IEEE TRANSACTIONS ON
SIGNAL PROCESSING
A PUBLICATION OF THE IEEE SIGNAL PROCESSING SOCIETY
www.signalprocessingsociety.org
Indexed in PubMed® and MEDLINE®, products of the United States National Library of Medicine
PubMed MEDLINE

Tensor Decomposition for Signal Processing and Machine Learning

Nicholas D. Sidiropoulos, *Fellow, IEEE*, Lieven De Lathauwer, *Fellow, IEEE*, Xiao Fu, *Member, IEEE*, Kejun Huang, *Member, IEEE*, Evangelos E. Papalexakis, and Christos Faloutsos

***Abstract*—Tensors or *multi-way arrays* are functions of three or more indices $(i, j, k, \cdots)$ – similar to matrices (two-way arrays), which are functions of two indices (r, c) for (row,column). Tensors have a rich history, stretching over almost a century, and touching upon numerous disciplines; but they have only recently become ubiquitous in signal and data analytics at the confluence of signal processing, statistics, data mining and machine learning. This overview article aims to provide a good starting point for researchers and practitioners interested in learning about and working with tensors. As such, it focuses on fundamentals and motivation (using various application examples), aiming to strike an appropriate balance of breadth *and depth* that will enable someone having taken first graduate courses in matrix algebra and probability to get started doing research and/or developing tensor algorithms and software. Some background in applied optimization is useful but not strictly required. The material covered includes tensor rank and rank decomposition; basic tensor factorization models and their relationships and properties (including fairly good coverage of identifiability); broad coverage of algorithms ranging from alternating optimization to stochastic gradient; statistical performance analysis; and applications ranging from source separation to collaborative filtering, mixture and topic modeling, classification, and multilinear subspace learning.**

***Index Terms*—Tensor decomposition, tensor factorization, rank, canonical polyadic decomposition (CPD), parallel factor analysis (PARAFAC), Tucker model, higher-order singular value decomposition (HOSVD), multilinear singular value decomposition (MLSVD), uniqueness, NP-hard problems, alternating optimization, alternating direction method of multipliers, gradient descent, Gauss-Newton, stochastic gradient, Cramér-Rao bound, communications, source separation, harmonic retrieval, speech separation, collaborative filtering, mixture modeling, topic modeling, classification, subspace learning.**

Overview paper submitted to *IEEE Trans. on Sig. Proc.*, June 23, 2016; revised December 13, 2016; accepted Jan. 25, 2017.

N.D. Sidiropoulos, X. Fu, and K. Huang are with the ECE Department, University of Minnesota, Minneapolis, USA; e-mail: (nikos,xfu,huang663)@umn.edu. Supported in part by NSF IIS-1247632, IIS-1447788.

Lieven De Lathauwer is with KU Leuven, Belgium; e-mail: Lieven.DeLathauwer@kuleuven.be. Supported by (1) KU Leuven Research Council: CoE EF/05/006 Optimization in Engineering (OPTEC), C1 project C16/15/059-nD; (2) F.W.O.: project G.0830.14N, G.0881.14N; (3) Belgian Federal Science Policy Office: IUAP P7 (DYSCO II, Dynamical systems, control and optimization, 20122017); (4) EU: The research leading to these results has received funding from the European Research Council under the European Unions Seventh Framework Programme (FP7/2007-2013) / ERC Advanced Grant: BIOTENSORS (no. 339804). This paper reflects only the authors' views and the EU is not liable for any use that may be made of the contained information.

E.E. Papalexakis and C. Faloutsos are with the CS Department, Carnegie Mellon University, USA; e-mail (epapalex,christos)@cs.cmu.edu. Supported in part by NSF IIS-1247489.
Digital Object Identifier 10.1109/TSP.2017.2690524

I. Introduction

Tensors[1] (of order higher than two) are arrays indexed by three or more indices, say $(i, j, k, \cdots)$ – a generalization of matrices, which are indexed by two indices, say (r, c) for (row, column). Matrices are two-way arrays, and there are three- and higher-way arrays (or *higher-order*) tensors.

Tensor algebra has many similarities but also many striking differences with matrix algebra – e.g., low-rank tensor factorization is essentially unique under mild conditions; determining tensor rank is NP-hard, on the other hand, and the best low-rank approximation of a higher rank tensor may not even exist. Despite such apparent paradoxes and the learning curve needed to digest tensor algebra notation and data manipulation, tensors have already found many applications in signal processing (speech, audio, communications, radar, biomedical), machine learning (clustering, dimensionality reduction, latent factor models, subspace learning), and well beyond. Psychometrics (loosely defined as mathematical methods for the analysis of personality data) and later Chemometrics (likewise, for chemical data) have historically been two important application areas driving theoretical and algorithmic developments. Signal processing followed, in the 90's, but the real spark that popularized tensors came when the computer science community (notably those in machine learning, data mining, computing) discovered the power of tensor decompositions, roughly a decade ago [1]–[3]. There are nowadays many hundreds, perhaps thousands of papers published each year on tensor-related topics. Signal processing applications include, e.g., unsupervised separation of unknown mixtures of speech signals [4] and code-division communication signals without knowledge of their codes [5]; and emitter localization for radar, passive sensing, and communication applications [6], [7]. There are many more applications of tensor techniques that are not immediately recognized as such, e.g., the analytical constant modulus algorithm [8], [9]. Machine learning applications include face recognition, mining musical scores, and detecting cliques in social networks – see [10]–[12] and references therein. More recently, there has been considerable work on tensor decompositions for learning latent variable models, particularly topic models [13], and connections between orthogonal tensor decomposition and the method of moments for computing the Latent Dirichlet Allocation (LDA – a widely used topic model).

[1]The term has different meaning in Physics, however it has been widely adopted across various disciplines in recent years to refer to what was previously known as a *multi-way array*.

After two decades of research on tensor decompositions and applications, the senior co-authors still couldn't point their new graduate students to a single "point of entry" to begin research in this area. This article has been designed to address this need: to provide a fairly comprehensive *and* deep overview of tensor decompositions that will enable someone having taken first graduate courses in matrix algebra and probability to get started doing research and/or developing related algorithms and software. While no single reference fits this bill, there are several very worthy tutorials and overviews that offer different points of view in certain aspects, and we would like to acknowledge them here. Among them, the highly-cited and clearly-written tutorial [14] that appeared 7 years ago in *SIAM Review* is perhaps the one closest to this article. It covers the basic models and algorithms (as of that time) well, but it does not go deep into uniqueness, advanced algorithmic, or estimation-theoretic aspects. The target audience of [14] is applied mathematics (SIAM). The recent tutorial [11] offers an accessible introduction, with many figures that help ease the reader into three-way thinking. It covers most of the bases and includes many motivating applications, but it also covers a lot more beyond the basics and thus stays at a high level. The reader gets a good roadmap of the area, without delving into it enough to prepare for research. Another recent tutorial on tensors is [15], which adopts a more abstract point of view of tensors as mappings from a linear space to another, whose coordinates transform multilinearly under a change of bases. This article is more suited for people interested in tensors as a mathematical concept, rather than how to use tensors in science and engineering. It includes a nice review of tensor rank results and a brief account of uniqueness aspects, but nothing in the way of algorithms or tensor computations. An overview of tensor techniques for large-scale numerical computations is given in [16], [17], geared towards a scientific computing audience; see [18] for a more accessible introduction. A gentle introduction to tensor decompositions can be found in the highly cited Chemometrics tutorial [19] – a bit outdated but still useful for its clarity – and the more recent book [20]. Finally, [21] is an upcoming tutorial with emphasis on scalability and data fusion applications – it does not go deep into tensor rank, identifiability, decomposition under constraints, or statistical performance benchmarking.

None of the above offers a comprehensive overview that is sufficiently deep to allow one to appreciate the underlying mathematics, the rapidly expanding and diversifying toolbox of tensor decomposition algorithms, and the basic ways in which tensor decompositions are used in signal processing and machine learning – and they are quite different. Our aim in this paper is to give the reader a tour that goes 'under the hood' on the technical side, and, at the same time, serve as a bridge between the two areas. Whereas we cannot include detailed proofs of some of the deepest results, we do provide insightful derivations of simpler results and *sketch* the line of argument behind more general ones. For example, we include a one-page self-contained proof of Kruskal's condition when one factor matrix is full column rank, which illuminates the role of Kruskal-rank in proving uniqueness. We also 'translate' between the signal processing (SP) and machine learning (ML) points of view. In the context of the canonical polyadic decomposition (CPD), also known as parallel factor analysis (PARAFAC), SP researchers (and Chemists) typically focus on the columns of the factor matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and the associated rank-1 factors $\mathbf{a}_f \circledcirc \mathbf{b}_f \circledcirc \mathbf{c}_f$ of the decomposition (where $\circledcirc$ denotes the outer product, see section II-C), because they are interested in *separation*. ML researchers often focus on the rows of $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, because they think of them as parsimonious latent space representations. For a user $\times$ item $\times$ context ratings tensor, for example, a row of $\mathbf{A}$ is a representation of the corresponding user in latent space, and likewise a row of $\mathbf{B}$ ($\mathbf{C}$) is a representation of the corresponding item (context) in the same latent space. The inner product of these three vectors is used to predict that user's rating of the given item in the given context. This is one reason why ML researchers tend to use inner (instead of outer) product notation. SP researchers are interested in model identifiability because it guarantees separability; ML researchers are interested in identifiability to be able to interpret the dimensions of the latent space. In co-clustering applications, on the other hand, the rank-1 tensors $\mathbf{a}_f \circledcirc \mathbf{b}_f \circledcirc \mathbf{c}_f$ capture *latent concepts* that the analyst seeks to learn from the data (e.g., cliques of users buying certain types of items in certain contexts). SP researchers are trained to seek *optimal* solutions, which is conceivable for small to moderate data; they tend to use computationally heavier algorithms. ML researchers are nowadays trained to think about scalability from day one, and thus tend to choose much more lightweight algorithms to begin with. There are many differences, but also many similarities and opportunities for cross-fertilization. Being conversant in both communities allows us to bridge the ground between and help SP and ML researchers better understand each other.

A. Roadmap

The rest of this article is structured as follows. We begin with some matrix preliminaries, including matrix rank and low-rank approximation, and a review of some useful matrix products and their properties. We then move to rank and rank decomposition for tensors. We briefly review bounds on tensor rank, multilinear (mode-) ranks, and relationship between tensor rank and multilinear rank. We also explain the notions of typical, generic, and border rank, and discuss why low-rank tensor approximation may not be well-posed in general. Tensors can be viewed as data or as multi-linear operators, and while we are mostly concerned with the former viewpoint in this article, we also give a few important examples of the latter as well. Next, we provide a fairly comprehensive account of uniqueness of low-rank tensor decomposition. This is the most advantageous difference when one goes from matrices to tensors, and therefore understanding uniqueness is important in order to make the most out of the tensor toolbox. Our exposition includes two stepping-stone proofs: one based on eigendecomposition, the other bearing Kruskal's mark ("down-converted to baseband" in terms of difficulty). The Tucker model and multilinear SVD come next, along with a discussion of their properties and connections with rank decomposition. A thorough discussion of algorithmic aspects follows, including

a detailed discussion of how different types of constraints can be handled, how to exploit data sparsity, scalability, how to handle missing values, and different loss functions. In addition to basic alternating optimization strategies, a host of other solutions are reviewed, including gradient descent, line search, Gauss-Newton, alternating direction method of multipliers, and stochastic gradient approaches. The next topic is statistical performance analysis, focusing on the widely-used Cramér-Rao bound and its efficient numerical computation. This section contains novel results and derivations that are of interest well beyond our present context – e.g., can also be used to characterize estimation performance for a broad range of constrained matrix factorization problems. The final main section of the article presents motivating applications in signal processing (communication and speech signal separation, multidimensional harmonic retrieval) and machine learning (collaborative filtering, mixture and topic modeling, classification, and multilinear subspace learning). We conclude with some pointers to online resources (toolboxes, software, demos), conferences, and some historical notes.

II. Preliminaries

A. Rank and rank decomposition for matrices

Consider an $I \times J$ matrix $\mathbf{X}$, and let colrank$(\mathbf{X})$:= the number of linearly independent columns of $\mathbf{X}$, i.e., the dimension of the range space of $\mathbf{X}$, dim(range$(\mathbf{X})$). colrank$(\mathbf{X})$ is the minimum $k \in \mathbb{N}$ such that $\mathbf{X} = \mathbf{A}\mathbf{B}^T$, where $\mathbf{A}$ is an $I \times k$ basis of range$(\mathbf{X})$, and $\mathbf{B}^T$ is $k \times J$ and holds the corresponding coefficients. This is because if we can generate all columns of $\mathbf{X}$, by linearity we can generate anything in range$(\mathbf{X})$, and vice-versa. We can similarly define rowrank$(\mathbf{X})$:= the number of linearly independent rows of $\mathbf{X}$ = dim(range$(\mathbf{X}^T)$), which is the minimum $\ell \in \mathbb{N}$ such that $\mathbf{X}^T = \mathbf{B}\mathbf{A}^T \Longleftrightarrow \mathbf{X} = \mathbf{A}\mathbf{B}^T$, where $\mathbf{B}$ is $J \times \ell$ and $\mathbf{A}^T$ is $\ell \times I$. Noting that

$$\mathbf{X} = \mathbf{A}\mathbf{B}^T = \mathbf{A}(:,1)(\mathbf{B}(:,1))^T + \cdots + \mathbf{A}(:,\ell)(\mathbf{B}(:,\ell))^T,$$

where $\mathbf{A}(:,\ell)$ stands for the ℓ-th column of $\mathbf{A}$, we have

$$\mathbf{X} = \mathbf{a}_1\mathbf{b}_1^T + \cdots + \mathbf{a}_\ell\mathbf{b}_\ell^T,$$

where $\mathbf{A} = [\mathbf{a}_1, \cdots, \mathbf{a}_\ell]$ and $\mathbf{B} = [\mathbf{b}_1, \cdots, \mathbf{b}_\ell]$. It follows that colrank$(\mathbf{X})$ = rowrank$(\mathbf{X})$ = rank$(\mathbf{X})$, and rank$(\mathbf{X})$ = minimum m such that $\mathbf{X} = \sum_{n=1}^{m} \mathbf{a}_n\mathbf{b}_n^T$, so the three definitions actually coincide – but only in the matrix (two-way tensor) case, as we will see later. Note that, per the definition above, $\mathbf{a}\mathbf{b}^T$ is a rank-1 matrix that is 'simple' in the sense that every column (or row) is proportional to any other column (row, respectively). In this sense, rank can be thought of as a measure of complexity. Note also that rank$(\mathbf{X}) \leq \min(I, J)$, because obviously $\mathbf{X} = \mathbf{X}\mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

