [13], [14]. The algorithm uses a trick by which products of the approximate Hessian, $\mathbf{H}$, with a vector $\mathbf{x}$ can be computed very efficiently thanks to a special structure of the Hessian, without actually constructing the Hessian. The efficient computation of the product $\mathbf{Hx}$ has been already proposed in [15]. In this paper, we extend this idea to the Tucker decomposition, structured Tucker decompositions, weighted decompositions (incomplete tensors), and constrained decompositions.

The dimension of the Krylov subspace influences the quality of the Hessian approximation and is a design variable of the technique. A higher dimension results in increasing complexity for each iteration, but it decreases the number of iterations needed to achieve the convergence. When the dimension exceeds a certain limit, the performance does not improve any more and the algorithm becomes equivalent to the ordinary LM algorithm. In comparison with the Nonlinear Least Squares (NLS) algorithm [10] in the Tensorlab, which has a similar performance and applicability, KLM has one more tuning parameter. Some tuning might be necessary to achieve optimum performance, see the simulation section.

In some difficult scenarios, cost functions in most tensor decompositions have many local minima, and traditional algorithms often terminate in some of them. In such situations, the concept of decomposition with a limited sensitivity may help [11]. Such decompositions were applied in compression of convolutional layers of neural networks [16]. We adopt this concept for STKD and demonstrate its usefulness in simulations.

In many applications, it is required that elements of one or more factor matrices must be nonnegative or bounded at certain intervals. The most efficient technique of such constrained CP decomposition is probably either the Alternating Direction of Multipliers (ADMM) [17] or its variant with Nesterov iteration [18]. This method is usually combined with ALS, as proposed in Liavas and Sidiropoulos [19]. In a supplementary material to this paper we show that ADMM and KLM can be combined together to update all factor matrices jointly. Note that a similar technique has been applied recently to nonnegative matrix factorization in [20].

The main achievements of this paper can be summarized as follows.

1) Tensor chain decomposition. In Section III, we show that TC can be reformulated as the structured Tucker decomposition (STKD), introduced in Section II. Thus, the KLM algorithm can provide a new tool for TC.
2) Sensitivity. In Section IV, we extend the notion of sensitivity to STKD. The sensitivity, if used as a constraint in KLM, can significantly improve performance of KLM in some difficult scenarios, as we show later in the simulation section.

Manuscript received August 1, 2020; revised November 25, 2020 and January 15, 2021; accepted February 10, 2021. Date of publication February 16, 2021; date of current version March 29, 2021. This research was partially supported by the Ministry of Education and Science of the Russian Federation under Grant 14.756.31.0001. The work of Petr Tichavský was supported by the Czech Science Foundation through Project 20-17720S. The guest editor coordinating the review of this manuscript and approving it for publication was Dr. H. Chen. *(Corresponding author: Petr Tichavský.)*

Petr Tichavský is with the Czech Academy of Sciences, Institute of Information Theory, and Automation, 18200 Prague, Czech Republic (e-mail: tichavsk@utia.cas.cz).

Anh-Huy Phan and Andrzej Cichocki are with the Skolkovo Institute of Science and Technology (Skoltech), Moscow 143026, Russia (e-mail: a.phan@skoltech.ru; a.cichocki@skoltech.ru).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/JSTSP.2021.3059521.

Digital Object Identifier 10.1109/JSTSP.2021.3059521

1932-4553 © 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: UTIA. Downloaded on April 09,2021 at 08:57:56 UTC from IEEE Xplore. Restrictions apply.

3) Fast computing of $\mathbf{y} = \mathbf{H}\mathbf{x}$. Here, $\mathbf{H}$ is the approximate Hessian of the STKD problem, and $\mathbf{x}$ is an arbitrary vector of appropriate dimension. This is an important ingredience of KLM. We acknowledge that this computation was already discussed in [10] and is implemented in Tensorlab, but we present it in closed form and for arbitrary dimension (Appendix C of the Supplementary materials).
4) Extensions. We present several extension of KLM in Section VII to show the high flexibility of the method. The most important one among them is probably the incorporation of smooth constraints (namely for sensitivity).

Section VIII presents the simulation results, mainly focused on the block-term decomposition and tensor chain modeling. Section IX concludes the paper.

Notation. Boldface lowercase and uppercase letters will be used for vectors and matrices, respectively. Tensors are written in calligraphic letters, $\mathcal{T}, \mathcal{K}, \mathcal{L}, \mathcal{Z}$, etc. The corresponding matricizations along the mode i, $i = 1, 2, 3$, will be denoted as $\mathbf{T}_{(i)}, \mathbf{K}_{(i)}, \mathbf{L}_{(i)}, \mathbf{Z}_{(i)}$, respectively. Superscript T denotes transpose, $\|\cdot\|_F$ represents the Frobenius norm of the argument, $\star$ is the elementwise (Hadamard) product, $\otimes$ is the Kronecker product, $\mathbf{I}$ represents the identity matrix, and $\text{vec}(\cdot)$ is operator of vectorization, which stacks all elements of a matrix or a tensor in one column vector.

II. Structured Tucker Tensor Decomposition

For simplicity of presentation, in this section we consider only the decomposition of order-three tensors. The models and algorithms allow a straightforward extension to higher-order tensors, as shown in Appendix C in Supplementary materials.

Assume that we are given a data tensor $\mathcal{T}$ of the size $I_1 \times I_2 \times I_3$ with elements T_{ijk} which can be either full, i.e., available, or incomplete. In the latter case, we are given an indicator tensor $\mathcal{W}$ of the same size, with binary elements W_{ijk}, which indicate whether the element T_{ijk} is available or not.

The Tucker decomposition of the tensor with multilinear rank (R_1, R_2, R_3) will be denoted as a quartet $[[\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{C}]]$ with a core tensor $\mathcal{K}$ of the size $R_1 \times R_2 \times R_3$ and factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ of the sizes $I_i \times R_i$, $i = 1, 2, 3$, respectively, such that

$$T_{ijk} \approx \sum_{p=1}^{R_1} \sum_{q=1}^{R_2} \sum_{r=1}^{R_3} K_{pqr} A_{ip} B_{jq} C_{kr}. \tag{1}$$

Symbolically, we shall write [2]

$$\mathcal{T} \approx [[\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{C}]]. \tag{2}$$

The Tucker decomposition is, in general, not unique. Usually, the ambiguity is reduced by assuming that the columns of the factor matrices are mutually orthogonal and have unit-norm. In some other cases, e.g., when the core tensor and/or the factor matrices ought to be nonnegative, this condition cannot be fulfilled.

The structured Tucker decomposition often means that only a part of the core tensor elements is nonzero. For example, canonical polyadic tensor decomposition is a special case of such a decomposition. The core tensor has a cubic shape, and all nonzero elements of the core lie on its space diagonal. Mathematically, $K_{ijk} = K_{iii}\delta_{ij}\delta_{ik}$, where δ_{ij} is the Kronecker delta.

Another important STKD is block term decomposition (BTD), see Fig. 1 or Fig. 2. Here, the nonzero elements of $\mathcal{K}$ form rectangular blocks along the spatial diagonal of $\mathcal{K}$. For BTD with non-overlapping core tensors, the decomposition can

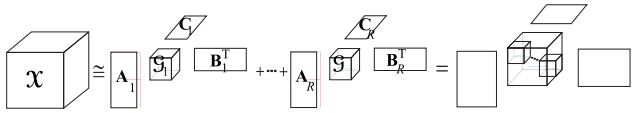

Fig. 1. Illustration of BTD as a Structured TKD.

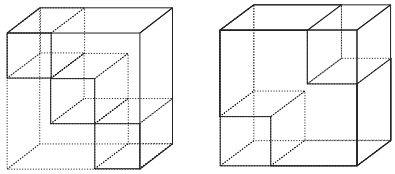

Fig. 2. Illustration of non-overlapping and overlapping core tensors in BTD.

be written as a sum of several Tucker tensors of the same size but of smaller multilinear ranks. In BTD, the data is explained by disjoint subspaces, each spanned by sub-factor matrices for each block term.

The structured TKD allows a more flexible BTD with overlapping or shared partial loading components in each mode, see Fig. 2.

The CPD and BTD are just only examples. It is possible to consider lower-triangular core tensor, what is a generalization of lower-triangular matrix. Here, $K_{ijk} = 0$ for all $i, j, k = 1, \ldots, R$ such that $i + j + k > R + 2$.

The zero and nonzero elements of $\mathcal{K}$ can be described by a tensor $\mathcal{L}$ of the same size as $\mathcal{K}$, composed of nulls and ones only, such that the following implication holds: $L_{ijk} = 0 \Rightarrow K_{ijk} = 0$. We will use the symbol $\mathcal{K}(\mathcal{L})$ for the column vector composed of the nonzero elements of $\mathcal{K}$. The number of nonzero elements of $\mathcal{K}$ is the number of ones in $\mathcal{L}$ and will be denoted $|\mathcal{L}|$.

The total number of parameters in the STKD is then $|\mathcal{L}| + I_1R_1 + I_2R_2 + I_3R_3$. The decomposition is meaningful if this number is significantly smaller than the total number of the tensor, $I_1I_2I_3$ in the case of unweighted decomposition, or the number of available tensor elements that is $|\mathcal{W}|$, i.e., the number of ones in $\mathcal{W}$. It is generally permitted that the size of the core tensor may exceed the original tensor's size in some dimensions, like the tensor rank in CPD may exceed some (or all) of the dimensions of the tensor.