B. Low-rank matrix approximation

In practice $\mathbf{X}$ is usually full-rank, e.g., due to measurement noise, and we observe $\mathbf{X} = \mathbf{L} + \mathbf{N}$, where $\mathbf{L} = \mathbf{A}\mathbf{B}^T$ is low-rank and $\mathbf{N}$ represents noise and 'unmodeled dynamics'. If the elements of $\mathbf{N}$ are sampled from a jointly continuous distribution, then $\mathbf{N}$ will be full rank almost surely – for the determinant of any square submatrix of $\mathbf{N}$ is a polynomial in the matrix entries, and a polynomial that is nonzero at one point is nonzero at every point except for a set of measure zero. In such cases, we are interested in approximating $\mathbf{X}$ with a low-rank matrix, i.e., in

$$\min_{\mathbf{L} \mid \text{rank}(\mathbf{L})=\ell} ||\mathbf{X} - \mathbf{L}||_F^2 \Longleftrightarrow \min_{\mathbf{A}\in\mathbb{R}^{I\times\ell},\ \mathbf{B}\in\mathbb{R}^{J\times\ell}} ||\mathbf{X} - \mathbf{A}\mathbf{B}^T||_F^2.$$

The solution is provided by the truncated SVD of $\mathbf{X}$, i.e., with $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, set $\mathbf{A} = \mathbf{U}(:,1:\ell)\mathbf{\Sigma}(1:\ell,1:\ell)$, $\mathbf{B} = \mathbf{V}(:,1:\ell)$ or $\mathbf{L} = \mathbf{U}(:,1:\ell)\mathbf{\Sigma}(1:\ell,1:\ell)(\mathbf{V}(:,1:\ell))^T$, where $\mathbf{U}(:,1:\ell)$ denotes the matrix containing columns 1 to ℓ of $\mathbf{U}$. However, this factorization is non-unique because $\mathbf{A}\mathbf{B}^T = \mathbf{A}\mathbf{M}\mathbf{M}^{-1}\mathbf{B}^T = (\mathbf{A}\mathbf{M})(\mathbf{B}\mathbf{M}^{-T})^T$, for any nonsingular $\ell\times\ell$ matrix $\mathbf{M}$, where $\mathbf{M}^{-T} = (\mathbf{M}^{-1})^T$. In other words: the factorization of the approximation is highly non-unique (when $\ell = 1$, there is only scaling ambiguity, which is usually inconsequential). As a special case, when $\mathbf{X} = \mathbf{L}$ (noise-free) so rank$(\mathbf{X}) = \ell$, low-rank decomposition of $\mathbf{X}$ is non-unique.

C. Some useful products and their properties

In this section we review some useful matrix products and their properties, as they pertain to tensor computations.

Kronecker product: The Kronecker product of $\mathbf{A}$ $(I \times K)$ and $\mathbf{B}$ $(J \times L)$ is the $IJ \times KL$ matrix

$$\mathbf{A} \otimes \mathbf{B} := \begin{bmatrix} \mathbf{B}\mathbf{A}(1,1) & \mathbf{B}\mathbf{A}(1,2) & \cdots & \mathbf{B}\mathbf{A}(1,K) \\ \mathbf{B}\mathbf{A}(2,1) & \mathbf{B}\mathbf{A}(2,2) & \cdots & \mathbf{B}\mathbf{A}(2,K) \\ \vdots & \vdots & \cdots & \vdots \\ \mathbf{B}\mathbf{A}(I,1) & \mathbf{B}\mathbf{A}(I,2) & \cdots & \mathbf{B}\mathbf{A}(I,K) \end{bmatrix}$$

The Kronecker product has many useful properties. From its definition, it follows that $\mathbf{b}^T \otimes \mathbf{a} = \mathbf{a}\mathbf{b}^T$. For an $I \times J$ matrix $\mathbf{X}$, define

$$\text{vec}(\mathbf{X}) := \begin{bmatrix} \mathbf{X}(:,1) \\ \mathbf{X}(:,2) \\ \vdots \\ \mathbf{X}(:,J) \end{bmatrix},$$

i.e., the $IJ \times 1$ vector obtained by vertically stacking the columns of $\mathbf{X}$. By definition of vec$(\cdot)$ it follows that vec$(\mathbf{a}\mathbf{b}^T) = \mathbf{b} \otimes \mathbf{a}$.

Consider the product $\mathbf{A}\mathbf{M}\mathbf{B}^T$, where $\mathbf{A}$ is $I \times K$, $\mathbf{M}$ is $K \times L$, and $\mathbf{B}$ is $J \times L$. Note that

$$\begin{aligned} \mathbf{A}\mathbf{M}\mathbf{B}^T &= \left(\sum_{k=1}^{K} \mathbf{A}(:,k)\mathbf{M}(k,:)\right)\mathbf{B}^T \\ &= \sum_{k=1}^{K}\sum_{\ell=1}^{L} \mathbf{A}(:,k)\mathbf{M}(k,\ell)(\mathbf{B}(:,\ell))^T. \end{aligned}$$

Therefore, using vec$(\mathbf{a}\mathbf{b}^T) = \mathbf{b}\otimes\mathbf{a}$ and linearity of the vec$(\cdot)$ operator

$$\begin{aligned} \text{vec}\left(\mathbf{A}\mathbf{M}\mathbf{B}^T\right) &= \sum_{k=1}^{K}\sum_{\ell=1}^{L} \mathbf{M}(k,\ell)\mathbf{B}(:,\ell) \otimes \mathbf{A}(:,k) \\ &= (\mathbf{B} \otimes \mathbf{A})\,\text{vec}(\mathbf{M}). \end{aligned}$$

This is useful when dealing with linear least squares problems of the following form

$$\min_{\mathbf{M}} ||\mathbf{X} - \mathbf{A}\mathbf{M}\mathbf{B}^T||_F^2 \Longleftrightarrow \min_{\mathbf{m}} ||\text{vec}(\mathbf{X}) - (\mathbf{B} \otimes \mathbf{A})\mathbf{m}||_2^2,$$

where $\mathbf{m} := \text{vec}(\mathbf{M})$.

Khatri–Rao product: Another useful product is the Khatri–Rao (column-wise Kronecker) product of two matrices *with the same number of columns* (see [20, p. 14] for a generalization). That is, with $\mathbf{A} = [\mathbf{a}_1, \cdots, \mathbf{a}_\ell]$ and $\mathbf{B} = [\mathbf{b}_1, \cdots, \mathbf{b}_\ell]$, the Khatri–Rao product of $\mathbf{A}$ and $\mathbf{B}$ is $\mathbf{A} \odot \mathbf{B} := [\mathbf{a}_1 \otimes \mathbf{b}_1, \cdots \mathbf{a}_\ell \otimes \mathbf{b}_\ell]$. It is easy to see that, with $\mathbf{D}$ being a diagonal matrix with vector $\mathbf{d}$ on its diagonal (we will write $\mathbf{D} = \text{Diag}(\mathbf{d})$, and $\mathbf{d} = \text{diag}(\mathbf{D})$, where we have implicitly defined operators $\text{Diag}(\cdot)$ and $\text{diag}(\cdot)$ to convert one to the other), the following property holds

$$\text{vec}\left(\mathbf{A}\mathbf{D}\mathbf{B}^T\right) = (\mathbf{B} \odot \mathbf{A})\,\mathbf{d},$$

which is useful when dealing with linear least squares problems of the following form

$$\min_{\mathbf{D}=\text{Diag}(\mathbf{d})} ||\mathbf{X} - \mathbf{A}\mathbf{D}\mathbf{B}^T||_F^2 \Longleftrightarrow \min_{\mathbf{d}} ||\text{vec}(\mathbf{X}) - (\mathbf{B} \odot \mathbf{A})\mathbf{d}||_2^2.$$

It should now be clear that the Khatri–Rao product $\mathbf{B} \odot \mathbf{A}$ is a subset of columns from $\mathbf{B} \otimes \mathbf{A}$. Whereas $\mathbf{B} \otimes \mathbf{A}$ contains the 'interaction' (Kronecker product) of *any* column of $\mathbf{A}$ with *any* column of $\mathbf{B}$, $\mathbf{B} \odot \mathbf{A}$ contains the Kronecker product of *any* column of $\mathbf{A}$ with *only the corresponding* column of $\mathbf{B}$.

Additional properties:

- $(\mathbf{A} \otimes \mathbf{B}) \otimes \mathbf{C} = \mathbf{A} \otimes (\mathbf{B} \otimes \mathbf{C})$ (associative); so we may simply write as $\mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C}$. Note though that $\mathbf{A} \otimes \mathbf{B} \neq \mathbf{B} \otimes \mathbf{A}$, so the Kronecker product is non-commutative.
- $(\mathbf{A} \otimes \mathbf{B})^T = \mathbf{A}^T \otimes \mathbf{B}^T$ (note order, unlike $(\mathbf{A}\mathbf{B})^T = \mathbf{B}^T\mathbf{A}^T$).
- $(\mathbf{A} \otimes \mathbf{B})^* = \mathbf{A}^* \otimes \mathbf{B}^* \Longrightarrow (\mathbf{A} \otimes \mathbf{B})^H = \mathbf{A}^H \otimes \mathbf{B}^H$, where *, ^{H} stand for conjugation and Hermitian (conjugate) transposition, respectively.
- $(\mathbf{A} \otimes \mathbf{B})(\mathbf{E} \otimes \mathbf{F}) = (\mathbf{A}\mathbf{E} \otimes \mathbf{B}\mathbf{F})$ (the *mixed product rule*). This is very useful – as a corollary, if $\mathbf{A}$ and $\mathbf{B}$ are square nonsingular, then it follows that $(\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1}$, and likewise for the pseudo-inverse. More generally, if $\mathbf{A} = \mathbf{U}_1\mathbf{\Sigma}_1\mathbf{V}_1^T$ is the SVD of $\mathbf{A}$, and $\mathbf{B} = \mathbf{U}_2\mathbf{\Sigma}_2\mathbf{V}_2^T$ is the SVD of $\mathbf{B}$, then it follows from the mixed product rule that $\mathbf{A} \otimes \mathbf{B} = (\mathbf{U}_1\mathbf{\Sigma}_1\mathbf{V}_1^T) \otimes (\mathbf{U}_2\mathbf{\Sigma}_2\mathbf{V}_2^T) = (\mathbf{U}_1 \otimes \mathbf{U}_2)(\mathbf{\Sigma}_1 \otimes \mathbf{\Sigma}_2)(\mathbf{V}_1 \otimes \mathbf{V}_2)_2^T$ is the SVD of $\mathbf{A} \otimes \mathbf{B}$. It follows that
- $\text{rank}(\mathbf{A} \otimes \mathbf{B}) = \text{rank}(\mathbf{A})\text{rank}(\mathbf{B})$.
- $\text{tr}(\mathbf{A} \otimes \mathbf{B}) = \text{tr}(\mathbf{A})\text{tr}(\mathbf{B})$, for square $\mathbf{A}$, $\mathbf{B}$.
- $\det(\mathbf{A} \otimes \mathbf{B}) = \det(\mathbf{A})\det(\mathbf{B})$, for square $\mathbf{A}$, $\mathbf{B}$.

The Khatri–Rao product has the following properties, among others:

- $(\mathbf{A} \odot \mathbf{B}) \odot \mathbf{C} = \mathbf{A} \odot (\mathbf{B} \odot \mathbf{C})$ (associative); so we may simply write as $\mathbf{A} \odot \mathbf{B} \odot \mathbf{C}$. Note though that $\mathbf{A} \odot \mathbf{B} \neq \mathbf{B} \odot \mathbf{A}$, so the Khatri–Rao product is non-commutative.
- $(\mathbf{A} \otimes \mathbf{B})(\mathbf{E} \odot \mathbf{F}) = (\mathbf{A}\mathbf{E}) \odot (\mathbf{B}\mathbf{F})$ (*mixed product rule*).

Tensor (outer) product: The *tensor product* or *outer product* of vectors $\mathbf{a}$ $(I \times 1)$ and $\mathbf{b}$ $(J \times 1)$ is defined as the $I \times J$ matrix $\mathbf{a} \circledcirc \mathbf{b}$ with elements $(\mathbf{a} \circledcirc \mathbf{b})(i,j) = \mathbf{a}(i)\mathbf{b}(j)$, $\forall i, j$.

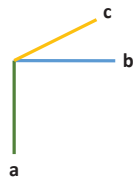

Fig. 1: Schematic of a rank-1 tensor.

Note that $\mathbf{a} \circledcirc \mathbf{b} = \mathbf{a}\mathbf{b}^T$. Introducing a third vector $\mathbf{c}$ $(K \times 1)$, we can generalize to the outer product of three vectors, which is an $I \times J \times K$ *three-way array* or *third-order tensor* $\mathbf{a} \circledcirc \mathbf{b} \circledcirc \mathbf{c}$ with elements $(\mathbf{a} \circledcirc \mathbf{b} \circledcirc \mathbf{c})(i,j,k) = \mathbf{a}(i)\mathbf{b}(j)\mathbf{c}(k)$. Note that the element-wise definition of the outer product naturally generalizes to three- and higher-way cases involving more vectors, but one loses the 'transposition' representation that is familiar in the two-way (matrix) case.

III. Rank and rank decomposition for tensors: CPD / PARAFAC

We know that the outer product of two vectors is a 'simple' rank-1 matrix – in fact we may define matrix rank as the minimum number of rank-1 matrices (outer products of two vectors) needed to synthesize a given matrix. We can express this in different ways: $\text{rank}(\mathbf{X}) = F$ if and only if (iff) F is the smallest integer such that $\mathbf{X} = \mathbf{A}\mathbf{B}^T$ for some $\mathbf{A} = [\mathbf{a}_1, \cdots, \mathbf{a}_F]$ and $\mathbf{B} = [\mathbf{b}_1, \cdots, \mathbf{b}_F]$, or, equivalently, $\mathbf{X}(i,j) = \sum_{f=1}^F \mathbf{A}(i,f)\mathbf{B}(j,f) = \sum_{f=1}^F \mathbf{a}_f(i)\mathbf{b}_f(j)$, $\forall i, j \Longleftrightarrow \mathbf{X} = \sum_{f=1}^F \mathbf{a}_f \circledcirc \mathbf{b}_f = \sum_{f=1}^F \mathbf{a}_f\mathbf{b}_f^T$.

A *rank-1* third-order tensor $\mathbf{X}$ of size $I \times J \times K$ is an outer product of three vectors: $\mathbf{X}(i,j,k) = \mathbf{a}(i)\mathbf{b}(j)\mathbf{c}(k)$, $\forall i \in \{1, \cdots, I\}$, $j \in \{1, \cdots, J\}$, and $k \in \{1, \cdots, K\}$; i.e., $\mathbf{X} = \mathbf{a} \circledcirc \mathbf{b} \circledcirc \mathbf{c}$ – see Fig. 1. A rank-1 N-th order tensor $\mathbf{X}$ is likewise an outer product of N vectors: $\mathbf{X}(i_1, \cdots, i_N) = \mathbf{a}_1(i_1) \cdots \mathbf{a}_N(i_N)$, $\forall i_n \in \{1, \cdots, I_n\}$, $\forall n \in \{1, \cdots, N\}$; i.e., $\mathbf{X} = \mathbf{a}_1 \circledcirc \cdots \circledcirc \mathbf{a}_N$. In the sequel we mostly focus on third-order tensors for brevity; everything naturally generalizes to higher-order tensors, and we will occasionally comment on such generalization, where appropriate.

The *rank* of tensor $\mathbf{X}$ is the minimum number of rank-1 tensors needed to produce $\mathbf{X}$ as their sum – see Fig. 2 for a tensor of rank three. Therefore, a tensor of rank at most F can be written as

$$\mathbf{X} = \sum_{f=1}^F \mathbf{a}_f \circledcirc \mathbf{b}_f \circledcirc \mathbf{c}_f \Longleftrightarrow \mathbf{X}(i,j,k) = \sum_{f=1}^F \mathbf{a}_f(i)\mathbf{b}_f(j)\mathbf{c}_f(k)$$

$$= \sum_{f=1}^F \mathbf{A}(i,f)\mathbf{B}(j,f)\mathbf{C}(k,f), \quad \begin{cases} & i \in \{1, \cdots, I\} \\ \forall & j \in \{1, \cdots, J\} \\ & k \in \{1, \cdots, K\} \end{cases}$$

where $\mathbf{A} := [\mathbf{a}_1, \cdots, \mathbf{a}_F]$, $\mathbf{B} := [\mathbf{b}_1, \cdots, \mathbf{b}_F]$, and $\mathbf{C} := [\mathbf{c}_1, \cdots, \mathbf{c}_F]$. It is also customary to use $a_{i,f} := \mathbf{A}(i,f)$, so $\mathbf{X}(i,j,k) = \sum_{f=1}^F a_{i,f}b_{j,f}c_{k,f}$. For brevity, we sometimes also use the notation $\mathbf{X} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ to denote the relationship $\mathbf{X} = \sum_{f=1}^F \mathbf{a}_f \circledcirc \mathbf{b}_f \circledcirc \mathbf{c}_f$.

Let us now fix $k = 1$ and look at the *frontal slab* $\mathbf{X}(:,:,1)$ of $\mathbf{X}$. Its elements can be written as

$$\mathbf{X}(i,j,1) = \sum_{f=1}^F \mathbf{a}_f(i)\mathbf{b}_f(j)\mathbf{c}_f(1)$$

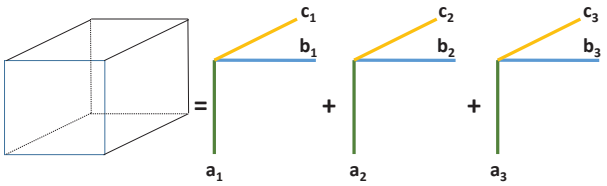

Fig. 2: Schematic of tensor of rank three.

$$\Longrightarrow \mathbf{X}(:,:,1) = \sum_{f=1}^{F} \mathbf{a}_f \mathbf{b}_f^T \mathbf{c}_f(1) =$$

$$\mathbf{A}\text{Diag}([\mathbf{c}_1(1), \mathbf{c}_2(1), \cdots, \mathbf{c}_F(1)])\mathbf{B}^T = \mathbf{A}\text{Diag}(\mathbf{C}(1,:))\mathbf{B}^T,$$

where we note that the elements of the first row of $\mathbf{C}$ weigh the rank-1 factors (outer products of corresponding columns of $\mathbf{A}$ and $\mathbf{B}$). We will denote $\text{D}_k(\mathbf{C}) := \text{Diag}(\mathbf{C}(k,:))$ for brevity. Hence, for any k,

$$\mathbf{X}(:,:,k) = \mathbf{A}\text{D}_k(\mathbf{C})\mathbf{B}^T.$$

Applying the vectorization property of $\odot$ it now follows that

$$\text{vec}(\mathbf{X}(:,:,k)) = (\mathbf{B} \odot \mathbf{A})(\mathbf{C}(k,:))^T,$$

and by parallel stacking, we obtain the matrix *unfolding* (or, matrix *view*)

$$\mathbf{X}_3 := [\text{vec}(\mathbf{X}(:,:,1)), \text{vec}(\mathbf{X}(:,:,2)), \cdots, \text{vec}(\mathbf{X}(:,:,K))] \rightarrow$$
$$\mathbf{X}_3 = (\mathbf{B} \odot \mathbf{A})\mathbf{C}^T, \quad (IJ \times K). \tag{1}$$

We see that, when cast as a matrix, a third-order tensor of rank F admits factorization in two matrix factors, *one of which is specially structured* – being the Khatri–Rao product of two smaller matrices. One more application of the vectorization property of $\odot$ yields the $IJK \times 1$ vector

$$\mathbf{x}_3 = (\mathbf{C} \odot (\mathbf{B} \odot \mathbf{A}))\,\mathbf{1} = (\mathbf{C} \odot \mathbf{B} \odot \mathbf{A})\,\mathbf{1},$$

where $\mathbf{1}$ is an $F \times 1$ vector of all 1's. Hence, when converted to a long vector, a tensor of rank F is a sum of F *structured* vectors, each being the Khatri–Rao / Kronecker product of three vectors (in the three-way case; or more vectors in higher-way cases).

In the same vain, we may consider lateral or horizontal slabs[2], e.g.,

$$\mathbf{X}(:,j,:) = \mathbf{A}\text{D}_j(\mathbf{B})\mathbf{C}^T \rightarrow \text{vec}(\mathbf{X}(:,j,:)) = (\mathbf{C} \odot \mathbf{A})(\mathbf{B}(j,:))^T.$$

Hence

$$\mathbf{X}_2 := [\text{vec}(\mathbf{X}(:,1,:)), \text{vec}(\mathbf{X}(:,2,:)), \cdots, \text{vec}(\mathbf{X}(:,J,:))] \rightarrow$$
$$\mathbf{X}_2 = (\mathbf{C} \odot \mathbf{A})\mathbf{B}^T, \quad (IK \times J), \tag{2}$$

and similarly[3] $\mathbf{X}(i,:,:) = \mathbf{B}\text{D}_i(\mathbf{A})\mathbf{C}^T$, so

$$\mathbf{X}_1 := [\text{vec}(\mathbf{X}(1,:,:)), \text{vec}(\mathbf{X}(2,:,:)), \cdots, \text{vec}(\mathbf{X}(I,:,:))] \rightarrow$$
$$\mathbf{X}_1 = (\mathbf{C} \odot \mathbf{B})\mathbf{A}^T, \quad (KJ \times I). \tag{3}$$

[2]A warning for Matlab aficionados: due to the way that Matlab stores and handles tensors, one needs to use the 'squeeze' operator, i.e., $\text{squeeze}(\mathbf{X}(:,j,:)) = \mathbf{A}\text{D}_j(\mathbf{B})\mathbf{C}^T$, and $\text{vec}(\text{squeeze}(\mathbf{X}(:,j,:))) = (\mathbf{C} \odot \mathbf{A})(\mathbf{B}(j,:))^T$.

[3]One needs to use the 'squeeze' operator here as well.

A. Low-rank tensor approximation

We are in fact ready to get a first glimpse on how we can go about estimating $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ from (possibly noisy) data $\mathbf{X}$. Adopting a least squares criterion, the problem is

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} ||\mathbf{X} - \sum_{f=1}^{F} \mathbf{a}_f \circledcirc \mathbf{b}_f \circledcirc \mathbf{c}_f||_F^2,$$

where $||\mathbf{X}||_F^2$ is the sum of squares of all elements of $\mathbf{X}$ (the subscript F in $||\cdot||_F$ stands for *Frobenius* (norm), and it should not be confused with the number of *factors* F in the rank decomposition – the difference will always be clear from context). Equivalently, we may consider

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} ||\mathbf{X}_1 - (\mathbf{C} \odot \mathbf{B})\mathbf{A}^T||_F^2.$$

Note that the above model is nonconvex (in fact *trilinear*) in $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$; but fixing $\mathbf{B}$ and $\mathbf{C}$, it becomes (conditionally) linear in $\mathbf{A}$, so that we may update

$$\mathbf{A} \leftarrow \arg\min_{\mathbf{A}} ||\mathbf{X}_1 - (\mathbf{C} \odot \mathbf{B})\mathbf{A}^T||_F^2,$$

and, using the other two matrix representations of the tensor, update

$$\mathbf{B} \leftarrow \arg\min_{\mathbf{B}} ||\mathbf{X}_2 - (\mathbf{C} \odot \mathbf{A})\mathbf{B}^T||_F^2,$$

and

$$\mathbf{C} \leftarrow \arg\min_{\mathbf{C}} ||\mathbf{X}_3 - (\mathbf{B} \odot \mathbf{A})\mathbf{C}^T||_F^2,$$

until convergence. The above algorithm, widely known as *Alternating Least Squares* (ALS) is a popular way of computing approximate low-rank models of tensor data. We will discuss algorithmic issues in depth at a later stage, but it is important to note that ALS is very easy to program, and we encourage the reader to do so – this exercise helps a lot in terms of developing the ability to 'think three-way'.

B. Bounds on tensor rank

For an $I \times J$ matrix $\mathbf{X}$, we know that $\text{rank}(\mathbf{X}) \leq \min(I,J)$, and $\text{rank}(\mathbf{X}) = \min(I,J)$ almost surely, meaning that rank-deficient real (complex) matrices are a set of Lebesgue measure zero in $\mathbb{R}^{I \times J}$ ($\mathbb{C}^{I \times J}$). What can we say about $I \times J \times K$ tensors $\mathbf{X}$? Before we get to this, a retrospective on the matrix case is useful. Considering $\mathbf{X} = \mathbf{A}\mathbf{B}^T$ where $\mathbf{A}$ is $I \times F$ and $\mathbf{B}$ is $J \times F$, the size of such parametrization (the *number of unknowns*, or *degrees of freedom* (DoF) in the model) of $\mathbf{X}$ is[4] $(I + J - 1)F$. The number of equations in $\mathbf{X} = \mathbf{A}\mathbf{B}^T$ is IJ, and equations-versus-unknowns considerations suggest that F of order $\min(I,J)$ may be needed – and this turns out being sufficient as well.

For third-order tensors, the DoF in the low-rank parametrization $\mathbf{X} = \sum_{f=1}^{F} \mathbf{a}_f \circledcirc \mathbf{b}_f \circledcirc \mathbf{c}_f$ is[5] $(I+J+K-2)F$, whereas the number of equations is IJK. This suggests that $F \geq \lceil \frac{IJK}{I+J+K-2} \rceil$ may be needed to describe an *arbitrary*

[4]Note that we have taken away F DoF due to the scaling / counter-scaling ambiguity, i.e., we may always multiply a column of $\mathbf{A}$ and divide the corresponding column of $\mathbf{B}$ with any nonzero number without changing $\mathbf{A}\mathbf{B}^T$.

[5]Note that here we can scale, e.g., $\mathbf{a}_f$ and $\mathbf{b}_f$ at will, and counter-scale $\mathbf{c}_f$, which explains the $(\ldots - 2)F$.

tensor $\mathbf{X}$ of size $I \times J \times K$, i.e., that third-order tensor rank can potentially be as high as $\min(IJ, JK, IK)$. In fact this turns out being sufficient as well. One way to see this is as follows: any frontal slab $\mathbf{X}(:,:,k)$ can always be written as $\mathbf{X}(:,:,k) = \mathbf{A}_k\mathbf{B}_k^T$, with $\mathbf{A}_k$ and $\mathbf{B}_k$ having at most $\min(I, J)$ columns. Upon defining $\mathbf{A} := [\mathbf{A}_1, \cdots, \mathbf{A}_K]$, $\mathbf{B} := [\mathbf{B}_1, \cdots, \mathbf{B}_K]$, and $\mathbf{C} := \mathbf{I}_{K\times K} \otimes \mathbf{1}_{1\times \min(I,J)}$ (where $\mathbf{I}_{K\times K}$ is an identity matrix of size $K \times K$, and $\mathbf{1}_{1\times \min(I,J)}$ is a vector of all 1's of size $1 \times \min(I, J)$), we can synthesize $\mathbf{X}$ as $\mathbf{X} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$. Noting that $\mathbf{A}_k$ and $\mathbf{B}_k$ have at most $\min(I, J)$ columns, it follows that we need at most $\min(IK, JK)$ columns in $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$. Using 'role symmetry' (switching the names of the 'ways' or 'modes'), it follows that we in fact need at most $\min(IJ, JK, IK)$ columns in $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and thus the rank of any $I \times J \times K$ three-way array $\mathbf{X}$ is bounded above by $\min(IJ, JK, IK)$. Another (cleaner but perhaps less intuitive) way of arriving at this result is as follows. Looking at the $IJ \times K$ matrix unfolding

$$\mathbf{X}_3 := [\text{vec}(\mathbf{X}(:,:,1)), \cdots, \text{vec}(\mathbf{X}(:,:,K))] = (\mathbf{B} \odot \mathbf{A})\mathbf{C}^T,$$

and noting that $(\mathbf{B} \odot \mathbf{A})$ is $IJ \times F$ and $\mathbf{C}^T$ is $F \times K$, the issue is what is the maximum inner dimension F that we need to be able to express an *arbitrary* $IJ \times K$ matrix $\mathbf{X}_3$ on the left (corresponding to an arbitrary $I \times J \times K$ tensor $\mathbf{X}$) as a Khatri–Rao product of two $I \times F$, $J \times F$ matrices, times another $F \times K$ matrix? The answer can be seen as follows:

$$\text{vec}(\mathbf{X}(:,:,k)) = \text{vec}(\mathbf{A}_k\mathbf{B}_k^T) = (\mathbf{B}_k \odot \mathbf{A}_k)\mathbf{1},$$

and thus we need at most $\min(I, J)$ columns per column of $\mathbf{X}_3$, which has K columns – QED.