Note that the tensor model is linear in the factor matrices and the core tensor individually, but not jointly. The "workhorse" Alternating Least Squares (ALS) algorithm consists of minimizing the criterion

$$\varphi(\boldsymbol{\theta}) = \|\mathcal{T} - [[\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{C}]]\|_F^2 \tag{3}$$

or (in the incomplete tensor case)

$$\varphi_W(\boldsymbol{\theta}) = \|\mathcal{W}^{1/2} \star (\mathcal{T} - [[\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{C}]])\|_F^2 \tag{4}$$

with respect to components (vectors) of

$$\boldsymbol{\theta} = [\mathcal{K}(\mathcal{L}); \text{vec}\,\mathbf{A}; \text{vec}\,\mathbf{B}; \text{vec}\,\mathbf{C}] \tag{5}$$

individually, in an alternating manner. In (4), $\mathcal{W}^{1/2}$ refers to the elementwise squared root of $\mathcal{W}$.[1] The separation by semicolons in (5) indicates stacking the vectors one above the other (like in Matlab). $\boldsymbol{\theta}$ is a long (tall) vector.

Unfortunately, it is known that ALS often exhibits a slow convergence. Its convergence is at most linear. Second-order

[1]Indeed, if the weights are only zeros and ones, the square root does not change anything.

Authorized licensed use limited to: UTIA. Downloaded on April 09,2021 at 08:57:56 UTC from IEEE Xplore. Restrictions apply.

methods like Gauss-Newton or damped Gauss-Newton method applied to CPD problem (Levenberg-Marquardt, (LM)) can obtain a significantly faster convergence [21], [22]. These methods require computing the Hessian for the problem, which can be difficult. This problem is alleviated in the Krylov-Levenberg-Marquardt method [11], [12], and is extended to the structured Tucker decomposition in this paper.

The LM or KLM methods permit handling an important generalization of the tensor model, where the parameter vector is subject to a linear transformation. This will be explained in Section VI.

III. Link Between STKD and Tensor Chain

Tensor Chain (TC) [5], [6] represents an order-N tensor in terms of N order-3 tensors that are interconnected in a cyclic network as

$$\mathcal{X} = \mathcal{G}_1 \bullet \mathcal{G}_2 \bullet \cdots \bullet \mathcal{G}_N \bullet \mathcal{G}_1 \tag{6}$$

where $\mathcal{G}_n$ are core tensors of size $R_n \times I_n \times R_{n+1}$, $R_1 = R_{N+1}$, "$\bullet$" represents the train contraction of two consecutive core tensors. A core tensor is connected to two core tensors, which forms a chain or train of core tensors. For example, the core tensor $\mathcal{G}_1$ is connected to $\mathcal{G}_N$ and $\mathcal{G}_2$, whereas the core tensor $\mathcal{G}_N$ is connected to $\mathcal{G}_{N-1}$ and $\mathcal{G}_1$. Numerically, the $(j_1, \ldots, j_N)$-th element of the tensor is given as

$$X_{j_1,\ldots,j_N} = \text{tr}\left(\mathbf{G}_{j_1}^{(1)}\mathbf{G}_{j_2}^{(2)}\cdots\mathbf{G}_{j_N}^{(N)}\right) \tag{7}$$

where $\mathbf{G}_{j_n}^{(n)} = \mathcal{G}_n(:, j_n, :)$ is the j_n-th slice of $\mathcal{G}_n$, $n = 1, \ldots, N$.

The tensor chain is an extension of the earlier tensor train [3]. It was designed to remove problems in TT decomposition with dimensions of the core tensors in the middle of the chain, which are often unbalanced and grow dramatically with the dimension of the tensor. Connecting the first and last core tensors of the model makes it more balanced because there are no first and last core tensors.

Unfortunately, it appears that the loop in TC may lead to severe numerical instability in finding the best approximation especially when $R_n R_{n+1} > I_n$, see Theorem 14.1.2.2 [23] and [24]. ALS and DMRG algorithms for computing the TC decomposition were proposed in [5].

This section proposes another interpretation of the model and possibly new algorithms to estimate it. For simplicity, we consider order-3 tensors only. The decomposition of higher-order tensors can be extended straightforwardly. We claim that

$$\mathcal{X} = [[\mathcal{M}; \mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3]] \tag{8}$$

where $\mathbf{U}_n$ are mode-2 unfolding of the core tensors $\mathcal{G}_n$ and are of size $I_n \times R_n R_{n+1}$, and $\mathcal{M}$ is a binary tensor of size $R_3 R_1 \times R_1 R_2 \times R_2 R_3$ associated with the multiplication of two matrices of sizes $R_3 \times R_1$ and $R_1 \times R_2$ [25]. The tensor has elements

$$M_{\alpha\beta\gamma} = \delta_{in}\delta_{jk}\delta_{\ell m} \tag{9}$$

where $\alpha = (i-1)R_1 + j$; $\beta = (k-1)R_2 + l$; $\gamma = (m-1)R_3 + n$ for $i, n = 1, \ldots, R_3$; $j, k = 1, \ldots, R_1$; $l, m = 1, \ldots, R_2$. See Appendix B in the suplementary materials for proof. An example of the distribution of zeros and ones in the tensor $\mathcal{M}$ is presented in the simulation section.

Thanks to (8), the TC decomposition can be interpreted as a structured TKD with a fixed core tensor. We can apply the KLM algorithm described below to find the factor matrices, $\mathbf{U}_n$ and, consequently, the core tensors $\mathcal{G}_n$, $n = 1, 2, 3$.

IV. Sensitivity of the Decomposition

The notion of *sensitivity* was introduced for CPD [11]. The motivation is that in some difficult cases, the CPD is not unique, or there is additive noise. In practical applications, traditional decomposition algorithms may lead to diverging solutions. In particular, it occurs in all unconstrained algorithms (ALS, LM, KLM) that some rank-one components in the decomposition converge to infinity in the norm, and the decomposition terms partially cancel each other. The basic remedy to this problem is applying ℓ_1-norm, or ℓ_2- norm regularization [27], [28] which is rather heuristic and not easy to adjust. A more sophisticated approach is to use either the so-called error-preserving correction (EPC) [26] or applying estimates with limited sensitivity [11].

In this section, we extend the definition of sensitivity to the structured Tucker decomposition as follows:

$$\begin{aligned} s(\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{C}) = \lim_{\sigma^2 \to 0} \frac{1}{\sigma^2} \mathrm{E}\{\| [[\mathcal{K} + \delta\mathcal{K}, \mathbf{A} + \delta\mathbf{A}, \\ \mathbf{B} + \delta\mathbf{B}, \mathbf{C} + \delta\mathbf{C}]] - [[\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{C}]] \|_F^2\}, \end{aligned} \tag{10}$$

where $\delta\mathcal{K}, \delta\mathbf{A}, \delta\mathbf{B}$ and $\delta\mathbf{C}$ are random Gaussian-distributed perturbations of the core tensor and the factor matrices with independently distributed elements of zero mean and variance σ^2, and E is the expectation operator. If some core tensor elements are constrained to be zero, we set the corresponding elements of $\delta\mathcal{K}$ to zero as well.

The sensitivity can be expressed directly in terms of the core tensor elements and the factor matrices. After a straightforward computation (see Appendix B for details), we get

$$\begin{aligned} s(\boldsymbol{\theta}) &= \frac{1}{\sigma^2}\mathrm{E}\{\| [[\delta\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{C}]] \|_F^2\} \\ &\quad + \frac{1}{\sigma^2}\mathrm{E}\{\| [[\mathcal{K}, \delta\mathbf{A}, \mathbf{B}, \mathbf{C}]] \|_F^2\} \\ &\quad + \frac{1}{\sigma^2}\mathrm{E}\{\| [[\mathcal{K}, \mathbf{A}, \delta\mathbf{B}, \mathbf{C}]] \|_F^2\} \\ &\quad + \frac{1}{\sigma^2}\mathrm{E}\{\| [[\mathcal{K}, \mathbf{A}, \mathbf{B}, \delta\mathbf{C}]] \|_F^2\} \\ &= \mathbf{1}^T \text{vec}\,[[\mathcal{L}, \mathbf{A} \star \mathbf{A}, \mathbf{B} \star \mathbf{B}, \mathbf{C} \star \mathbf{C}]] \\ &\quad + I_1 \| [[\mathcal{K}, \mathbf{I}_{R_1}, \mathbf{B}, \mathbf{C}]] \|_F^2 \\ &\quad + I_2 \| [[\mathcal{K}, \mathbf{A}, \mathbf{I}_{R_2}, \mathbf{C}]] \|_F^2 \\ &\quad + I_3 \| [[\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{I}_{R_3}]] \|_F^2, \end{aligned} \tag{11}$$

where $\mathbf{1}$ is a vector composed of 1's, and $\mathcal{L}$ is the tensor indicating the nonzero elements of the core tensor $\mathcal{K}$.

A. Sensitivity for CPD

In the special case of CPD, the core tensor $\mathcal{K}$ comprises only R nonzero elements on its diagonal. These elements can be set to one because the energy of each rank-one factor can be absorbed in the factor matrices. Thus we can take $\mathcal{K}$ as fixed, not being perturbed in the definition of sensitivity. Then, the expression

Authorized licensed use limited to: UTIA. Downloaded on April 09,2021 at 08:57:56 UTC from IEEE Xplore. Restrictions apply.