This upper bound on tensor rank is important because it spells out that tensor rank is finite, and not much larger than the equations-versus-unknowns bound that we derived earlier. On the other hand, it is also useful to have lower bounds on rank. Towards this end, concatenate the frontal slabs one next to each other

$$[\mathbf{X}(:,:,1) \cdots \mathbf{X}(:,:,K)] = \mathbf{A}\left[\mathbf{D}_k(\mathbf{C})\mathbf{B}^T \cdots \mathbf{D}_k(\mathbf{C})\mathbf{B}^T\right]$$

since $\mathbf{X}(:,:,k) = \mathbf{A}\mathbf{D}_k(\mathbf{C})\mathbf{B}^T$. Note that $\mathbf{A}$ is $I \times F$, and it follows that F must be greater than or equal to the dimension of the column span of $\mathbf{X}$, i.e., the number of linearly independent columns needed to synthesize any of the JK columns $\mathbf{X}(:, j, k)$ of $\mathbf{X}$. By role symmetry, and upon defining

$$R_1(\mathbf{X}) := \text{dim colspan}(\mathbf{X}) := \text{dim span}\left\{\mathbf{X}(:, j, k)\right\}_{\forall j,k},$$

$$R_2(\mathbf{X}) := \text{dim rowspan}(\mathbf{X}) := \text{dim span}\left\{\mathbf{X}(i, :, k)\right\}_{\forall i,k},$$

$$R_3(\mathbf{X}) := \text{dim fiberspan}(\mathbf{X}) := \text{dim span}\left\{\mathbf{X}(i, j, :)\right\}_{\forall i,j},$$

we have that $F \geq \max(R_1(\mathbf{X}), R_2(\mathbf{X}), R_3(\mathbf{X}))$. $R_1(\mathbf{X})$ is the *mode-1 or mode-A rank* of $\mathbf{X}$, and likewise $R_2(\mathbf{X})$ and $R_3(\mathbf{X})$ are the *mode-2 or mode-B* and *mode-3 or mode-C* ranks of $\mathbf{X}$, respectively. $R_1(\mathbf{X})$ is sometimes called the *column rank*, $R_2(\mathbf{X})$ the *row rank*, and $R_3(\mathbf{X})$ the *fiber* or *tube rank* of $\mathbf{X}$. The triple $(R_1(\mathbf{X}), R_2(\mathbf{X}), R_3(\mathbf{X}))$ is called the *multilinear rank* of $\mathbf{X}$.

At this point it is worth noting that, for matrices we have that column rank = row rank = rank, i.e., in our current notation, for a matrix $\mathbf{M}$ (which can be thought of as an $I \times J \times 1$ third-order tensor) it holds that $R_1(\mathbf{M}) = R_2(\mathbf{M}) = \text{rank}(\mathbf{M})$, but for nontrivial tensors $R_1(\mathbf{X})$, $R_2(\mathbf{X})$, $R_3(\mathbf{X})$ and $\text{rank}(\mathbf{X})$ are in general different, with $\text{rank}(\mathbf{X}) \geq \max(R_1(\mathbf{X}), R_2(\mathbf{X}), R_3(\mathbf{X}))$. Since $R_1(\mathbf{X}) \leq I$, $R_2(\mathbf{X}) \leq J$, $R_3(\mathbf{X}) \leq K$, it follows that $\text{rank}(\mathbf{M}) \leq \min(I, J)$ for matrices but $\text{rank}(\mathbf{X})$ *can be* $> \max(I, J, K)$ for tensors.

Now, going back to the first way of explaining the upper bound we derived on tensor rank, it should be clear that we only need $\min(R_1(\mathbf{X}), R_2(\mathbf{X}))$ rank-1 factors to describe any given frontal slab of the tensor, and so we can describe all slabs with at most $\min(R_1(\mathbf{X}), R_2(\mathbf{X}))K$ rank-1 factors; with a little more thought, it is apparent that $\min(R_1(\mathbf{X}), R_2(\mathbf{X}))R_3(\mathbf{X})$ is enough. Appealing to role symmetry, it then follows that $F \leq \min(R_1(\mathbf{X})R_2(\mathbf{X}), R_2(\mathbf{X})R_3(\mathbf{X}), R_1(\mathbf{X})R_3(\mathbf{X}))$, where $F := \text{rank}(\mathbf{X})$. Dropping the explicit dependence on $\mathbf{X}$ for brevity, we have

$$\max(R_1, R_2, R_3) \leq F \leq \min(R_1R_2, R_2R_3, R_1R_3).$$

C. Typical, generic, and border rank of tensors

Consider a $2 \times 2 \times 2$ tensor $\mathbf{X}$ whose elements are i.i.d., drawn from the standard normal distribution $\mathcal{N}(0, 1)$ ($\mathbf{X} =$ randn(2,2,2) in Matlab). The rank of $\mathbf{X}$ *over the real field*, i.e., when we consider

$$\mathbf{X} = \sum_{f=1}^{F} \mathbf{a}_f \circledcirc \mathbf{b}_f \circledcirc \mathbf{c}_f, \quad \mathbf{a}_f \in \mathbb{R}^{2\times 1}, \mathbf{b}_f \in \mathbb{R}^{2\times 1}, \mathbf{c}_f \in \mathbb{R}^{2\times 1}, \forall f$$

is [22]

$$\text{rank}(\mathbf{X}) = \begin{cases} 2, & \text{with probability } \frac{\pi}{4} \\ 3, & \text{with probability } 1 - \frac{\pi}{4} \end{cases}$$

This is very different from the matrix case, where $\text{rank}(\text{randn}(2,2)) = 2$ with probability 1. To make matters more (or less) curious, the rank of the same $\mathbf{X} = \text{randn}(2,2,2)$ is in fact 2 with probability 1 when we instead consider decomposition over the complex field, i.e., using $\mathbf{a}_f \in \mathbb{C}^{2\times 1}, \mathbf{b}_f \in \mathbb{C}^{2\times 1}, \mathbf{c}_f \in \mathbb{C}^{2\times 1}, \forall f$. As another example [22], for $\mathbf{X} = \text{randn}(3,3,2)$,

$$\text{rank}(\mathbf{X}) = \begin{cases} \begin{array}{ll} 3, & \text{with probability } \frac{1}{2} \\ 4, & \text{with probability } \frac{1}{2} \end{array}, & \text{over } \mathbb{R}; \\ \\ 3, \quad \text{with probability } 1, & \text{over } \mathbb{C}. \end{cases}$$

To understand this behavior, consider the $2 \times 2 \times 2$ case. We have two 2×2 slabs, $\mathbf{S}_1 := \mathbf{X}(:,:,1)$ and $\mathbf{S}_2 := \mathbf{X}(:,:,2)$. For $\mathbf{X}$ to have $\text{rank}(\mathbf{X}) = 2$, we must be able to express these two slabs as

$$\mathbf{S}_1 = \mathbf{A}\mathbf{D}_1(\mathbf{C})\mathbf{B}^T, \text{ and } \quad \mathbf{S}_2 = \mathbf{A}\mathbf{D}_2(\mathbf{C})\mathbf{B}^T,$$

for some 2×2 real or complex matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, depending on whether we decompose over the real or the complex field. Now, if $\mathbf{X} =$ randn(2,2,2), then both $\mathbf{S}_1$ and $\mathbf{S}_2$ are nonsingular matrices, almost surely (with probability 1). It follows from the above equations that $\mathbf{A}$, $\mathbf{B}$, $\mathbf{D}_1(\mathbf{C})$, and $\mathbf{D}_2(\mathbf{C})$ must all be nonsingular too. Denoting

$\tilde{\mathbf{A}} := \mathbf{A}\mathbf{D}_1(\mathbf{C})$, $\mathbf{D} := (\mathbf{D}_1(\mathbf{C}))^{-1}\mathbf{D}_2(\mathbf{C})$, it follows that $\mathbf{B}^T = (\tilde{\mathbf{A}})^{-1}\mathbf{S}_1$, and substituting in the second equation we obtain $\mathbf{S}_2 = \tilde{\mathbf{A}}\mathbf{D}(\tilde{\mathbf{A}})^{-1}\mathbf{S}_1$, i.e., we obtain the eigen-problem

$$\mathbf{S}_2\mathbf{S}_1^{-1} = \tilde{\mathbf{A}}\mathbf{D}(\tilde{\mathbf{A}})^{-1}.$$

It follows that for rank$(\mathbf{X}) = 2$ over $\mathbb{R}$, the matrix $\mathbf{S}_2\mathbf{S}_1^{-1}$ should have two *real* eigenvalues; but complex conjugate eigenvalues do arise with positive probability. When they do, we have rank$(\mathbf{X}) = 2$ over $\mathbb{C}$, but rank$(\mathbf{X}) \geq 3$ over $\mathbb{R}$ – and it turns out that rank$(\mathbf{X}) = 3$ over $\mathbb{R}$ is enough.

We see that the rank of a tensor for decomposition over $\mathbb{R}$ is a random variable that can take more than one value with positive probability. These values are called *typical ranks*. For decomposition over $\mathbb{C}$ the situation is different: rank(randn(2,2,2)) $= 2$ with probability 1, so there is only one typical rank. When there is only one typical rank (that occurs with probability 1 then) we call it *generic rank*.

All these differences with the usual matrix algebra may be fascinating – and they don't end here either. Consider

$$\mathbf{X} = \mathbf{u} \circledcirc \mathbf{u} \circledcirc \mathbf{v} + \mathbf{u} \circledcirc \mathbf{v} \circledcirc \mathbf{u} + \mathbf{v} \circledcirc \mathbf{u} \circledcirc \mathbf{u},$$

where $||\mathbf{u}|| = ||\mathbf{v}|| = 1$, with $|<\mathbf{u},\mathbf{v}>| \neq 1$, where $<\cdot,\cdot>$ stands for the inner product. This tensor has rank equal to 3, however it can be *arbitrarily well* approximated [23] by the following sequence of rank-two tensors (see also [14]):

$$\mathbf{X}_n = n(\mathbf{u} + \frac{1}{n}\mathbf{v}) \circledcirc (\mathbf{u} + \frac{1}{n}\mathbf{v}) \circledcirc (\mathbf{u} + \frac{1}{n}\mathbf{v}) - n\mathbf{u} \circledcirc \mathbf{u} \circledcirc \mathbf{u}$$

$$= \mathbf{u} \circledcirc \mathbf{u} \circledcirc \mathbf{v} + \mathbf{u} \circledcirc \mathbf{v} \circledcirc \mathbf{u} + \mathbf{v} \circledcirc \mathbf{u} \circledcirc \mathbf{u} +$$

$$+\frac{1}{n}\mathbf{v} \circledcirc \mathbf{v} \circledcirc \mathbf{u} + \frac{1}{n}\mathbf{v} \circledcirc \mathbf{u} \circledcirc \mathbf{v} + \frac{1}{n}\mathbf{u} \circledcirc \mathbf{v} \circledcirc \mathbf{v} + \frac{1}{n^2}\mathbf{v} \circledcirc \mathbf{v} \circledcirc \mathbf{v},$$

so

$$\mathbf{X}_n = \mathbf{X} + \text{terms that vanish as } n \to \infty.$$

$\mathbf{X}$ has rank equal to 3, but *border rank* equal to 2 [15]. It is also worth noting that $\mathbf{X}_n$ contains two diverging rank-1 components that progressively cancel each other approximately, leading to ever-improving approximation of $\mathbf{X}$. This situation is actually encountered in practice when fitting tensors of border rank lower than their rank. Also note that the above example shows clearly that the low-rank tensor approximation problem

$$\min_{\{\mathbf{a}_f,\mathbf{b}_f,\mathbf{c}_f\}_{f=1}^F} \left|\left| \mathbf{X} - \sum_{f=1}^{F} \mathbf{a}_f \circledcirc \mathbf{b}_f \circledcirc \mathbf{c}_f \right|\right|_F^2,$$

is ill-posed in general, for there is no minimum if we pick F equal to the border rank of $\mathbf{X}$ – the set of tensors of a given rank is not closed. There are many ways to fix this ill-posedness, e.g., by adding constraints such as element-wise non-negativity of $\mathbf{a}_f, \mathbf{b}_f, \mathbf{c}_f$ [24], [25] in cases where $\mathbf{X}$ is element-wise non-negative (and these constraints are physically meaningful), or orthogonality [26] – any application-specific constraint that prevents terms from diverging while approximately canceling each other will do. An alternative is to add norm regularization to the cost function, such as $\lambda\left(||\mathbf{A}||_F^2 + ||\mathbf{B}||_F^2 + ||\mathbf{C}||_F^2\right)$. This can be interpreted as coming from a Gaussian prior on the sought parameter matrices; yet, if not properly justified, regularization may produce artificial results and a false sense of security.

TABLE I: Maximum attainable rank over $\mathbb{R}$.

Size	Maximum attainable rank over $\mathbb{R}$
$I \times J \times 2$	$\min(I, J) + \min(I, J, \lfloor \max(I, J)/2 \rfloor)$
$2 \times 2 \times 2$	3
$3 \times 3 \times 3$	5

TABLE II: Typical rank over $\mathbb{R}$

Size	Typical ranks over $\mathbb{R}$
$I \times I \times 2$	$\{I, I+1\}$
$I \times J \times 2$, $I > J$	$\min(I, 2J)$
$I \times J \times K$, $I > JK$	JK

TABLE III: Symmetry may affect typical rank.