(11) translates in

$$s(\mathbf{A},\mathbf{B},\mathbf{C}) = \sum_{r=1}^{R}(I_1\,\|\mathbf{b}_r\|^2\|\mathbf{c}_r\|^2 + I_2\,\|\mathbf{a}_r\|^2\|\mathbf{c}_r\|^2 + I_3\,\|\mathbf{a}_r\|^2\|\mathbf{b}_r\|^2) \quad (12)$$

where $\mathbf{a}_r, \mathbf{b}_r$ and $\mathbf{c}_r$ are columns of the factor matrices. Note the missing factors I_1, I_2, I_3 in the corresponding expression in [11]. The difference between (12) and [11] disappears if the scaling ambiguity of the CP decomposition is taken into account. The columns in two factor matrices can be arbitrarily scaled, and the change can be compensated by an appropriate adjustment in the third-factor matrix. In other words, it holds that

$$[[\mathbf{a}_r, \mathbf{b}_r, \mathbf{c}_r]] = [[\alpha_r\mathbf{a}_r, \beta_r\mathbf{b}_r, \gamma_r\mathbf{c}_r]]$$

for any $\alpha_r, \beta_r, \gamma_r$ such that $\alpha_r\beta_r\gamma_r = 1$ for $r = 1, \ldots, R$. The scales should be selected to minimize the sensitivity (12); otherwise, it could be arbitrarily large. It can be easily shown that the minimum sensitivity is achieved if

$$I_1\,\|\mathbf{b}_r\|^2\|\mathbf{c}_r\|^2 = I_2\,\|\mathbf{a}_r\|^2\|\mathbf{c}_r\|^2 = I_3\,\|\mathbf{a}_r\|^2\|\mathbf{b}_r\|^2.$$

The resultant expression for sensitivity which minimizes (12) with respect to the scale changes is

$$s^*(\mathbf{A},\mathbf{B},\mathbf{C}) = 3(I_1I_2I_3)^{1/3}\sum_{r=1}^{R}(\|\mathbf{a}_r\|\|\mathbf{b}_r\|\|\mathbf{c}_r\|)^{4/3}. \quad (13)$$

If the columns of the factor matrices are normalized so that $\|\mathbf{a}_r\| = \|\mathbf{b}_r\| = \|\mathbf{c}_r\|$, as is recommended in [11], then the expression (13) is equivalent to the erroneous expression in [11] up to a fixed multiplicative factor.

V. Krylov-Levenberg-Marquardt Algorithm

The Levenberg-Marquardt algorithm consists in a sequence of iterations

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}' = \boldsymbol{\theta} - (\mathbf{H} + \mu\mathbf{I})^{-1}\mathbf{g}$$

where an error gradient $\mathbf{g}$, and an approximate Hessian $\mathbf{H}$ are defined through a Jacobi matrix $\mathbf{J}$ as

$$\mathbf{J} = \frac{\partial \text{vec}(\mathcal{T})}{\partial \boldsymbol{\theta}} \quad (14)$$

$$\mathbf{g} = \mathbf{J}^T\mathbf{W}\text{vec}(\mathcal{T} - \hat{\mathcal{T}}) \quad (15)$$

$$\mathbf{H} = \mathbf{J}^T\mathbf{W}\mathbf{J} \quad (16)$$

$\mathbf{W} = \text{diag}(\text{vec}(\mathcal{W}))$, and μ is a damping parameter that is updated through the iterations. In short, a new vector $\boldsymbol{\theta}$ is accepted if it reduces the cost function value. Otherwise, the step is skipped, and another estimate with a higher value of μ is applied. For more details of the update rule for μ see, e.g., [29], [30].

In the Krylov-Levenberg-Marquardt algorithm, the expression $(\mathbf{H} + \mu\mathbf{I})^{-1}\mathbf{g}$ is replaced by the approximation

$$(\mathbf{H} + \mu\mathbf{I})^{-1}\mathbf{g} \approx \frac{1}{\mu}\mathbf{g} - \frac{1}{\mu}\mathbf{U}(\mu\mathbf{Q}^{-1} + \mathbf{U}^T\mathbf{U})^{-1}(\mathbf{U}^T\mathbf{g}) \quad (17)$$

where columns of matrix $\mathbf{U}$ form an orthogonal basis of the so-called Krylov subspace that is the linear hull of

$$[\mathbf{g}, \mathbf{H}\mathbf{g}, \mathbf{H}^2\mathbf{g}, \ldots, \mathbf{H}^{M-1}\mathbf{g}] \quad (18)$$

and

$$\mathbf{Q} = \mathbf{U}^T\mathbf{H}\mathbf{U}. \quad (19)$$

The integer M representing the Krylov subspace dimension is a design parameter controlling the approximation accuracy. The matrix $\mathbf{U}$ is obtained through a Gramm-Schmidt orthogonalization process, and $\mathbf{Q}$ is received as a side product of the process. See [11], [12] for more details.

The total complexity of computing (17) depends on the complexity of the products $\mathbf{H}\mathbf{x}$ and on the order of the approximation M. If the complexity of the product is denoted C and $N_p = |\mathcal{L}| + I_1R_1 + I_2R_2 + I_3R_3$ is the total number of the parameters of the model, then the complexity of (17) is $O(MC + M^2N_p + M^3)$. If M is not too high, then the complexity per iteration grows roughly linearly with M. Note that M is the number of the $\mathbf{y} = \mathbf{H}\mathbf{x}$ operations used in each update.

We will see in the simulation section that the optimum parameter M is related to the complexity of the problem, namely to the rank R in the CP decomposition or size of the kernel tensor in STKD. A rule of thumb might be that M should be comparable to the rank or the size of the kernel.

VI. Fast Computing of $\mathbf{y} = \mathbf{H}\mathbf{x}$

In the case of unweighted CPD, this part is well explained in [15]. The case of weighted CPD is treated in [12]. In this section, we show that this product can be computed efficiently for the unweighted Tucker decomposition. The extensions for weighted Tucker and other STKDs will be given in the next section.

Assume that a data tensor to be decomposed is

$$\mathcal{T} \approx \mathcal{T}(\boldsymbol{\theta}) = [[\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{C}]], \quad (20)$$

where $\boldsymbol{\theta}$ is composed of elements of the factor matrices

$$\boldsymbol{\theta} = [\text{vec}(\mathcal{K}); \text{vec}(\mathbf{A}); \text{vec}(\mathbf{B}); \text{vec}(\mathbf{C})]. \quad (21)$$

The Jacobi matrix has now four parts,

$$\mathbf{J} = \frac{\partial \text{vec}(\mathcal{T})}{\partial \boldsymbol{\theta}} = [\mathbf{J}_K, \mathbf{J}_A, \mathbf{J}_B, \mathbf{J}_C]. \quad (22)$$

To deal with products of the type $\mathbf{J}\mathbf{x}$, we write the arbitrary vector $\mathbf{x}$ as,

$$\mathbf{x} = [\text{vec}\,\mathcal{X}_K; \text{vec}\,\mathbf{X}_A; \text{vec}\,\mathbf{X}_B; \text{vec}\,\mathbf{X}_C] \quad (23)$$

where $\mathcal{X}_K$ is a tensor of the shape of $\mathcal{K}$, and $\mathbf{X}_A, \mathbf{X}_B$, and $\mathbf{X}_C$ are matrices of the sizes of $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$, respectively. Similarly, the outcome $\mathbf{y} = \mathbf{H}\mathbf{x}$ would have four parts as well,

$$\mathbf{y} = [\text{vec}\,\mathcal{Y}_K; \text{vec}\,\mathbf{Y}_A; \text{vec}\,\mathbf{Y}_B; \text{vec}\,\mathbf{Y}_C] \quad (24)$$

where the tensor $\mathcal{Y}_K$ and matrices $\mathbf{Y}_A, \mathbf{Y}_B, \mathbf{Y}_C$ have the same size as $\mathcal{K}$, $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, respectively.

Let us consider first computing the product $\mathbf{y} = \mathbf{H}\mathbf{x}$ for the unweighted case, where $\mathbf{H} = \mathbf{J}^T\mathbf{J}$. It is easy to see that

$$\begin{aligned}\mathbf{J}\mathbf{x} &= \mathbf{J}_K\text{vec}\,\mathcal{X}_K + \mathbf{J}_A\text{vec}\,\mathbf{X}_A + \mathbf{J}_B\text{vec}\,\mathbf{X}_B + \mathbf{J}_C\text{vec}\,\mathbf{X}_C \\ &= \text{vec}\,[[\mathcal{X}_K, \mathbf{A}, \mathbf{B}, \mathbf{C}]] + \text{vec}\,[[\mathcal{K}, \mathbf{X}_A, \mathbf{B}, \mathbf{C}]] \\ &\quad + \text{vec}\,[[\mathcal{K}, \mathbf{A}, \mathbf{X}_B, \mathbf{C}]] + \text{vec}\,[[\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{X}_C]] \\ &= \text{vec}\,\mathcal{Z} \end{aligned} \quad (25)$$

Authorized licensed use limited to: UTIA. Downloaded on April 09,2021 at 08:57:56 UTC from IEEE Xplore. Restrictions apply.

where

$$\mathcal{Z} = [[\mathcal{X}_K, \mathbf{A}, \mathbf{B}, \mathbf{C}]] + [[\mathcal{K}, \mathbf{X}_A, \mathbf{B}, \mathbf{C}]] + [[\mathcal{K}, \mathbf{A}, \mathbf{X}_B, \mathbf{C}]] + [[\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{X}_C]]. \quad (26)$$

We need to compute

$$\mathbf{y} = \mathbf{J}^T\mathbf{J}\mathbf{x} = [\mathbf{J}_K; \mathbf{J}_A; \mathbf{J}_B; \mathbf{J}_C]^T\mathbf{J}\mathbf{x}. \quad (27)$$

Note that the matricization of the tensor $\mathcal{T}$ in (20) along the first mode can be written as

$$\mathbf{T}_{(1)} = \mathbf{A}[[\mathcal{K}, \mathbf{I}_{R_1}, \mathbf{B}, \mathbf{C}]]_{(1)}, \quad (28)$$

where $\mathbf{I}_{R_1}$ is the identity matrix of size $R_1 \times R_1$, and therefore, using the identity $\text{vec}(\mathbf{ABC}) = (\mathbf{C}^T \otimes \mathbf{A})\text{vec}\,\mathbf{B}$, it holds

$$\text{vec}\,\mathcal{T} = \text{vec}\,\mathbf{T}_{(1)} = \{[[\mathcal{K}, \mathbf{I}_{R_1}, \mathbf{B}, \mathbf{C}]]_{(1)}^T \otimes \mathbf{I}_{I_1}\}\text{vec}\,\mathbf{A}. \quad (29)$$