Size	Typical ranks, $\mathbb{R}$ partial symmetry	Typical ranks, $\mathbb{R}$ no symmetry
$I \times I \times 2$	$\{I, I+1\}$	$\{I, I+1\}$
$9 \times 3 \times 3$	6	9

Some useful results on maximal and typical rank for decomposition over $\mathbb{R}$ are summarized in Tables I, II, III – see [14], [27] for more results of this kind, as well as original references. Notice that, for a tensor of a given size, there is always one typical rank over $\mathbb{C}$, which is therefore generic. For $I_1 \times I_2 \times \cdots \times I_N$ tensors, this generic rank is the value $\lceil \frac{\prod_{n=1}^N I_n}{\sum_{n=1}^N I_n - N + 1} \rceil$ that can be expected from the equations-versus-unknowns reasoning, except for the so-called defective cases (i) $I_1 > \prod_{n=2}^N I_n - \sum_{n=2}^N (I_n - 1)$ (assuming w.l.o.g. that the first dimension I_1 is the largest), (ii) the third-order case of dimension $(4, 4, 3)$, (iii) the third-order cases of dimension $(2p+1, 2p+1, 3)$, $p \in \mathbb{N}$, and (iv) the fourth-order cases of dimension $(p, p, 2, 2)$, $p \in \mathbb{N}$, where it is 1 higher [6]. Also note that the typical rank may change when the tensor is constrained in some way; e.g., when the frontal slabs are symmetric, we have the results in Table III, so symmetry may restrict the typical rank. Also, one may be interested in symmetric or asymmetric rank decomposition (i.e., symmetric or asymmetric rank-1 factors) in this case, and therefore symmetric or regular rank. Consider, for example, a fully symmetric tensor, i.e., one such that $\mathbf{X}(i,j,k) = \mathbf{X}(i,k,j) = \mathbf{X}(j,i,k) = \mathbf{X}(j,k,i) = \mathbf{X}(k,i,j) = \mathbf{X}(k,j,i)$, i.e., its value is invariant to any permutation of the three indices (the concept readily generalizes to N-way tensors $\mathbf{X}(i_i, \cdots, i_N)$). Then the *symmetric rank* of $\mathbf{X}$ over $\mathbb{C}$ is defined as the minimum R such that $\mathbf{X}$ can be written as $\mathbf{X} = \sum_{r=1}^R \mathbf{a}_r \circledcirc \mathbf{a}_r \circledcirc \cdots \circledcirc \mathbf{a}_r$, where the outer product involves N copies of vector $\mathbf{a}_r$, and $\mathbf{A} := [\mathbf{a}_1, \cdots, \mathbf{a}_R] \in \mathbb{C}^{I \times R}$. It has been shown that this symmetric rank equals $\lceil \binom{I+N-1}{N} / I \rceil$ almost surely except in the defective cases $(N, I) = (3, 5), (4, 3), (4, 4), (4, 5)$, where it is 1 higher [29]. Taking $N = 3$ as a special case, this formula gives $\frac{(I+1)(I+2)}{6}$. We also remark that constraints such as nonnegativity of a factor matrix can strongly affect rank.

Given a particular tensor $\mathbf{X}$, determining rank$(\mathbf{X})$ is NP-hard [30]. There is a well-known example of a $9 \times 9 \times 9$

[6] In fact this has been verified for $R \leq 55$, with the probability that a defective case has been overlooked less than 10^{-55}, the limitations being a matter of computing power [28].

tensor[7] whose rank (border rank) has been bounded between 19 and 23 (14 and 21, resp.), but has not been pinned down yet. At this point, the reader may rightfully wonder whether this is an issue in practical applications of tensor decomposition, or merely a mathematical curiosity? The answer is not black-and-white, but rather nuanced: In most applications, one is really interested in fitting a model that has the "essential" or "meaningful" number of components that we usually call the (useful signal) rank, which is usually much less than the actual rank of the tensor that we observe, due to noise and other imperfections. Determining this rank is challenging, even in the matrix case. There exist heuristics and a few more disciplined approaches that can help, but, at the end of the day, the process generally involves some trial-and-error.

An exception to the above is certain applications where the tensor actually models a mathematical object (e.g., a multilinear map) rather than "data". A good example of this is Strassen's matrix multiplication tensor – see the insert entitled *Tensors as bilinear operators*. A vector-valued (multiple-output) bilinear map can be represented as a third-order tensor, a vector-valued trilinear map as a fourth-order tensor, etc. When working with tensors that represent such maps, one is usually interested in exact factorization, and thus the mathematical rank of the tensor. The border rank is also of interest in this context, when the objective is to obtain a very accurate approximation (say, to within machine precision) of the given map. There are other applications (such as *factorization machines*, to be discussed later) where one is forced to approximate a general multilinear map in a possibly crude way, but then the number of components is determined by other means, not directly related to notions of rank.

Consider again the three matrix views of a given tensor $\mathbf{X}$ in (3), (2), (1). Looking at $\mathbf{X}_1$ in (1), note that if $(\mathbf{C} \odot \mathbf{B})$ is full column rank and so is $\mathbf{A}$, then $\text{rank}(\mathbf{X}_1) = F = \text{rank}(\mathbf{X})$. Hence this matrix view of $\mathbf{X}$ is rank-revealing. For this to happen it is *necessary* (but not sufficient) that $JK \geq F$, and $I \geq F$, so F has to be small: $F \leq \min(I, JK)$. Appealing to role symmetry of the three modes, it follows that $F \leq \max(\min(I, JK), \min(J, IK), \min(K, IJ))$ is necessary to have a rank-revealing matricization of the tensor. However, we know that the (perhaps unattainable) upper bound on $F = \text{rank}(\mathbf{X})$ is $F \leq \min(IJ, JK, IK)$, hence for matricization to reveal rank, it must be that the rank is really small relative to the upper bound. More generally, what holds for sure, as we have seen, is that $F = \text{rank}(\mathbf{X}) \geq \max(\text{rank}(\mathbf{X}_1), \text{rank}(\mathbf{X}_2), \text{rank}(\mathbf{X}_3))$.

Before we move on, let us extend what we have done so far to the case of N-way tensors. Let us start with 4-way tensors, whose rank decomposition can be written as

$$\mathbf{X}(i,j,k,\ell) = \sum_{f=1}^{F} \mathbf{a}_f(i)\mathbf{b}_f(j)\mathbf{c}_f(k)\mathbf{e}_f(\ell), \forall \begin{cases} i \in \{1, \cdots, I\} \\ j \in \{1, \cdots, J\} \\ k \in \{1, \cdots, K\} \\ \ell \in \{1, \cdots, L\} \end{cases}$$

$$\text{or, equivalently } \mathbf{X} = \sum_{f=1}^{F} \mathbf{a}_f \circledcirc \mathbf{b}_f \circledcirc \mathbf{c}_f \circledcirc \mathbf{e}_f.$$

[7]See the insert entitled *Tensors as bilinear operators*.

Tensors as bilinear operators: When multiplying two 2×2 matrices $\mathbf{M}_1$, $\mathbf{M}_2$, every element of the 2×2 result $\mathbf{P} = \mathbf{M}_1\mathbf{M}_2$ is a bilinear form $\text{vec}(\mathbf{M}_1)^T\mathbf{X}_k\text{vec}(\mathbf{M}_2)$, where $\mathbf{X}_k$ is 4×4, holding the coefficients that produce the k-th element of $\text{vec}(\mathbf{P})$, $k \in \{1, 2, 3, 4\}$. Collecting the slabs $\{\mathbf{X}_k\}_{k=1}^4$ into a $4 \times 4 \times 4$ tensor $\mathbf{X}$, matrix multiplication can be implemented by means of evaluating 4 bilinear forms involving the 4 frontal slabs of $\mathbf{X}$. Now suppose that $\mathbf{X}$ admits a rank decomposition involving matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ (all $4 \times F$ in this case). Then any element of $\mathbf{P}$ can be written as $\text{vec}(\mathbf{M}_1)^T\mathbf{A}\mathbf{D}_k(\mathbf{C})\mathbf{B}^T\text{vec}(\mathbf{M}_2)$. Notice that $\mathbf{B}^T\text{vec}(\mathbf{M}_2)$ can be computed using F inner products, and the same is true for $\text{vec}(\mathbf{M}_1)^T\mathbf{A}$. If the elements of $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ take values in $\{0, \pm 1\}$ (as it turns out, this is true for the "naive" as well as the minimal decomposition of $\mathbf{X}$), then these inner products require no multiplication – only selection, addition, subtraction. Letting $\mathbf{u}^T := \text{vec}(\mathbf{M}_1)^T\mathbf{A}$ and $\mathbf{v} := \mathbf{B}^T\text{vec}(\mathbf{M}_2)$, it remains to compute $\mathbf{u}^T\mathbf{D}_k(\mathbf{C})\mathbf{v} = \sum_{f=1}^{F} \mathbf{u}(f)\mathbf{v}(f)\mathbf{C}(k,f)$, $\forall k \in \{1, 2, 3, 4\}$. This entails F multiplications to compute the products $\{\mathbf{u}(f)\mathbf{v}(f)\}_{f=1}^F$ – the rest is all selections, additions, subtractions if $\mathbf{C}$ takes values in $\{0, \pm 1\}$. Thus F multiplications suffice to multiply two 2×2 matrices – and it so happens, that the rank of Strassen's $4 \times 4 \times 4$ tensor is 7, so $F = 7$ suffices. Contrast this to the "naive" approach which entails $F = 8$ multiplications (or, a "naive" decomposition of Strassen's tensor involving $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ all of size 4×8).

Upon defining $\mathbf{A} := [\mathbf{a}_1, \cdots, \mathbf{a}_F]$, $\mathbf{B} := [\mathbf{b}_1, \cdots, \mathbf{b}_F]$, $\mathbf{C} := [\mathbf{c}_1, \cdots, \mathbf{c}_F]$, $\mathbf{E} := [\mathbf{e}_1, \cdots, \mathbf{e}_F]$, we may also write

$$\mathbf{X}(i,j,k,\ell) = \sum_{f=1}^{F} \mathbf{A}(i,f)\mathbf{B}(j,f)\mathbf{C}(k,f)\mathbf{E}(\ell,f),$$

and we sometimes also use $\mathbf{X}(i,j,k,\ell) = \sum_{f=1}^{F} a_{i,f}b_{j,f}c_{k,f}e_{\ell,f}$. Now consider $\mathbf{X}(:,:,:,1)$, which is a third-order tensor. Its elements are given by

$$\mathbf{X}(i,j,k,1) = \sum_{f=1}^{F} a_{i,f}b_{j,f}c_{k,f}e_{1,f},$$

where we notice that the 'weight' $e_{1,f}$ is independent of i, j, k, it only depends on f, so we would normally absorb it in, say, $a_{i,f}$, if we only had to deal with $\mathbf{X}(:,:,:,1)$ – but here we don't, because we want to model $\mathbf{X}$ as a whole. Towards this end, let us vectorize $\mathbf{X}(:,:,:,1)$ into an $IJK \times 1$ vector

$$\text{vec}\left(\text{vec}\left(\mathbf{X}(:,:,:,1)\right)\right) = (\mathbf{C} \odot \mathbf{B} \odot \mathbf{A})(\mathbf{E}(1,:))^T,$$

where the result on the right should be contrasted with $(\mathbf{C} \odot \mathbf{B} \odot \mathbf{A})\mathbf{1}$, which would have been the result had we absorbed $e_{1,f}$ in $a_{i,f}$. Stacking one next to each other the vectors corresponding to $\mathbf{X}(:,:,:,1)$, $\mathbf{X}(:,:,:,2)$, $\cdots$, $\mathbf{X}(:,:,:,L)$, we obtain $(\mathbf{C} \odot \mathbf{B} \odot \mathbf{A})\mathbf{E}^T$; and after one more $\text{vec}(\cdot)$ we get $(\mathbf{E} \odot \mathbf{C} \odot \mathbf{B} \odot \mathbf{A})\mathbf{1}$.

It is also easy to see that, if we fix the last two indices and vary the first two, we get

$$\mathbf{X}(:,:,k,\ell) = \mathbf{A}\mathbf{D}_k(\mathbf{C})\mathbf{D}_\ell(\mathbf{E})\mathbf{B}^T,$$

Multiplying two complex numbers: Another interesting example involves the multiplication of two complex numbers – each represented as a 2×1 vector comprising its real and imaginary part. Let $j := \sqrt{-1}$, $x = x_r + jx_i \leftrightarrow \mathbf{x} := [x_r \ x_i]^T$, $y = y_r + jy_i \leftrightarrow \mathbf{y} := [y_r \ y_i]^T$. Then $xy = (x_r y_r - x_i y_i) + j(x_r y_i + x_r y_r) =: z_r + jz_i$. It appears that 4 real multiplications are needed to compute the result; but in fact 3 are enough. To see this, note that the $2 \times 2 \times 2$ multiplication tensor in this case has frontal slabs

$$\mathbf{X}(:,:,1) = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad \mathbf{X}(:,:,2) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

whose rank is at most 3, because

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}^T - \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}^T,$$

and

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}^T - \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}^T - \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}^T,$$

Thus taking

$$\mathbf{A} = \mathbf{B} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 1 & -1 & 0 \\ -1 & -1 & 1 \end{bmatrix},$$

we only need to compute $p_1 = x_r y_r$, $p_2 = x_i y_i$, $p_3 = (x_r + x_i)(y_r + y_i)$, and then $z_r = p_1 - p_2$, $z_i = p_3 - p_1 - p_2$. Of course, we did not need tensors to invent these computation schedules – but tensors can provide a way of obtaining them.

so that

$$\text{vec}\left(\mathbf{X}(:,:,k,\ell)\right) = (\mathbf{B} \odot \mathbf{A})(\mathbf{C}(k,:) * \mathbf{E}(\ell,:))^T,$$

where $*$ stands for the Hadamard (element-wise) matrix product. If we now stack these vectors one next to each other, we obtain the following "balanced" matricization[8] of the 4-th order tensor $\mathbf{X}$:

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

This is interesting because the inner dimension is F, so if $\mathbf{B} \odot \mathbf{A}$ and $\mathbf{E} \odot \mathbf{C}$ are both full column rank, then $F = \text{rank}(\mathbf{X}_b)$, i.e., the matricization $\mathbf{X}_b$ is *rank-revealing* in this case. Note that full column rank of $\mathbf{B} \odot \mathbf{A}$ and $\mathbf{E} \odot \mathbf{C}$ requires $F \leq \min(IJ, KL)$, which seems to be a more relaxed condition than in the three-way case. The catch is that, for 4-way tensors, the corresponding upper bound on tensor rank (obtained in the same manner as for third-order tensors) is $F \leq \min(IJK, IJL, IKL, JKL)$ – so the upper bound on tensor rank increases as well. Note that the boundary where matricization can reveal tensor rank remains off by one order of magnitude relative to the upper bound on rank, when $I = J = K = L$. In short: matricization can generally reveal the tensor rank in low-rank cases only.

Note that once we have understood what happens with 3-way and 4-way tensors, generalizing to N-way tensors for any integer $N \geq 3$ is easy. For a general N-way tensor, we can write it in scalar form as

$$\mathbf{X}(i_1, \cdots, i_N) = \sum_{f=1}^{F} \mathbf{a}_f^{(1)}(i_1) \cdots \mathbf{a}_f^{(N)}(i_N) = \sum_{f=1}^{F} a_{i_1,f}^{(1)} \cdots a_{i_N,f}^{(N)},$$

and in (combinatorially!) many different ways, including

$$\mathbf{X}_N = (\mathbf{A}_{N-1} \odot \cdots \odot \mathbf{A}_1)\mathbf{A}_N^T \rightarrow \text{vec}(\mathbf{X}_N) = (\mathbf{A}_N \odot \cdots \odot \mathbf{A}_1)\mathbf{1}.$$

We sometimes also use the shorthand $\text{vec}(\mathbf{X}_N) = \left(\odot_{n=N}^{1} \mathbf{A}_n\right)\mathbf{1}$, where $\text{vec}(\cdot)$ is now a compound operator, and the order of vectorization only affects the ordering of the factor matrices in the Khatri–Rao product.