Hence, it follows that

$$\mathbf{J}_A = \frac{\partial \text{vec}\,\mathcal{T}}{\partial \text{vec}\,\mathbf{A}} = [[\mathcal{K}, \mathbf{I}_{R_1}, \mathbf{B}, \mathbf{C}]]_{(1)}^T \otimes \mathbf{I}_{I_1}. \quad (30)$$

Finally,

$$\begin{aligned}\mathbf{J}_A^T\mathbf{J}\mathbf{x} &= \{[[\mathcal{K}, \mathbf{I}_{R_1}, \mathbf{B}, \mathbf{C}]]_{(1)} \otimes \mathbf{I}_{I_1}\}\text{vec}\,\mathcal{Z} \\ &= \{[[\mathcal{K}, \mathbf{I}_{R_1}, \mathbf{B}, \mathbf{C}]]_{(1)} \otimes \mathbf{I}_{I_1}\}\text{vec}\,\mathbf{Z}_{(1)} \\ &= \text{vec}\,\{\mathbf{Z}_{(1)}[[\mathcal{K}, \mathbf{I}_{R_1}, \mathbf{B}, \mathbf{C}]]_{(1)}^T\}. \quad (31)\end{aligned}$$

Similarly, we get

$$\mathbf{J}_B^T\mathbf{J}\mathbf{x} = \text{vec}\,\{\mathbf{Z}_{(2)}[[\mathcal{K}, \mathbf{A}, \mathbf{I}_{R_2}, \mathbf{C}]]_{(2)}^T\} \quad (32)$$

$$\mathbf{J}_C^T\mathbf{J}\mathbf{x} = \text{vec}\,\{\mathbf{Z}_{(3)}[[\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{I}_{R_3}]]_{(3)}^T\}. \quad (33)$$

For the core tensor we obtain

$$\mathbf{J}_K^T\mathbf{J}\mathbf{x} = \text{vec}\,\{[[\mathcal{Z}, \mathbf{A}^T, \mathbf{B}^T, \mathbf{C}^T]]\}. \quad (34)$$

This completes the computation of $\mathbf{y}$ in (27). What remains is to provide a recipe for computing the error gradient $\mathbf{g}$ in (15). It has four parts as well,

$$\mathbf{g} = [\mathbf{g}_K; \mathbf{g}_A; \mathbf{g}_B; \mathbf{g}_C]. \quad (35)$$

Let $\mathcal{E}$ be the approximation error, i.e., the difference between the given tensor and the model,

$$\mathcal{E} = \mathcal{T} - [[\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{C}]]. \quad (36)$$

Then we easily find that

$$\mathbf{g}_K = \text{vec}\,[[\mathcal{E}, \mathbf{A}^T, \mathbf{B}^T, \mathbf{C}^T]] \quad (37)$$

$$\mathbf{g}_A = \text{vec}\,\{\mathbf{E}_{(1)}[[\mathcal{K}, \mathbf{I}_{R_1}, \mathbf{B}, \mathbf{C}]]_{(1)}^T\} \quad (38)$$

$$\mathbf{g}_B = \text{vec}\,\{\mathbf{E}_{(2)}[[\mathcal{K}, \mathbf{A}, \mathbf{I}_{R_2}, \mathbf{C}]]_{(2)}^T\} \quad (39)$$

$$\mathbf{g}_C = \text{vec}\,\{\mathbf{E}_{(3)}[[\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{I}_{R_3}]]_{(3)}^T\}. \quad (40)$$

If the size of the core tensor $\mathcal{K}$ is smaller than the size of the data tensor $\mathcal{T}$, it might be convenient to use the following alternative expressions to evaluate (31)–(33) and (38)–(40),

$$\mathbf{J}_A^T\mathbf{J}\mathbf{x} = \text{vec}\,\{[[\mathcal{Z}, \mathbf{I}_{I_1}, \mathbf{B}^T, \mathbf{C}^T]]_{(1)}\mathbf{K}_{(1)}^T\} \quad (41)$$

$$\mathbf{J}_B^T\mathbf{J}\mathbf{x} = \text{vec}\,\{[[\mathcal{Z}, \mathbf{A}^T, \mathbf{I}_{I_2}, \mathbf{C}^T]]_{(2)}\mathbf{K}_{(2)}^T\} \quad (42)$$

$$\mathbf{J}_C^T\mathbf{J}\mathbf{x} = \text{vec}\,\{[[\mathcal{Z}, \mathbf{A}^T, \mathbf{B}^T, \mathbf{I}_{I_3}]]_{(3)}\mathbf{K}_{(3)}^T\} \quad (43)$$

$$\mathbf{g}_A = \text{vec}\,\{[[\mathcal{E}, \mathbf{I}_{I_1}, \mathbf{B}^T, \mathbf{C}^T]]_{(1)}\mathbf{K}_{(1)}^T\} \quad (44)$$

$$\mathbf{g}_B = \text{vec}\,\{[[\mathcal{E}, \mathbf{A}^T, \mathbf{I}_{I_2}, \mathbf{C}^T]]_{(2)}\mathbf{K}_{(2)}^T\} \quad (45)$$

$$\mathbf{g}_C = \text{vec}\,\{[[\mathcal{E}, \mathbf{A}^T, \mathbf{B}^T, \mathbf{I}_{I_3}]]_{(3)}\mathbf{K}_{(3)}^T\}. \quad (46)$$

We note that neither computation of the error gradient nor the product $\mathbf{y} = \mathbf{H}\mathbf{x}$ requires larger memory than the memory needed for storing the data tensor $\mathcal{T}$, $I_1I_2I_3$ or the core tensor $\mathcal{K}$, $R_1R_2R_3$. The complexity of the computation of the product $\mathbf{H}\mathbf{x}$ is $C = O((R_1 + R_2 + R_3)I_1I_2I_3 + (I_1 + I_2 + I_3)R_1R_2R_3)$.

VII. Extensions

In this section, we extend the results of the previous section in five different directions. Note that some of these extensions have been already proposed in [10].

A. Weighted Tucker Decomposition

The vector representing the error gradient in this case is computed according to (37) and (38)–(40) or (44)–(46) with the difference that the error tensor $\mathcal{E}$ in (36) is replaced with $\mathcal{E}_W = \mathcal{E} \star \mathcal{W}$. For computing the product $\mathbf{y} = \mathbf{H}\mathbf{x}$ we need to replace the tensor $\mathcal{Z}$ from (26) in (31)–(34) or (41)–(43) and (34) with tensor $\mathcal{Z}_W = \mathcal{Z} \star \mathcal{W}$.

B. Nulls in the Core Tensor

Assume that the nonzero elements of the core tensor $\mathcal{K}$ are indicated by the tensor $\mathcal{L}$. The vector of the error gradient will be shorter in that case,

$$\mathbf{g} = [\mathbf{g}_K(\mathcal{L}); \mathbf{g}_A; \mathbf{g}_B; \mathbf{g}_C]. \quad (47)$$

The product $\mathbf{y} = \mathbf{H}\mathbf{x}$ can be computed in the same way as in the previous case with the difference that the tensor $\mathcal{X}_K$ is obtained from the first $|\mathcal{L}|$ elements of the vector $\mathbf{x}$. These elements are distributed in $\mathcal{X}_K$ in accord with the tensor $\mathcal{L}$, and the remaining elements will be zero. Similarly, after computing $\mathcal{Y}_K$, the first $|\mathcal{L}|$ elements of $\mathbf{y}$ are obtained as $\mathcal{Y}_K(\mathcal{L})$.

C. Fixed Core Tensor

In some applications, the core tensor can be assumed to be fixed and known in advance. For example, the CP decomposition can be cast in this class: the core tensor is spatially diagonal. We can fix its elements to 1, because the energy of each rank-one term can be distributed in the factor matrices. Another example is the tensor chain in Section III.

The KLM algorithm presented above can be modified to the case with fixed $\mathcal{K}$ by omitting the initial part of the parameter $\boldsymbol{\theta}$ containing elements of $\mathcal{K}$. Similarly, the vector representing the error gradient will be shorter, e.g.,

$$\mathbf{g} = [\mathbf{g}_A; \mathbf{g}_B; \mathbf{g}_C], \quad (48)$$

and similarly the vectors $\mathbf{x}$ and $\mathbf{y}$ in the equation $\mathbf{y} = \mathbf{H}\mathbf{x}$ will contain only entries corresponding to the factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$. The product $\mathbf{y} = \mathbf{H}\mathbf{x}$ will be computed in the same way as in the previous cases. In each iteration, only the factor matrices would be updated.

Authorized licensed use limited to: UTIA. Downloaded on April 09,2021 at 08:57:56 UTC from IEEE Xplore. Restrictions apply.

D. Linear Transformation of the Factor Matrices

In the previous sections, we have presented an implementation of the KLM algorithm for the Tucker decomposition, where the matrix $\mathbf{H}$ has a special structure. Assume now that parameter vector $\boldsymbol{\theta}$ can be obtained as a linear transformation of a new parameter $\boldsymbol{\vartheta}$ of a lower dimension,

$$\boldsymbol{\theta} = [\text{vec}(\mathcal{K}); \text{vec}(\mathbf{A}); \text{vec}(\mathbf{B}); \text{vec}(\mathbf{C})] = \mathbf{X}\boldsymbol{\vartheta}. \tag{49}$$

where $\mathbf{X}$ is a fixed tall matrix, and $\boldsymbol{\vartheta}$ is the new vector parameter. For example, we can facilitate symmetric or partially symmetric decomposition, say $\mathbf{A} = \mathbf{B} = \mathbf{C}$ or $\mathbf{A} = \mathbf{B}$. In the former case, we can define $\boldsymbol{\vartheta} = [\text{vec}(\mathcal{K}); \text{vec}(\mathbf{A})]$. Another example is enforcing a Toeplitz structure to some or all factor matrices or to a core tensor $\mathcal{K}$. In this way, it is possible, e.g., to implement the low-rank tensor deconvolution [31], PARallel FACtor with LINear Depedences (PARALIND) [33], or CANonical DEcomposition with LINear Constraints (CANDELINC) [34]. A similar technique is used in [10] and in Tensorlab. There are many possibilities, and all of them differ in the matrix $\mathbf{X}$. Note that the former model with some core tensor elements fixed to zero is a special case of the linear transformation considered in this subsection.

We now consider a Levenberg-Marquardt algorithm for the new parameter $\boldsymbol{\vartheta}$. The corresponding Jacobi matrix would be

$$\mathbf{J}_\vartheta = \frac{\partial \text{vec}\,\mathcal{T}}{\partial \boldsymbol{\vartheta}} = \frac{\partial \text{vec}\,\mathcal{T}}{\partial \boldsymbol{\theta}} \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\vartheta}} = \mathbf{J}\mathbf{X} \tag{50}$$

and the corresponding approximate Hessian that reads

$$\mathbf{H}_\vartheta = \mathbf{J}_\vartheta^T \mathbf{J}_\vartheta = \mathbf{X}^T \mathbf{J}^T \mathbf{J}\mathbf{X} = \mathbf{X}^T \mathbf{H}\mathbf{X} \tag{51}$$

in the unweighted case and

$$\mathbf{H}_\vartheta = \mathbf{J}_\vartheta^T \mathbf{W}\mathbf{J}_\vartheta = \mathbf{X}^T \mathbf{J}^T \mathbf{W}\mathbf{J}\mathbf{X} = \mathbf{X}^T \mathbf{H}\mathbf{X} \tag{52}$$

in the weighted case. The Krylov subspace variant of the LM method requires computing the products

$$\mathbf{H}_\vartheta \mathbf{x} = \mathbf{X}^T \mathbf{H}\mathbf{X}\mathbf{x} = \mathbf{X}^T \mathbf{H}(\mathbf{X}\mathbf{x}). \tag{53}$$

We can compute this product $\mathbf{y} = \mathbf{H}_\vartheta \mathbf{x}$ through the product of $\mathbf{H}$ with the vector $(\mathbf{X}\mathbf{x})$ via the algorithm proposed in the previous section. Note that the computation remains efficient, because evaluation of the matrix $\mathbf{H}$ is not needed.

E. Decompositions With Limited Sensitivity

Consider now a smooth constraint $c(\boldsymbol{\theta}) = 0$, e.g., the constraint on sensitivity. The LM algorithm can be modified as follows,

$$\boldsymbol{\theta}_1' = \boldsymbol{\theta}_0 - (\mathbf{H} + \mu\mathbf{I})^{-1}\mathbf{g} + \frac{\mathbf{u}_0^T (\mathbf{H} + \mu\mathbf{I})^{-1}\mathbf{g}}{\mathbf{u}_0^T (\mathbf{H} + \mu\mathbf{I})^{-1}\mathbf{u}_0} (\mathbf{H} + \mu\mathbf{I})^{-1}\mathbf{u}_0. \tag{54}$$

where

$$\mathbf{u}_0 = \left. \frac{\partial c(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}. \tag{55}$$

$\boldsymbol{\theta}_1$ obeying the constraint $c(\boldsymbol{\theta}_1) = 0$ is obtained from $\boldsymbol{\theta}_1'$ by appropriate scale change. See [11] for more details.

We have found empirically that for computing $(\mathbf{H} + \mu\mathbf{I})^{-1}\mathbf{u}_0$ it is convenient to generate another Krylov subspace,

$$[\mathbf{u}_0, \mathbf{H}\mathbf{u}_0, \mathbf{H}^2\mathbf{u}_0, \ldots, \mathbf{H}^{M-1}\mathbf{u}_0]$$

and compute its orthogonal basis $\mathbf{U}_u$. Then, we have

$$(\mathbf{H} + \mu\mathbf{I})^{-1}\mathbf{u}_0 \approx \frac{1}{\mu}\mathbf{u}_0 - \frac{1}{\mu}\mathbf{U}_u \left(\mu\mathbf{Q}_u^{-1} + \mathbf{U}_u^T\mathbf{U}_u\right)^{-1} \left(\mathbf{U}_u^T\mathbf{u}_0\right) \tag{56}$$

cf. (17), where

$$\mathbf{Q}_u = \mathbf{U}_u^T \mathbf{H}\mathbf{U}_u. \tag{57}$$

It appears that this procedure leads to a more accurate approximation than using the former Krylov subspace (18), or even (18) with increased dimension M. Investigation in this direction exceeds the scope of this paper.

The vector $\mathbf{u}_0$ is the gradient of the constraint in the latest value of $\boldsymbol{\theta}$. If the constraint is the sensitivity, $\mathbf{u}_0$ is composed of four parts,

$$\mathbf{u}_0 = [\mathbf{u}_K(\mathcal{L}); \mathbf{u}_A; \mathbf{u}_B; \mathbf{u}_C] \tag{58}$$

where

$$\begin{aligned} \mathbf{u}_K = {} & I_1 \text{vec}\,[[\mathcal{K}, \mathbf{I}_{R_1}, \mathbf{B}^T\mathbf{B}, \mathbf{C}^T\mathbf{C}]] \\ & + I_2 \text{vec}\,[[\mathcal{K}, \mathbf{A}^T\mathbf{A}, \mathbf{I}_{R_2}, \mathbf{C}^T\mathbf{C}]] \\ & + I_3 \text{vec}\,[[\mathcal{K}, \mathbf{A}^T\mathbf{A}, \mathbf{B}^T\mathbf{B}, \mathbf{I}_{R_3}]] \end{aligned} \tag{59}$$

and

$$\begin{aligned} \mathbf{u}_A = {} & \text{vec}\,\{\mathbf{A} \star (\mathbf{1}_{I_1}\mathbf{1}_{I_2 I_3}^T [[\mathcal{L}, \mathbf{I}_{R_1}, \mathbf{B} \star \mathbf{B}, \mathbf{C} \star \mathbf{C}]]_{(1)}^T)\} \\ & + I_2 \text{vec}\,\{\mathbf{A}[[\mathcal{K}, \mathbf{I}_{R_1}, \mathbf{I}_{R_2}, \mathbf{C}]]_{(1)} [[\mathcal{K}, \mathbf{I}_{R_1}, \mathbf{I}_{R_2}, \mathbf{C}]]_{(1)}^T\} \\ & + I_3 \text{vec}\,\{\mathbf{A}[[\mathcal{K}, \mathbf{I}_{R_1}, \mathbf{B}, \mathbf{I}_{R_3}]]_{(1)} [[\mathcal{K}, \mathbf{I}_{R_1}, \mathbf{B}, \mathbf{I}_{R_3}]]_{(1)}^T\} \\ \mathbf{u}_B = {} & \text{vec}\,\{\mathbf{B} \star (\mathbf{1}_{I_2}\mathbf{1}_{I_1 I_3}^T [[\mathcal{L}, \mathbf{A} \star \mathbf{A}, \mathbf{I}_{R_2}, \mathbf{C} \star \mathbf{C}]]_{(2)}^T)\} \\ & + I_1 \text{vec}\,\{\mathbf{B}[[\mathcal{K}, \mathbf{I}_{R_1}, \mathbf{I}_{R_2}, \mathbf{C}]]_{(2)} [[\mathcal{K}, \mathbf{I}_{R_1}, \mathbf{I}_{R_2}, \mathbf{C}]]_{(2)}^T\} \\ & + I_3 \text{vec}\,\{\mathbf{B}[[\mathcal{K}, \mathbf{A}, \mathbf{I}_{R_2}, \mathbf{I}_{R_3}]]_{(2)} [[\mathcal{K}, \mathbf{A}, \mathbf{I}_{R_2}, \mathbf{I}_{R_3}]]_{(2)} \\ \mathbf{u}_C = {} & \text{vec}\,\{\mathbf{C} \star (\mathbf{1}_{I_2}\mathbf{1}_{I_1 I_3}^T [[\mathcal{L}, \mathbf{A} \star \mathbf{A}, \mathbf{B} \star \mathbf{B}, \mathbf{I}_{R_3}]]_{(3)}^T)\} \\ & + I_1 \text{vec}\,\{\mathbf{C}[[\mathcal{K}, \mathbf{I}_{R_1}, \mathbf{B}, \mathbf{I}_{R_3}]]_{(3)} [[\mathcal{K}, \mathbf{I}_{R_1}, \mathbf{B}, \mathbf{I}_{R_3}]]_{(3)}^T\} \\ & + I_2 \text{vec}\,\{\mathbf{C}[[\mathcal{K}, \mathbf{A}, \mathbf{I}_{R_2}, \mathbf{I}_{R_3}]]_{(3)} [[\mathcal{K}, \mathbf{A}, \mathbf{I}_{R_2}, \mathbf{I}_{R_3}]]_{(3)}\}. \end{aligned}$$

F. Nonnegativity Constraints

In many applications, there is a need to impose nonnegativity constraint on some or all parameters of the model (core tensor and factor matrices). This can be done through a variant of the Alternating Direction of Multipliers (ADMM) [17] called Krylov-ADMM. In short, the procedure applies the fast computation of the term $(\mathbf{H} + \rho\mathbf{I})^{-1}\mathbf{g}$, described in the previous sections, in the ADMM procedure. Details can be found in Supplementary materials to this paper.