IV. Uniqueness, demystified

We have already emphasized what is perhaps the most significant advantage of low-rank decomposition of third- and higher-order tensors versus low-rank decomposition of matrices (second-order tensors): namely, the former is essentially unique under mild conditions, whereas the latter is never essentially unique, unless the rank is equal to one, or else we impose additional constraints on the factor matrices. The reason why uniqueness happens for tensors but not for matrices may seem like a bit of a mystery at the beginning. The purpose of this section is to shed light in this direction, by assuming more stringent conditions than necessary to enable simple and insightful proofs. First, a concise definition of essential uniqueness.

Definition 1. *Given a tensor $\mathbf{X}$ of rank F, we say that its CPD is* essentially unique *if the F rank-1 terms in its decomposition (the outer products or "chicken feet") in Fig. 2 are unique, i.e., there is no other way to decompose $\mathbf{X}$ for the given number of terms. Note that we can of course permute these terms without changing their sum, hence there exists an inherently unresolvable permutation ambiguity in the rank-1 tensors. If $\mathbf{X} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$, with $\mathbf{A}: \ I \times F$, $\mathbf{B}: \ J \times F$, and $\mathbf{C}: \ K \times F$, then essential uniqueness means that $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are unique up to a common permutation and scaling / counter-scaling of columns, meaning that if $\mathbf{X} = [\![\bar{\mathbf{A}}, \bar{\mathbf{B}}, \bar{\mathbf{C}}]\!]$, for some $\bar{\mathbf{A}}: \ I \times F$, $\bar{\mathbf{B}}: \ J \times F$, and $\bar{\mathbf{C}}: \ K \times F$, then there exists a permutation matrix $\mathbf{\Pi}$ and diagonal scaling matrices $\mathbf{\Lambda}_1, \mathbf{\Lambda}_2, \mathbf{\Lambda}_3$ such that*

$$\bar{\mathbf{A}} = \mathbf{A}\mathbf{\Pi}\mathbf{\Lambda}_1, \ \bar{\mathbf{B}} = \mathbf{B}\mathbf{\Pi}\mathbf{\Lambda}_2, \ \bar{\mathbf{C}} = \mathbf{C}\mathbf{\Pi}\mathbf{\Lambda}_3, \ \mathbf{\Lambda}_1\mathbf{\Lambda}_2\mathbf{\Lambda}_3 = \mathbf{I}.$$

Remark 1. *Note that if we under-estimate the true rank $F = \text{rank}(\mathbf{X})$, it is impossible to fully decompose the given tensor using $R < F$ terms by definition. If we use $R > F$, uniqueness cannot hold unless we place conditions on $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$. In particular, for uniqueness it is necessary that each of the matrices $\mathbf{A} \odot \mathbf{B}$, $\mathbf{B} \odot \mathbf{C}$ and $\mathbf{C} \odot \mathbf{A}$ is full column rank. Indeed, if for instance $\mathbf{a}_R \otimes \mathbf{b}_R = \sum_{r=1}^{R-1} d_r \mathbf{a}_r \otimes \mathbf{b}_r$, then $\mathbf{X} = [\![\mathbf{A}(:, 1:R-1), \mathbf{B}(:, 1:R-1), \mathbf{C}(:, 1:R-1) + \mathbf{c}_R \mathbf{d}^T]\!]$, with $\mathbf{d} = [d_1, \cdots, d_{R-1}]^T$, is an alternative decomposition that involves only $R-1$ rank-1 terms, i.e. the number of rank-1 terms has been overestimated.*

We begin with the simplest possible line of argument. Consider an $I \times J \times 2$ tensor $\mathbf{X}$ of rank $F \leq \min(I, J)$. We know that the maximal rank of an $I \times J \times 2$ tensor over

[8] An alternative way to obtain this is to start from $(\mathbf{E} \odot \mathbf{C} \odot \mathbf{B} \odot \mathbf{A})\mathbf{1} = ((\mathbf{E} \odot \mathbf{C}) \odot (\mathbf{B} \odot \mathbf{A}))\mathbf{1} =$ vectorization of $(\mathbf{B} \odot \mathbf{A})(\mathbf{E} \odot \mathbf{X})^T$, by the vectorization property of $\odot$.

$\mathbb{R}$ is $\min(I,J)+\min(I,J,\lfloor \max(I,J)/2 \rfloor)$, and typical rank is $\min(I,2J)$ when $I>J$, or $\{I,I+1\}$ when $I=J$ (see Tables I, II) – so here we purposefully restrict ourselves to low-rank tensors (over $\mathbb{C}$ the argument is more general).

Let us look at the two frontal slabs of $\mathbf{X}$. Since $\text{rank}(\mathbf{X}) = F$, it follows that

$$\mathbf{X}^{(1)} := \mathbf{X}(:,:,1) = \mathbf{A}\mathbf{D}_1(\mathbf{C})\mathbf{B}^T,$$
$$\mathbf{X}^{(2)} := \mathbf{X}(:,:,2) = \mathbf{A}\mathbf{D}_2(\mathbf{C})\mathbf{B}^T,$$

where $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ are $I\times F$, $J\times F$, and $2\times F$, respectively. Let us assume that the multilinear rank of $\mathbf{X}$ is $(F,F,2)$, which implies that $\text{rank}(\mathbf{A}) = \text{rank}(\mathbf{B}) = F$. Now define the pseudo-inverse $\mathbf{E} := (\mathbf{B}^T)^\dagger$. It is clear that the columns of $\mathbf{E}$ are generalized eigenvectors of the matrix pencil $(\mathbf{X}^{(1)}, \mathbf{X}^{(2)})$:

$$\mathbf{X}^{(1)}\mathbf{e}_f = c_{1,f}\mathbf{a}_f, \quad \mathbf{X}^{(2)}\mathbf{e}_f = c_{2,f}\mathbf{a}_f.$$

(In the case $I=J$ and assuming that $\mathbf{X}^{(2)}$ is full rank, the Generalized EVD (GEVD) is algebraically equivalent with the basic EVD $\mathbf{X}^{(2)^{-1}}\mathbf{X}^{(1)} = \mathbf{B}^{-T}\mathbf{D}\mathbf{B}^T$ where $\mathbf{D} := \text{diag}(c_{1,1}/c_{2,1}, \cdots, c_{1,F}/c_{2,F})$; however, there are numerical differences.) For the moment we assume that the generalized eigenvalues are distinct, i.e. no two columns of $\mathbf{C}$ are proportional. There is freedom to scale the generalized eigenvectors (they remain generalized eigenvectors), and obviously one cannot recover the order of the columns of $\mathbf{E}$. This means that there is permutation and scaling ambiguity in recovering $\mathbf{E}$. That is, what we do recover is actually $\widetilde{\mathbf{E}} = \mathbf{E}\mathbf{\Pi}\mathbf{\Lambda}$, where $\mathbf{\Pi}$ is a permutation matrix and $\mathbf{\Lambda}$ is a nonsingular diagonal scaling matrix. If we use $\widetilde{\mathbf{E}}$ to recover $\mathbf{B}$, we will in fact recover $(\widetilde{\mathbf{E}}^T)^\dagger = \mathbf{B}\mathbf{\Pi}\mathbf{\Lambda}^{-1}$ – that is, $\mathbf{B}$ up to the same column permutation and scaling. It is now easy to see that we can recover $\mathbf{A}$ and $\mathbf{C}$ by going back to the original equations for $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ and multiplying from the right by $\widetilde{\mathbf{E}} = [\tilde{\mathbf{e}}_1, \cdots, \tilde{\mathbf{e}}_F]$. Indeed, since $\tilde{\mathbf{e}}_{\tilde{f}} = \lambda_{\tilde{f},\tilde{f}}\mathbf{e}_f$ for some f, we obtain per column a rank-1 matrix $\left[\mathbf{X}^{(1)}\tilde{\mathbf{e}}_{\tilde{f}}, \mathbf{X}^{(2)}\tilde{\mathbf{e}}_{\tilde{f}}\right] = \lambda_{\tilde{f},\tilde{f}}\mathbf{a}_f\mathbf{c}_f^T$, from which the corresponding column of $\mathbf{A}$ and $\mathbf{C}$ can be recovered.

The basic idea behind this type of EVD-based uniqueness proof has been rediscovered many times under different disguises and application areas. We refer the reader to Harshman (who also credits Jenkins) [31], [32]. The main idea is similar to a well-known parameter estimation technique in signal processing, known as ESPRIT [33]. A detailed and streamlined EVD proof that also works when $I\neq J$ and $F<\min(I,J)$ and is constructive (suitable for implementation) can be found in the supplementary material. That proof owes much to ten Berge [34] for the random slab mixing argument.

Remark 2. *Note that if we start by assuming that rank*$(\mathbf{X}) = F$ *over* $\mathbb{R}$*, then, by definition, all the matrices involved will be real, and the eigenvalues in* $\mathbf{D}$ *will also be real. If rank*$(\mathbf{X}) = F$ *over* $\mathbb{C}$*, then whether* $\mathbf{D}$ *is real or complex is not an issue.*

Note that there are F linearly independent eigenvectors *by construction* under our working assumptions. Next, if two or more of the generalized eigenvalues are identical, then linear combinations of the corresponding eigenvectors are also eigenvectors, corresponding to the same generalized eigenvalue. Hence distinct generalized eigenvalues are *necessary* for uniqueness.[9] The generalized eigenvalues are distinct if and only if *any two* columns of $\mathbf{C}$ are linearly independent – in which case we say that $\mathbf{C}$ has *Kruskal rank* ≥ 2. The definition of Kruskal rank is as follows.

Definition 2. *The* Kruskal rank $k_{\mathbf{A}}$ *of an* $I\times F$ *matrix* $\mathbf{A}$ *is the largest integer* k *such that* any k *columns of* $\mathbf{A}$ *are linearly independent. Clearly,* $k_{\mathbf{A}} \leq r_{\mathbf{A}} :=$ *rank*$(\mathbf{A}) \leq \min(I,F)$*. Note that* $k_{\mathbf{A}} = s_{\mathbf{A}} - 1 :=$ *spark*$(\mathbf{A}) - 1$*, where spark*$(\mathbf{A})$ *is the minimum number of linearly dependent columns of* $\mathbf{A}$ *(when this is* $\leq F$*). Spark is a familiar notion in the compressed sensing literature, but Kruskal rank was defined earlier.*

We will see that the notion of Kruskal rank plays an important role in uniqueness results in our context, notably in what is widely known as Kruskal's result (in fact, a "common denominator" implied by a number of results that Kruskal has proven in his landmark paper [35]). Before that, let us summarize the result we have just obtained.

Theorem 1. *Given* $\mathbf{X} = [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]$*, with* $\mathbf{A}: I\times F$*,* $\mathbf{B}: J\times F$*, and* $\mathbf{C}: 2\times F$*, if* $F>1$ *it is necessary for uniqueness of* $\mathbf{A}$*,* $\mathbf{B}$ *that* $k_{\mathbf{C}} = 2$*. If, in addition* $r_{\mathbf{A}} = r_{\mathbf{B}} = F$*, then rank*$(\mathbf{X}) = F$ *and the decomposition of* $\mathbf{X}$ *is essentially unique.*

For tensors that consist of $K\geq 2$ slices, one can consider a pencil of two random slice mixtures and infer the following result from Theorem 1.

Theorem 2. *Given* $\mathbf{X} = [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]$*, with* $\mathbf{A}: I\times F$*,* $\mathbf{B}: J\times F$*, and* $\mathbf{C}: K\times F$*, if* $F>1$ *it is necessary for uniqueness of* $\mathbf{A}$*,* $\mathbf{B}$ *that* $k_{\mathbf{C}}\geq 2$*. If, in addition* $r_{\mathbf{A}} = r_{\mathbf{B}} = F$*, then rank*$(\mathbf{X}) = F$ *and the decomposition of* $\mathbf{X}$ *is essentially unique.*

A probabilistic version of Theorem 2 goes as follows.

Theorem 3. *Given* $\mathbf{X} = [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]$*, with* $\mathbf{A}: I\times F$*,* $\mathbf{B}: J\times F$*, and* $\mathbf{C}: K\times F$*, if* $I\geq F$*,* $J\geq F$ *and* $K\geq 2$*, then rank*$(\mathbf{X}) = F$ *and the decomposition of* $\mathbf{X}$ *in terms of* $\mathbf{A}$*,* $\mathbf{B}$*, and* $\mathbf{C}$ *is essentially unique,* almost surely *(meaning that it is essentially unique for all* $\mathbf{X} = [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]$ *except for a set of measure zero with respect to the Lebesgue measure in* $\mathbb{R}^{(I+J+K-2)F}$ *or* $\mathbb{C}^{(I+J+K-2)F}$*).*

Now let us relax our assumptions and require that only one of the loading matrices is full column rank, instead of two. After some reflection, the matricization $\mathbf{X}^{(JI\times K)} := (\mathbf{A}\odot\mathbf{B})\,\mathbf{C}^T$ yields the following condition, which is both necessary and sufficient.

Theorem 4. *[36] Given* $\mathbf{X} = [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]$*, with* $\mathbf{A}: I\times F$*,* $\mathbf{B}: J\times F$*, and* $\mathbf{C}: K\times F$*, and assuming* $r_{\mathbf{C}} = F$*, it holds that the decomposition* $\mathbf{X} = [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]$ *is essentially unique* $\Longleftrightarrow$ *nontrivial linear combinations of columns of* $\mathbf{A}\odot\mathbf{B}$ *cannot be written as* $\otimes$ *product of two vectors.*

Despite its conceptual simplicity and appeal, the above condition is hard to check. In [36] it is shown that it is possible to recast this condition as an equivalent criterion

[9] Do note however that, even in this case, uniqueness breaks down only partially, as eigenvectors corresponding to other, distinct eigenvalues are still unique up to scaling.

on the solutions of a system of quadratic equations – which is also hard to check, but will serve as a stepping stone to easier conditions and even generalizations of the EVD-based computation. Let $\mathbf{M}_k(\mathbf{A})$ denote the $\binom{I}{k} \times \binom{F}{k}$ k-th compound matrix containing all $k \times k$ minors of $\mathbf{A}$, e.g., for

$$\mathbf{A} = \begin{bmatrix} a_1 & 1 & 0 & 0 \\ a_2 & 0 & 1 & 0 \\ a_3 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{M}_2(\mathbf{A}) = \begin{bmatrix} -a_2 & a_1 & 0 & 1 & 0 & 0 \\ -a_3 & 0 & a_1 & 0 & 1 & 0 \\ 0 & -a_3 & a_2 & 0 & 0 & 1 \end{bmatrix}.$$

Starting from a vector $\mathbf{d} = [d_1, \cdots, d_F]^T \in \mathbb{C}^F$, let $\mathbf{v}_k(\mathbf{d})$ consistently denote $[d_1 d_2 \cdots d_k, d_1 d_2 \cdots d_{k-1} d_{k+1}, \cdots, d_{F-k+1} d_{F-k+2} \cdots d_F]^T \in \mathbb{C}^{\binom{F}{k}}$. Theorem 4 can now be expressed as follows.