VIII. Simulations

Example 1 [Role of Parameter M] The influence of parameter M on the convergence of KLM will be studied in the case of CP decomposition of the matrix multiplicative tensor introduced in Section III. In particular, we take the case of the one corresponding to the multiplication of two matrices 4×4. The tensor has the size $16 \times 16 \times 16$. The distribution of ones and zeros in the tensor is plotted in Fig. 3. Actually, we show the tensor reshaped

Authorized licensed use limited to: UTIA. Downloaded on April 09,2021 at 08:57:56 UTC from IEEE Xplore. Restrictions apply.

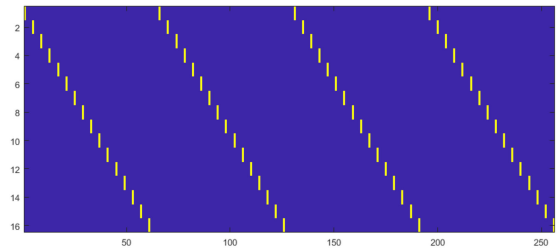

Fig. 3. Matricized tensor $\mathcal{M}$ from Example 1. Blue means zeros and yellow marks denote ones.

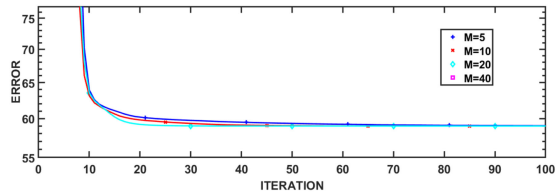

Fig. 4. Median learning curve of KLM with rank $R = 5$ and various M.

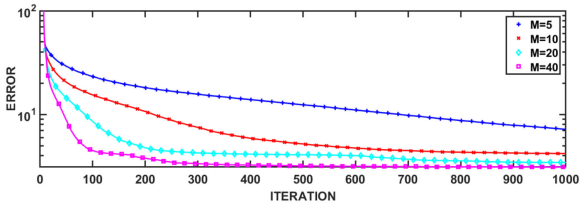

Fig. 5. Median learning curve of KLM with rank $R = 49$ and various M.

into a matrix of the size 16×256. The tensor has 64 ones and its rank is at most 49 [25].

First, we approximate the tensor by another one of rank $R = 5$ using the KLM algorithm with $M = 5, 10, 20$ and 40. In Fig. 4 we plot the median learning curve (error vs. iteration number) of 500 independent runs of the algorithm with random initialization and various M. We can see that the algorithm converges quickly regardless of the value of M. The conclusion is that if the rank is small, then small M suffices.

The influence of the parameter M becomes apparent at higher ranks, see Fig. 5 for rank $R = 49$. Note that the ideal fitting error should be zero, but this case never happened in the 500 trials. The minimum error (the squared Frobenius norm of the error tensor) was 1. Similar results (not shown here) were obtained by KLM with the sensitivity constraint set to 1200. In that case, the minimum fitting error was smaller than 1 in some cases.

Example 2 [Block-term decomposition] In this example, we generate the tensor from its block-diagonal core tensor $\mathcal{K}$ of the size $15 \times 15 \times 15$ with three blocks of the size $5 \times 5 \times 5$ on its main diagonal, at random, having i.i.d. Gaussian random entries with $N(0, 1)$ distribution. The factor matrices $\mathbf{A}, \mathbf{B},$ and $\mathbf{C}$ have the size 12×15 and the tensor $\mathcal{T} = [[\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{C}]]$ has the size $12 \times 12 \times 12$. It means that $\mathcal{T}$ is smaller in size than the core tensor. It has $12^3 = 1728$ elements. The number of the model parameters is $3 \times 5^3 + 3 \times 12 \times 15 = 915$. For simplicity, there was no additive noise.

We compare the performance of three algorithms: KLM with $M = 30$, KLM with bounded sensitivity and $M = 30$, and the NLS algorithm (Structured Data Fusion - Nonlinear Least Squares) of Tensorlab [9]. The bound on the sensitivity for the second algorithm was set to the sensitivity of the original model. The week point of the algorithm is that a qualified guess on the desired sensitivity is needed. The algorithms were initialized by random factor matrices and random core tensor in 230 independent trials. We let each of them perform 500 iterations.

TABLE I
COMPARISON OF THREE BTD ALGORITHMS FROM THE EXAMPLE WITH SYNTHETIC TENSORS

	KLM	KLM-BND	NLS
ERROR	6.2	3.9	6.5
# SUCCESS. RUNS	27 (12%)	89 (39%)	27 (12%)
TIME [s]	22.2	43.9	238.6
SENSITIVITY	$3.56 \cdot 10^7$	$0.25 \cdot 10^7$	$3.49 \cdot 10^7$

Theoretically, since there is no noise, the fitting error should converge to zero for all three algorithms. We have observed that this is not always the case. The algorithms differ in finding the right global minimum of the cost function among many other false minima. Also, we have limited the number of iterations to 500, so that the achieved minimum fitting error is not always close to zero.

In Table I, we present the following characteristics: (1) median fitting error per tensor element, i.e., the median fitting error divided by the number of the tensor elements, (2) number of cases where the right minimum of the cost function was achieved, numerically when the final fitting error was smaller than 10^{-3}, (3) computational time in seconds, and (4) median sensitivity of the estimates.

We can see that KLM and NLS have nearly the same median fitting error and the same number of successful runs and produce estimates with roughly the same sensitivity, although the sensitivity of the original model was significantly smaller. With the bounded sensitivity, KLM provided estimates with desired sensitivity and has about a $3\times$ larger probability of converging to the right local minimum, which is cca 39%. The computational time of the bounded KLM is about twice as long as the time of the unbounded KLM. NLS is the slowest algorithm in this example. Indeed, the probability of success could be increased by using multiple random initializations.

In Fig. 6, we present 25 learning curves for all three algorithms as illustration, and the median of the learning curves. The error is measured as the squared Frobenius norm of the error tensor.

The experiment was repeated once again with the difference that the bounded KLM was applied differently. It started the decomposition with low initial sensitivity, and after every 30 iterations, if it had not yet converged, the bound on sensitivity was increased $1.5\times$. In this way, we achieved the convergence to the desired global minimum in all the simulation trials. 300 iterations were sufficient in all cases.

The results were summarized in two scatter plots in Fig. 7. The former diagrams show the achieved fitting error as a function of the sensitivity of the estimated models. We can see that in most trials, KLM and NLS produce results with high fitting error and high sensitivity, unlike the bounded KLM. The latter diagram shows the output sensitivity as a function of the sensitivity of the original model. We note that in all trials, successful or not, KLM and NLS produce estimates with significantly higher sensitivities than necessary.

Example 3 [Decomposition of convolutional kernels in Alexnet CNN] In this example, we tested the BTD algorithms on real-world data. We used four convolutional kernels in the well-known convolutional neural network Alex-net [32]. These

Authorized licensed use limited to: UTIA. Downloaded on April 09,2021 at 08:57:56 UTC from IEEE Xplore. Restrictions apply.

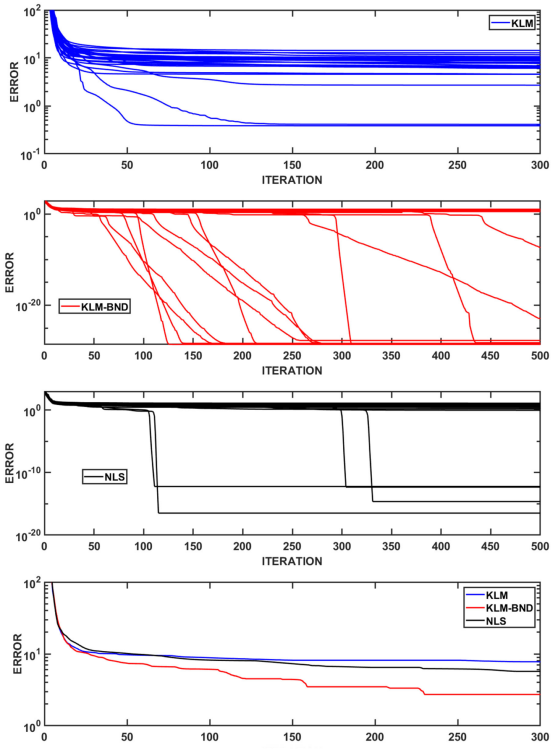

Fig. 6. Example of 25 learning curves of KLM, KLM with limited sensitivity, and NSF, and medians of the learning curves.

TABLE II
COMPARISON OF KLM-BND AND NLS ALGORITHMS IN THE EXAMPLE WITH REAL-WORLD TENSORS

Layer no./ size	Algorithm	Sensitivity	Relative fitting error
2	KLM-bnd	6.5307e+06	0.75058
$25 \times 48 \times 256$	NLS	1.8305e+12	0.75302
3	KLM-bnd	1.7378e+08	0.85923
$9 \times 256 \times 384$	NLS	1.0404e+12	0.85952
4	KLM-bnd	1.3588e+08	0.90215
$9 \times 192 \times 384$	NLS	1.055e+11	0.90224
5	KLM-bnd	2.1769e+07	0.85361
$9 \times 192 \times 256$	NLS	6.2318e+10	0.85397

tensors have the order 4, but were reshaped to order-3 tensors by folding their first two dimensions. It is known that most convolutional neural networks are overparameterized and exhibit a high computational cost mainly due to the huge number of parameters in the convolutional and fully connected layers. By factorizing the convolutional kernels in low-rank tensor formats, the convolutional layers can be replaced by a sequence of new layers with much smaller kernel sizes. This reduces the number of parameters and computational cost in the original CNN model. In [16], we have shown that CPD with sensitivity control can significantly improve compression of widely used CNN including ResNet, VGG, over ordinary CPD methods due to severe degeneracy of the decomposition results. By keeping the decomposition at low sensitivity, the compressed CNNs retain their original accuracy (with a minor loss). This example demonstrates an application of the `BTD` with sensitivity control in compression of CNN.

We decomposed four tensors whose sizes are given in Table II by `BTD` models with 15 blocks of size $2 \times 2 \times 2$, so that the core

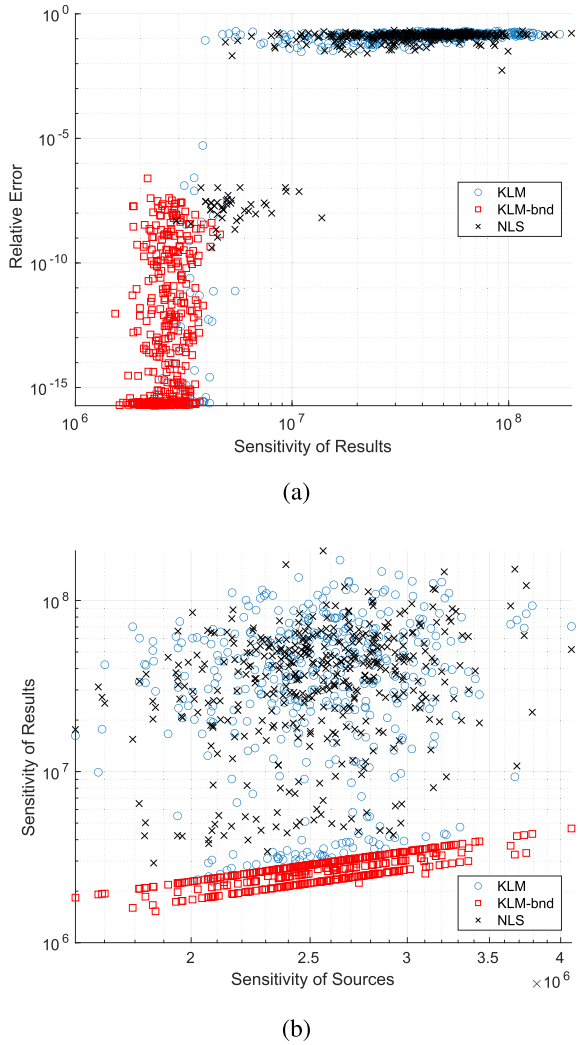

Fig. 7. (a) Fitting errors versus the model sensitivity and, (b) the output sensitivity versus the initial sensitivity.

tensor had the size $30 \times 30 \times 30$. The initial decomposition of the tensor was obtained as follows. The factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ were taken at random with $N(0,1)$-distributed elements. The initial core tensors were obtained by the least-squares update rule. The initial sensitivity bound for the algorithm `KLM-bnd` was taken as the sensitivity of the initial model. After every 30 iterations of the algorithm, the sensitivity bound is increased $1.5\times$ and the algorithm is halted if the fitting error is not reduced anymore. The performance is compared to that of `NLS` of Tensorlab, with 500 iterations. Table II shows the outcome, sensitivity, and fitting error. We can see that both algorithms achieve approximately the same fitting error, but our algorithm provides the estimate with a much smaller sensitivity measure. This opens a new application of sensitivity bounded `BTD` in compression of CNNs.

Example 4 [Tensor chain modeling]

We demonstrate the proposed KLM algorithm for TC decomposition of synthetic tensor of size $7 \times 7 \times 7$ and rank $R_n = 3$ for all n. The factor matrices, $\mathbf{U}_n$, are of size 7×9. Note that for this case, because the factor matrices have more columns than rows, algorithms based on SVD are not applicable, whereas the TC-ALS [5] does not work efficiently. Fig. 8 compares

Authorized licensed use limited to: UTIA. Downloaded on April 09,2021 at 08:57:56 UTC from IEEE Xplore. Restrictions apply.

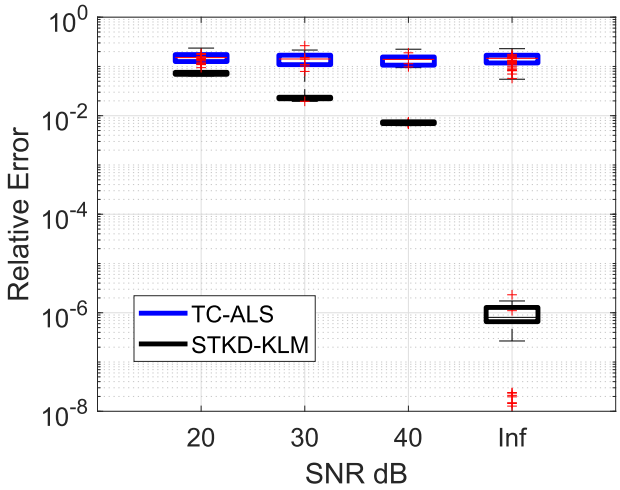

Fig. 8. Comparison of performance of KLM for structured TKD and TC-ALS.

TABLE III
PERFORMANCE COMPARISON FOR THREE MODELS, TKD, BTD AND BTD WITH OVERLAPPING CORES

Model	ACC	Mutual Information	Relative Error
TKD-(6,6,6)	0.87	0.45	0.7583
BTD-(3-3)	0.94	0.70	0.7727
OverBTD-(5-(4)-5)	0.98	0.89	0.7594

relative errors obtained using the two algorithms, TC-ALS and STKD-KLM with norm correction for different noise levels, SNR $= 20$, 30, 40 dB, and a noise-free case. The results were averaged over 100 independent runs. For all the test cases, the TC decomposition achieved through STKD outperformed the TC-ALS. The proposed method can find almost the exact decomposition for the noise-free case.

Example 5 [TKD vs. Overlapping BTD for clustering of handwritten digits].

In this example, we illustrate an application of the overlapping BTD for clustering of handwritten digit images in the MNIST dataset. From 100 images for each digit 5 and 6, which are of size 28×28, we computed their Gabor features with 8 orientations and 4 scales. The Gabor images are vectorized and concatenated into an order-3 tensor of size $784 \times 32 \times 200$. We applied TKD with multilinear rank-(6, 6, 6), BTD with two core tensors of size $3 \times 3 \times 3$ and the overlapping BTD with two core tensors of size $5 \times 5 \times 5$ sharing a term of size $4 \times 4 \times 4$. For all three decomposition models, the factor matrices were of the same size and comprised 6 columns. The difference between them is due to the structured of the core tensors. For Tucker decomposition, we used the HOOI algorithm [35], NLS for BTD [10], and KLM for overlapping BTD.

Six compressed features in the third-factor matrices were used to cluster the digits using the K-means algorithm. Table III compares the performance of the three models. Obviously, the TKD with multilinear rank-(6, 6, 6) explained the data better than the two other block term models. However, TKD does not achieve the best clustering accuracy.

The BTD-(3-3) with two smaller core tensors obtained an accuracy of 94%; the two subspaces can discriminate the two classes of digits.

The BTD with overlapping core tensors can be considered a decomposition that seeks joint subspaces associated with the overlapping part and individual subspace for each class. This model achieved the best accuracy of 98% with the highest mutual information of 0.89.

An extension of this example can be found in the supplementary material to this paper.

IX. CONCLUSIONS

We presented novel algorithms for structured or constrained Tucker tensor decomposition and their application in block term decomposition, tensor chain modeling, classification of handwritten digits, and the compression of convolutional layers in neural networks.

Matlab codes related to the KLM algorithm are available on the Internet at https://github.com/Tichavsky/tensor-decomposition.

REFERENCES

[1] 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 Trans. Signal Process.*, vol. 65, no. 13, pp. 3551–3582, Jul. 2017.

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

[3] I. V. Oseledets and E. E. Tyrtyshnikov, "Breaking the curse of dimensionality, or how to use SVD in many dimensions," *SIAM J. Sci. Comput.*, vol. 31, no. 5, pp. 3744–3759, 2009.

[4] Y. Zniyed, R. Boyer, A. L. F. de Almeida, and G. Favier, "High-order tensor estimation via trains of coupled third-order CP and tucker decompositions," *Linear Algebra Appl.*, vol. 588, 2020, pp. 304–337.

[5] M. Espig, W. Hackbusch, S. Handschuh, and R. Schneider, "Optimization problems in contracted tensor networks," *Comput. Vis. Sci.*, vol. 14, no. 6, pp. 271–285, 2011.

[6] B. N. Khoromskij, "O(d log n)-quantics approximation of n-d tensors in high-dimensional numerical modeling," *Constructive Approximation*, vol. 34, no. 2, pp. 257–280, 2011.

[7] L. De Lathauwer, "Decompositions of a higher-order tensor in block terms-part II: Definitions and uniqueness," *SIAM J. Matrix Anal. Appl.*, vol. 30, no 3, pp. 1033–1066.

[8] C. Chatzichristos, E. Kofidis, M. Morante, S. Theodoridis, "Blind fMRI source unmixing via higher-order tensor decompositions," *J. Neurosci. Methods*, vol. 315, pp. 17–47, Mar. 2019.

[9] N. Vervliet, O. Debals *et al.*, "Tensorlab 3.0," [Online]. Available: https://www.tensorlab.net, Mar. 2016

[10] L. Sorber, M. Van Barel and L. De Lathauwer, "Structured data fusion," *IEEE J. Sel. Top. Signal Process.*, vol. 9, no. 4, pp. 586–600, Jun. 2015.