Theorem 5. *[36] Given $\mathbf{X} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$, with $\mathbf{A}: I \times F$, $\mathbf{B}: J \times F$, and $\mathbf{C}: K \times F$, and assuming $r_{\mathbf{C}} = F$, it holds that the decomposition*

$$\mathbf{X} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \text{ is essentially unique } \iff$$
$$(\mathbf{M}_2(\mathbf{B}) \odot \mathbf{M}_2(\mathbf{A}))\, \mathbf{v}_2(\mathbf{d}) = \mathbf{0}$$
$$\text{implies that } \mathbf{v}_2(\mathbf{d}) = [d_1 d_2, d_1 d_3, \cdots, d_{F-1} d_F]^T = \mathbf{0},$$
$$\text{i.e., at most one entry of } \mathbf{d} \text{ is nonzero.}$$

The size of $\mathbf{M}_2(\mathbf{B}) \odot \mathbf{M}_2(\mathbf{A})$ is $\binom{I}{2}\binom{J}{2} \times \binom{F}{2}$. A sufficient condition that can be checked with basic linear algebra is readily obtained by ignoring the structure of $\mathbf{v}_2(\mathbf{d})$.

Theorem 6. *[36], [37] If $r_{\mathbf{C}} = F$, and $r_{\mathbf{M}_2(\mathbf{B}) \odot \mathbf{M}_2(\mathbf{A})} = \binom{F}{2}$, then rank$(\mathbf{X}) = F$ and the decomposition of $\mathbf{X} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ is essentially unique.*

The generic version of Theorems 4 and 5 has been obtained from an entirely different (algebraic geometry) point of view:

Theorem 7. *[38]–[40] Given $\mathbf{X} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$, with $\mathbf{A}: I \times F$, $\mathbf{B}: J \times F$, and $\mathbf{C}: K \times F$, let $K \geq F$ and $\min(I, J) \geq 3$. Then rank$(\mathbf{X}) = F$ and the decomposition of $\mathbf{X}$ is essentially unique, almost surely, if and only if $(I-1)(J-1) \geq F$.*

The next theorem is the generic version of Theorem 6; the second inequality implies that $\mathbf{M}_2(\mathbf{B}) \odot \mathbf{M}_2(\mathbf{A})$ does not have more columns than rows.

Theorem 8. *Given $\mathbf{X} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$, with $\mathbf{A}: I \times F$, $\mathbf{B}: J \times F$, and $\mathbf{C}: K \times F$, if $K \geq F$ and $I(I-1)J(J-1) \geq 2F(F-1)$, then rank$(\mathbf{X}) = F$ and the decomposition of $\mathbf{X}$ is essentially unique, almost surely.*

Note that $(I-1)(J-1) \geq F \iff IJ - I - J + 1 \geq F \Rightarrow IJ \geq F - 1 + I + J \Rightarrow IJ \geq F - 1$, and multiplying the first and the last inequality yields $I(I-1)J(J-1) \geq F(F-1)$. So Theorem 7 is at least a factor of 2 more relaxed than Theorem 8. Put differently, ignoring the structure of $\mathbf{v}_2(\mathbf{d})$ makes us lose about a factor of 2 generically.

On the other hand, Theorem 6 admits a remarkable constructive interpretation. Consider any rank-revealing decomposition, such as a QR-factorization or an SVD, of $\mathbf{X}^{(JI \times K)} = \mathbf{E}\mathbf{F}^T$, involving a $JI \times F$ matrix $\mathbf{E}$ and a $K \times F$ matrix $\mathbf{F}$ that both are full column rank. (At this point, recall that full column rank of $\mathbf{A} \odot \mathbf{B}$ is necessary for uniqueness, and that $\mathbf{C}$ is full column rank by assumption.) We are interested in finding an $F \times F$ (invertible) basis transformation matrix $\mathbf{G}$ such that $\mathbf{A} \odot \mathbf{B} = \mathbf{E}\mathbf{G}$ and $\mathbf{C} = \mathbf{F}\mathbf{G}^{-T}$. It turns out that, under the conditions in Theorem 6 and through the computation of second compound matrices, an $F \times F \times F$ auxiliary tensor $\mathbf{Y}$ can be derived from the given tensor $\mathbf{X}$, admitting the CPD $\mathbf{Y} = [\![\widetilde{\mathbf{G}}, \widetilde{\mathbf{G}}, \mathbf{H}]\!]$, in which $\widetilde{\mathbf{G}}$ equals $\mathbf{G}$ up to column-wise scaling and permutation, and in which the $F \times F$ matrix $\mathbf{H}$ is nonsingular [37]. As the three loading matrices are full column rank, uniqueness of the auxiliary CPD is guaranteed by Theorem 2, and it can be computed by means of an EVD. Through a more sophisticated derivation of an auxiliary tensor, [41] attempts to regain the "factor of 2" above and extend the result up to the necessary and sufficient generic bound in Theorem 7; that the latter bound is indeed reached has been verified numerically up to $F = 24$.

Several results have been extended to situations where none of the loading matrices is full column rank, using m-th compound matrices ($m > 2$). For instance, the following theorem generalizes Theorem 6:

Theorem 9. *[42], [43] Given $\mathbf{X} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$, with $\mathbf{A}: I \times F$, $\mathbf{B}: J \times F$, and $\mathbf{C}: K \times F$. Let $m_{\mathbf{C}} = F - k_{\mathbf{C}} + 2$. If $\max(\min(k_{\mathbf{A}}, k_{\mathbf{B}} - 1), \min(k_{\mathbf{A}} - 1, k_{\mathbf{B}})) + k_{\mathbf{C}} \geq F + 1$ and $\mathbf{M}_{m_{\mathbf{C}}}(\mathbf{A}) \odot \mathbf{M}_{m_{\mathbf{C}}}(\mathbf{B})$ has full column rank, then rank$(\mathbf{X}) = F$ and the decomposition $\mathbf{X} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ is essentially unique.*

(To see that Theorem 9 reduces to Theorem 6 when $r_{\mathbf{C}} = F$, note that $r_{\mathbf{C}} = F$ implies $k_{\mathbf{C}} = F$ and recall that $\min(k_{\mathbf{A}}, k_{\mathbf{B}}) > 1$ is necessary for uniqueness.) Under the conditions in Theorem 9 computation of the CPD can again be reduced to a GEVD [43].

It can be shown [42], [43] that Theorem 9 implies the next theorem, which is the most well-known result covered by Kruskal; this includes the possibility of reduction to GEVD.

Theorem 10. *[35] Given $\mathbf{X} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$, with $\mathbf{A}: I \times F$, $\mathbf{B}: J \times F$, and $\mathbf{C}: K \times F$, if $k_{\mathbf{A}} + k_{\mathbf{B}} + k_{\mathbf{C}} \geq 2F + 2$, then rank$(\mathbf{X}) = F$ and the decomposition of $\mathbf{X}$ is essentially unique.*

Note that Theorem 10 is symmetric in $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, while in Theorem 9 the role of $\mathbf{C}$ is different from that of $\mathbf{A}$ and $\mathbf{B}$. Kruskal's condition is sharp, in the sense that there exist decompositions that are not unique as soon as F goes beyond the bound [44]. This does not mean that uniqueness is impossible beyond Kruskal's bound – as indicated, Theorem 9 also covers other cases. (Compare the generic version of Kruskal's condition, $\min(I, F) + \min(J, F) + \min(K, F) \geq 2F + 2$, with Theorem 7, for instance.)

Kruskal's original proof is beyond the scope of this overview paper; instead, we refer the reader to [45] for a compact version that uses only matrix algebra, and to the supplementary material for a relatively simple proof of an intermediate result which still conveys the flavor of Kruskal's derivation.

With respect to generic conditions, one could wonder whether a CPD is not unique almost surely for any value of F strictly less than the generic rank, see the equations-versus-unknowns discussion in Section III. For symmetric decompositions this has indeed been proved, with the exceptions $(N, I; F) = (6, 3; 9), (4, 4; 9), (3, 6; 9)$ where there are two decompositions generically [46]. For unsymmetric decompositions it has been verified for tensors up to 15000 entries (larger tensors can be analyzed with a larger computational effort) that the only exceptions are $(I_1, \cdots, I_N; F) = (4, 4, 3; 5)$, $(4, 4, 4; 6)$, $(6, 6, 3; 8)$, $(p, p, 2, 2; 2p-1)$ for $p \in \mathbb{N}$, $(2, 2, 2, 2, 2; 5)$, and the so-called unbalanced case $I_1 > \alpha$, $F \geq \alpha$, with $\alpha = \prod_{n=2}^{N} I_n - \sum_{n=2}^{N}(I_n - 1)$ [47].

Note that in the above we assumed that the factor matrices are unconstrained. (Partial) symmetry can be integrated in the deterministic conditions by substituting for instance $\mathbf{A} = \mathbf{B}$. (Partial) symmetry does change the generic conditions, as the number of equations / number of parameters ratio is affected, see [39] and references therein for variants. For the partial Hermitian symmetry $\mathbf{A} = \mathbf{B}^*$ we can do better by constructing the extended $I \times I \times 2K$ tensor $\mathbf{X}^{(\text{ext})}$ via $x^{(\text{ext})}_{i,j,k} = x_{i,j,k}$ for $k \leq K$ and $x^{(\text{ext})}_{i,j,k} = x^*_{j,i,k}$ for $K + 1 \leq k \leq 2K$. We have $\mathbf{X}^{(\text{ext})} = \left[\!\left[\mathbf{A}, \mathbf{A}^*, \mathbf{C}^{(\text{ext})}\right]\!\right]$, with $\mathbf{C}^{(\text{ext})} = \left[\mathbf{C}^T, \mathbf{C}^H\right]^T$. Since obviously $r_{\mathbf{C}^{(\text{ext})}} \geq r_{\mathbf{C}}$ and $k_{\mathbf{C}^{(\text{ext})}} \geq k_{\mathbf{C}}$, uniqueness is easier to establish for $\mathbf{X}^{(\text{ext})}$ than for $\mathbf{X}$ [48]. By exploiting orthogonality, some deterministic conditions can be relaxed as well [49]. For a thorough study of implications of nonnegativity, we refer to [25].

Summarizing, there exist several types of uniqueness conditions. First, there are probabilistic conditions that indicate whether it is reasonable to expect uniqueness for a certain number of terms, given the size of the tensor. Second, there are deterministic conditions that allow one to establish uniqueness for a particular decomposition – this is useful for an a posteriori analysis of the uniqueness of results obtained by a decomposition algorithm. There also exist deterministic conditions under which the decomposition can actually be computed using only conventional linear algebra (EVD or GEVD), at least under noise-free conditions. In the case of (mildly) noisy data, such algebraic algorithms can provide a good starting value for optimization-based algorithms (which will be discussed in Section VII), i.e. the algebraic solution is refined in an optimization step. Further, the conditions can be affected by constraints. While in the matrix case constraints can make a rank decomposition unique that otherwise is not unique, for tensors the situation is rather that constraints affect the range of values of F for which uniqueness holds.

There exist many more uniqueness results that we didn't touch upon in this overview, but the ones that we did present give a good sense of what is available and what one can expect. In closing this section, we note that many (but not all) of the above results have been extended to the case of higher-order (order $N > 3$) tensors. For example, the following result generalizes Kruskal's theorem to tensors of arbitrary order:

Theorem 11. *[50] Given* $\mathbf{X} = \left[\!\left[\mathbf{A}_1, \ldots, \mathbf{A}_N\right]\!\right]$, *with* $\mathbf{A}_n : I_n \times F$, *if* $\sum_{n=1}^{N} k_{\mathbf{A}_n} \geq 2F + N - 1$, *then the decomposition of* $\mathbf{X}$ *in terms of* $\{\mathbf{A}_n\}_{n=1}^{N}$ *is essentially unique.*

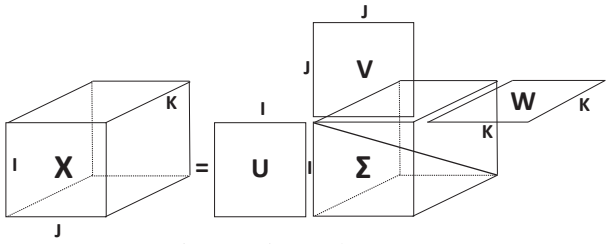

Fig. 3: Diagonal tensor SVD?

This condition is sharp in the same sense as the $N = 3$ version is sharp [44]. The starting point for proving Theorem 11 is that a fourth-order tensor of rank F can be written in third-order form as $\mathbf{X}_{[1,2;3;4]} = \left[\!\left[\mathbf{A}_1 \odot \mathbf{A}_2, \mathbf{A}_3, \mathbf{A}_4\right]\!\right] = \left[\!\left[\mathbf{A}_{[1,2]}, \mathbf{A}_3, \mathbf{A}_4\right]\!\right]$ – i.e., can be viewed as a third-order tensor with a specially structured mode loading matrix $\mathbf{A}_{[1,2]} := \mathbf{A}_1 \odot \mathbf{A}_2$. Therefore, Kruskal's third-order result can be applied, and what matters is the k-rank of the Khatri–Rao product $\mathbf{A}_1 \odot \mathbf{A}_2$ – see property 2 in the supplementary material, and [50] for the full proof.