[11] P. Tichavský, A. H. Phan, and A. Cichocki, "Sensitivity in tensor decomposition" *IEEE Signal Process. Lett.*, vol. 26, no. 11, pp. 1653–1657, Nov. 2019.

[12] P. Tichavský, A. H. Phan, and A. Cichocki, "Weighted Krylov-Levenberg-Marquardt method for canonical polyadic tensor decomposition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Barcelona, Spain, pp. 3917–3921, 2020.

[13] A. N. Krylov, "On the numerical solution of equation by which are determined in technical problems the frequencies of small vibrations of material systems". *Izvestija SSSR (News Academy Sciences USSR)*, Otdel. mat. i estest. nauk, 1931, VII, Nr. 4, 491–539 (*in Russian*).

[14] J. Liesen and Z. Strakos, *Krylov Subspace Methods: Principles and Analysis*, Oxford, U.K.: Oxford Science Publications, 2012.

[15] N. Vervliet and L. De Lathauwer, "Numerical optimization based algorithms for data fusion," in *Data Fusion Methodology and Applications*, vol. 31, M. Cocchi, Ed., Amsterdam, Netherlands: Elsevier, 2019.

[16] A.-H. Phan *et al.*, "Stable low-rank tensor decomposition for compression of convolutional neural network," in *Proc. 16th Eur. Conf. Comput. Vis. - ECCV 2020*, Glasgow, 2020, Springer International Publishing, LNCS 12374, 1931, pp. 522–539.

[17] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.* vol. 3, no. 1, pp. 1–122, Jan. 2011.

Authorized licensed use limited to: UTIA. Downloaded on April 09,2021 at 08:57:56 UTC from IEEE Xplore. Restrictions apply.

[18] A. P. Liavas, G. Kostoulas, G. Lourakis, K. Huang, and N. D. Sidiropoulos, "Nesterov-based alternating optimization for nonnegative tensor factorization: Algorithm and parallel implementation," *IEEE Trans. Signal Process.*, vol. 66, no. 3, pp. 5052–5065, Feb. 2018.
[19] A. P. Liavas and N. D. Sidiropoulos, "Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers. "*IEEE Trans. Signal Process.*, vol. 63, no. 20, pp. 5450–5463, Oct. 2015.
[20] K. Huang and X. Fu, "Low-complexity proximal Gauss-Newton algorithm for nonnegative matrix factorization," in *Proc. IEEE Global Conf. Signal Inf. Process.* (GlobalSIP), Ottawa, ON, Canada, 2019, pp. 1–5.
[21] A.-H Phan, P. Tichavský and A. Cichocki, "Low complexity damped Gauss-Newton algorithms for farallel factor analysis," *SIAM J. Matrix Anal. Appl.*, vol. 34, no. 1, pp. 126–147, Jan. 2013.
[22] P. Tichavský, A. H. Phan, and A. Cichocki, "A further improvement of a fast damped Gauss-Newton algorithm for CANDECOMP-PARAFAC tensor decomposition," in *Proc. Int. Conf. Acoust., Speech Signal Process.*, Vancouver, Canada, May 2013, pp. 5964–5968.
[23] J. M. Landsberg, *Tensors: Geometry and Applications*, Graduate Studies in Mathematics, vol. 128, Amer. Math. Soc., Providence, RI, USA, 2012.
[24] S. Handschuh, Numerical methods in tensor networks, Ph.D. thesis, Fac. Math. Inform., Univ. Leipzig, Germany, Leipzig, Germany, 2015.
[25] P. Tichavsky, A.-H. Phan, and A. Cichocki, "Numerical CP decomposition of some difficult tensors," *J. Comput. Appl. Math.*, vol. 317, pp. 362–370, 2017.
[26] A. H. Phan, P. Tichavský, and A. Cichocki, "Error preserving correction: A method for CP decomposition at a target error bound," *IEEE Trans. Signal Process.*, vol. 67, no. 5, pp. 1175–1190, Mar. 2019.
[27] Q. Shi, H. Lu, and Y.-M. Cheung, "Rank-one matrix completion with automatic rank estimation via L1-norm regularization," *IEEE Trans. Neural Netw. Learn. Syst. (TNNLS)*, Vol. 29, no.10, pp. 4744–4757, Oct. 2018.
[28] Q. Shi, Y.-M Cheung, and H. Lu, "Feature extraction for incomplete data via low-rank tensor decomposition with feature regularization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 6, pp. 1803–1817, Jun. 2019.
[29] K. Madsen, H. B. Nielsen, O. Tingleff, "Methods for nonlinear least squares problems, *second ed.*," Depart. Math. Model., Tech. Univ. Denmark, Lyngby, Denmark, 2004.
[30] J. Nocedal, S. J. Wright, *Numerical Optimization*, Springer 2006.
[31] A. H. Phan, P. Tichavský, and A. Cichocki, "Low rank tensor deconvolution," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2015, pp. 2169–2173.
[32] A. Krizhevsky, S. Ilya and H. Geoffrey E., "ImageNet classification with deep convolutional neural networks," *Adv. Neural Inf. Process. Syst.* 25, pp. 1097–1105, 2012.
[33] R. Bro, R. Harshman, N. D. Sidiropoulos, and M. E. Lundy, "Modeling multi-way data with linearly dependent loadings," *J. Chemometrics*, vol 26, no. 7-8, 2009.
[34] J. D. Carroll, S. Pruzansky, and J. B. Kruskal, "Candelinc: A. general approach to multidimensional analysis of many-way arrays with linear constraints on parameters," *Psychometrika*, vol. 45, pp. 3–24, 1980.
[35] L. De Lathauwer, B. De Moor, and J. Vandewalle, "*On the best rank-1 and rank-(R1,R2,. .., RN) approximation of higher-order tensors*" *SIAM J. Matrix Anal. Appl.*, vol. 21, pp. 1324–1342, Mar. 2000.

Petr Tichavský (Senior Member, IEEE) received the Ph.D. degree in theoretical cybernetics and the Research Professor degree from the Czechoslovak Academy of Sciences, Prague, Czech Republic, in 1992 and 2017, respectively. He is currently with the Institute of Information Theory and Automation, Czech Academy of Sciences. He is the author or coauthor of research articles in his research areas, which include sinusoidal frequency/frequency-rate estimation, adaptive filtering and tracking of time-varying signal parameters, algorithm-independent bounds on achievable performance, sensor array processing, independent component analysis, blind source separation, and tensor decompositions. From 2008 to 2011 and since 2016, he has been a Member of the IEEE SPS Committee Signal Processing Theory and Methods. He was also the General Co-Chair of the 36th IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP 2011 in Prague. From 2002 to 2004, he was an Associate Editor for the IEEE SIGNAL PROCESSING LETTERS.From 2005 to 2009 and from 2011 to 2016, he was an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING.

Anh-Huy Phan received the master's degree from the Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam, in 2005 and the Ph.D. degree from the Kyushu Institute of Technology, Kitakyushu, Japan, in 2011. From October 2011 to March 2018, he was a Research Scientist with the Laboratory for Advanced Brain Signal Processing, Brain Science Institute (BSI), RIKEN, Wako, Japan, and from April 2012 to April 2015, a Visiting Research Scientist with the TOYOTA Collaboration Center, BSI-RIKEN. He was appointed as an Assistant Professor in May 2018 and since June 2020, he has been an Associate Professor with the Center for Computational and Data-Intensive Science and Engineering, Skolkovo Institute of Science and Technology (Skoltech), Moscow, Russia. He was a Visiting Scientist with Brain Science Institute, RIKEN and a Visiting Associate Professor with the Tokyo University of Agriculture and Technology, Fuchu, Japan. He is the Author of three monographs. His research interests include multilinear algebra, tensor computation, tensor networks, nonlinear system, blind source separation, and brain-computer interface. He was the recipient of the best paper awards for articles in the IEEE SPM (2018) and the ICONIP (2016), and the Outstanding Reviewer Award for maintaining the prestige of ICASSP 2019.

Andrzej Cichocki (Fellow, IEEE) received the M.Sc. (with honors), Ph.D., and Dr.Sc. (habilitation) degrees in electrical engineering from the Warsaw University of Technology, Warsaw, Poland. For several years he was an Alexander-von-Humboldt Research Fellow and a Guest Professor with University Erlangen-Nurnberg, Erlangen, Germany. From 1995 to 2018, he was the Team Leader and the Head of the laboratory for Advanced Brain Signal Processing, RIKEN Brain Science Institute, Wako, Japan. He is currently a Professor with the Skolkovo Institute of Science and Technology, Moscow, Russia, and also a Visiting/Adjunct Professor with the Tokyo University of Agriculture and Technology, Fuchu, Japan, Hangzhou Dianzi University, Hangzhou, China, Nicolaus Copernicus University, Toruń, Poland, and the Institute of Systems Research, Polish Academy of Science, Warsaw, Poland. He is the author of more than 500 peer-review papers and six monographs in English (two of them translated to Chinese). His current research interests include deep learning, tensor decompositions, tensor networks for big data analytics, multiway blind source separation, and Brain Computer Interface and their biomedical applications. He was or currently is an Associate Editor for the IEEE TRANSACTIONS ON SIGNALS PROCESSING, the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, the IEEE TRANSACTIONS ON CYBERNETICS, and the *Journal of Neuroscience Methods.* He was the founding Editor-in-Chief of the *Computational Intelligence and Neuroscience* journal. His publications currently report more than 47,000 citations according to Google Scholar, with an h-index of 100. He is currently among three the most cited Polish computer scientists.

Authorized licensed use limited to: UTIA. Downloaded on April 09,2021 at 08:57:56 UTC from IEEE Xplore. Restrictions apply.