V. The Tucker model and Multilinear Singular Value Decomposition

A. Tucker and CPD

Any $I \times J$ matrix $\mathbf{X}$ can be decomposed via SVD as $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{U}^T\mathbf{U} = \mathbf{I} = \mathbf{U}\mathbf{U}^T$, $\mathbf{V}^T\mathbf{V} = \mathbf{I} = \mathbf{V}\mathbf{V}^T$, $\mathbf{\Sigma}(i, j) \geq 0$, $\mathbf{\Sigma}(i, j) > 0$ only when $j = i$ and $i \leq r_{\mathbf{X}}$, and $\mathbf{\Sigma}(i, i) \geq \mathbf{\Sigma}(i + 1, i + 1)$, $\forall i$. With $\mathbf{U} := [\mathbf{u}_1, \cdots, \mathbf{u}_I]$, $\mathbf{V} := [\mathbf{v}_1, \cdots, \mathbf{v}_J]$, and $\sigma_f := \mathbf{\Sigma}(f, f)$, we can thus write $\mathbf{X} = \mathbf{U}(:, 1 : F)\mathbf{\Sigma}(1 : F, 1 : F)(\mathbf{V}(:, 1 : F))^T = \sum_{f=1}^{F} \sigma_f \mathbf{u}_f \mathbf{v}_f^T$.

The question here is whether we can generalize the SVD to tensors, and if there is a way of doing so that retains the many beautiful properties of matrix SVD. The natural generalization would be to employ another matrix, of size $K \times K$, call it $\mathbf{W}$, such that $\mathbf{W}^T\mathbf{W} = \mathbf{I} = \mathbf{W}\mathbf{W}^T$, and a nonnegative $I \times J \times K$ *core tensor* $\mathbf{\Sigma}$ such that $\mathbf{\Sigma}(i, j, k) > 0$ only when $k = j = i$ – see the schematic illustration in Fig. 3. Is it possible to decompose an arbitrary tensor in this way? A back-of-the-envelop calculation shows that the answer is no. Even disregarding the orthogonality constraints, the degrees of freedom in such a decomposition would be less[10] than $I^2 + J^2 + K^2 + \min(I, J, K)$, which is in general $< IJK$ – the number of (nonlinear) equality constraints. [Note that, for matrices, $I^2 + J^2 + \min(I, J) > I^2 + J^2 > IJ$, always.] A more formal way to look at this is that the model depicted in Fig. 3 can be written as

$$\sigma_1 \mathbf{u}_1 \circledcirc \mathbf{v}_1 \circledcirc \mathbf{w}_1 + \sigma_2 \mathbf{u}_2 \circledcirc \mathbf{v}_2 \circledcirc \mathbf{w}_2 + \cdots + \sigma_m \mathbf{u}_m \circledcirc \mathbf{v}_m \circledcirc \mathbf{w}_m,$$

where $m := \min(I, J, K)$. The above is a tensor of rank at most $\min(I, J, K)$, but we know that tensor rank can be (much) higher than that. Hence we certainly have to give up diagonality. Consider instead a full (possibly dense, but ideally sparse) core tensor $\mathbf{G}$, as illustrated in Fig. 4. An element-

[10] Since the model exhibits scaling/counter-scaling invariances.

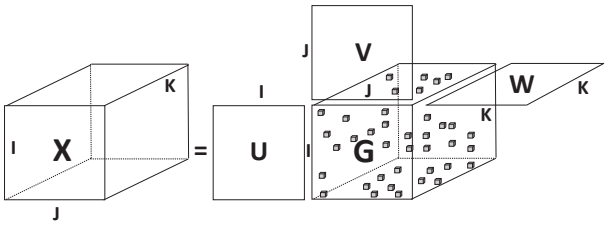

Fig. 4: The Tucker model

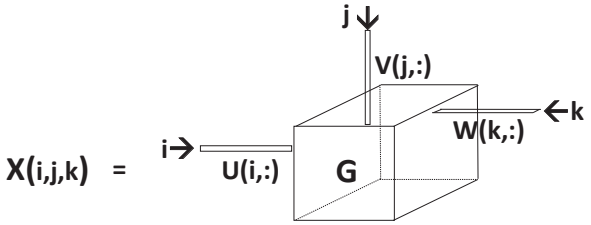

Fig. 5: Element-wise view of the Tucker model

wise interpretation of the decomposition in Fig. 4 is shown in Fig. 5. From Fig. 5, we write

$$\mathbf{X}(i,j,k) = \sum_{\ell=1}^{I}\sum_{m=1}^{J}\sum_{n=1}^{K} \mathbf{G}(\ell,m,n)\mathbf{U}(i,\ell)\mathbf{V}(j,m)\mathbf{W}(k,n),$$

or, equivalently,

$$\mathbf{X} = \sum_{\ell=1}^{I}\sum_{m=1}^{J}\sum_{n=1}^{K} \mathbf{G}(\ell,m,n)\mathbf{U}(:,\ell) \circledcirc \mathbf{V}(:,m) \circledcirc \mathbf{W}(:,n),$$

$$\text{or } \mathbf{X} = \sum_{\ell=1}^{I}\sum_{m=1}^{J}\sum_{n=1}^{K} \mathbf{G}(\ell,m,n)\mathbf{u}_\ell \circledcirc \mathbf{v}_m \circledcirc \mathbf{w}_n, \qquad (4)$$

where $\mathbf{u}_\ell := \mathbf{U}(:,\ell)$ and likewise for the $\mathbf{v}_m$, $\mathbf{w}_n$. Note that each column of $\mathbf{U}$ interacts with every column of $\mathbf{V}$ and every column of $\mathbf{W}$ in this decomposition, and the strength of this interaction is encoded in the corresponding element of $\mathbf{G}$. This is different from the rank decomposition model (CPD) we were discussing until this section, which only allows interactions between corresponding columns of $\mathbf{A}, \mathbf{B}, \mathbf{C}$, i.e., the only outer products that can appear in the CPD are of type $\mathbf{a}_f \circledcirc \mathbf{b}_f \circledcirc \mathbf{c}_f$. On the other hand, we emphasize that the *Tucker model* in (4) also allows "mixed" products of non-corresponding columns of $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$. Note that *any* tensor $\mathbf{X}$ can be written in Tucker form (4), and a trivial way of doing so is to take $\mathbf{U} = \mathbf{I}_{I\times I}$, $\mathbf{V} = \mathbf{I}_{J\times J}$, $\mathbf{W} = \mathbf{I}_{K\times K}$, and $\mathbf{G} = \mathbf{X}$. Hence we may seek a possibly sparse $\mathbf{G}$, which could help reveal the underlying "essential" interactions between triples of columns of $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$. This is sometimes useful when one is interested in quasi-CPD models. The main interest in Tucker though is for finding subspaces and for tensor approximation purposes.

From the above discussion, it may appear that CPD is a special case of the Tucker model, which appears when $\mathbf{G}(\ell,m,n) = 0$ for all ℓ, m, n except possibly for $\ell = m = n$. However, when $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$ are all square, such a restricted diagonal Tucker form can only model tensors up to rank $\min(I,J,K)$. If we allow "fat" (and therefore, clearly, non-orthogonal) $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$ in Tucker though, it is possible to think of CPD as a special case of such a "blown-up" non-orthogonal Tucker model.

By a similar token, if we allow column repetition in $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ for CPD, i.e., every column of $\mathbf{A}$ is repeated JK times, and we call the result $\mathbf{U}$; every column of $\mathbf{B}$ is repeated IK times, and we call the result $\mathbf{V}$; and every column of $\mathbf{C}$ is repeated IJ times, and we call the result $\mathbf{W}$, then it is possible to think of non-orthogonal Tucker as a special case of CPD – but notice that, due to column repetitions, this particular CPD model has k-ranks equal to one in all modes, and is therefore highly non-unique.

In a nutshell, both CPD and Tucker are sum-of-outer-products models, and one can argue that the most general form of one contains the other. What distinguishes the two is uniqueness, which is related but not tantamount to model parsimony ("minimality"); and modes of usage, which are quite different for the two models, as we will see.

B. MLSVD and approximation

By now the reader must have developed some familiarity with vectorization, and it should be clear that the Tucker model can be equivalently written in various useful ways, such as in vector form as

$$\mathbf{x} := \text{vec}(\mathbf{X}) = (\mathbf{U} \otimes \mathbf{V} \otimes \mathbf{W})\,\mathbf{g},$$

where $\mathbf{g} := \text{vec}(\mathbf{G})$, and the order of vectorization of $\mathbf{X}$ only affects the order in which the factor matrices $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$ appear in the Kronecker product chain, and of course the corresponding permutation of the elements of $\mathbf{g}$. From the properties of the Kronecker product, we know that the expression above is the result of vectorization of matrix

$$\mathbf{X}_1 = (\mathbf{V} \otimes \mathbf{W})\,\mathbf{G}_1\mathbf{U}^T$$

where the $KJ \times I$ matrix $\mathbf{X}_1$ contains all rows (mode-1 vectors) of tensor $\mathbf{X}$, and the $KJ \times I$ matrix $\mathbf{G}_1$ is a likewise reshaped form of the core tensor $\mathbf{G}$. From this expression it is evident that we can linearly transform the columns of $\mathbf{U}$ and absorb the inverse transformation in $\mathbf{G}_1$, i.e.,

$$\mathbf{G}_1\mathbf{U}^T = \mathbf{G}_1\mathbf{M}^{-T}\,(\mathbf{U}\mathbf{M})^T\,,$$

from which it follows immediately that the Tucker model is not unique. Recalling that $\mathbf{X}_1$ contains all rows of tensor $\mathbf{X}$, and letting r_1 denote the row-rank (mode-1 rank) of $\mathbf{X}$, it is clear that, without loss of generality, we can pick $\mathbf{U}$ to be an $I \times r_1$ orthonormal basis of the row-span of $\mathbf{X}$, and absorb the linear transformation in $\mathbf{G}$, which is thereby reduced from $I \times J \times K$ to $r_1 \times J \times K$. Continuing in this fashion with the other two modes, it follows that, without loss of generality, the Tucker model can be written as

$$\mathbf{x} := \text{vec}(\mathbf{X}) = (\mathbf{U}_{r_1} \otimes \mathbf{V}_{r_2} \otimes \mathbf{W}_{r_3})\,\mathbf{g},$$

where $\mathbf{U}_{r_1}$ is $I \times r_1$, $\mathbf{V}_{r_2}$ is $J \times r_2$, $\mathbf{W}_{r_3}$ is $K \times r_3$, and $\mathbf{g} := \text{vec}(\mathbf{G})$ is $r_1 r_2 r_3 \times 1$ – the vectorization of the $r_1 \times r_2 \times r_3$ reduced-size core tensor $\mathbf{G}$. This compact-size Tucker model is depicted in Fig. 6. Henceforth we drop the

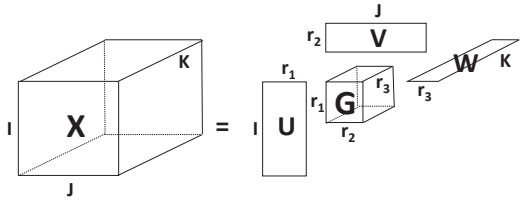

Fig. 6: Compact (reduced) Tucker model: r_1, r_2, r_3 are the mode (row, column, fiber, resp.) ranks of $\mathbf{X}$.

subscripts from $\mathbf{U}_{r_1}$, $\mathbf{V}_{r_2}$, $\mathbf{W}_{r_3}$ for brevity – the meaning will be clear from context. The Tucker model with orthonormal $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$ chosen as the right singular vectors of the matrix unfoldings $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{X}_3$, respectively, is also known as the multilinear SVD (MLSVD) (earlier called the higher-order SVD: HOSVD) [51], and it has several interesting and useful properties, as we will soon see.

It is easy to see that orthonormality of the columns of $\mathbf{U}_{r_1}$, $\mathbf{V}_{r_2}$, $\mathbf{W}_{r_3}$ implies orthonormality of the columns of their Kronecker product. This is because $(\mathbf{U}_{r_1} \otimes \mathbf{V}_{r_2})^T(\mathbf{U}_{r_1} \otimes \mathbf{V}_{r_2}) = (\mathbf{U}_{r_1}^T \otimes \mathbf{V}_{r_2}^T)(\mathbf{U}_{r_1} \otimes \mathbf{V}_{r_2}) = (\mathbf{U}_{r_1}^T\mathbf{U}_{r_1}) \otimes (\mathbf{V}_{r_2}^T\mathbf{V}_{r_2}) = \mathbf{I} \otimes \mathbf{I} = \mathbf{I}$. Recall that $\mathbf{x}_1 \perp \mathbf{x}_2 \iff \mathbf{x}_1^T\mathbf{x}_2 = 0 \implies ||\mathbf{x}_1 + \mathbf{x}_2||_2^2 = ||\mathbf{x}_1||_2^2 + ||\mathbf{x}_2||_2^2$. It follows that

$$||\mathbf{X}||_F^2 := \sum_{\forall\ i,j,k} |\mathbf{X}(i,j,k)|^2 = ||\mathbf{x}||_2^2 = ||\mathbf{g}||_2^2 = ||\mathbf{G}||_F^2,$$

where $\mathbf{x} = \text{vec}(\mathbf{X})$, and $\mathbf{g} = \text{vec}(\mathbf{G})$. It also follows that, if we drop certain outer products from the decomposition $\mathbf{x} = (\mathbf{U} \otimes \mathbf{V} \otimes \mathbf{W})\,\mathbf{g}$, or equivalently from (4), i.e., set the corresponding core elements to zero, then, by orthonormality

$$\left|\left|\mathbf{X} - \widehat{\mathbf{X}}\right|\right|_F^2 = \sum_{(\ell,m,n)\in\mathcal{D}} |\mathbf{G}(\ell,m,n)|^2,$$

where $\mathcal{D}$ is the set of dropped core element indices. So, if we order the elements of $\mathbf{G}$ in order of decreasing magnitude, and discard the "tail", then $\widehat{\mathbf{X}}$ will be close to $\mathbf{X}$, and we can quantify the error without having to reconstruct $\mathbf{X}$, take the difference and evaluate the norm.

In trying to generalize the matrix SVD, we are tempted to consider dropping entire columns of $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$. Notice that, for matrix SVD, this corresponds to zeroing out small singular values on the diagonal of matrix $\mathbf{\Sigma}$, and per the Eckart–Young theorem, it is optimal in that it yields the best low-rank approximation of the given matrix. Can we do the same for higher-order tensors?

First note that we can permute the slabs of $\mathbf{G}$ in any direction, and permute the corresponding columns of $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$ accordingly – this is evident from (4). In this way, we may bring the frontal slab with the highest energy $||\mathbf{G}(:,:,n)||_F^2$ up front, then the one with second highest energy, etc. Next, we can likewise order the lateral slabs of the core *without changing the energy of the frontal slabs*, and so on – and in this way, we can compact the energy of the core on its upper-left-front corner. We can then truncate the core, keeping only its upper-left-front dominant part of size $r_1^{'} \times r_2^{'} \times r_3^{'}$, with $r_1^{'} \leq r_1$, $r_2^{'} \leq r_2$, and $r_3^{'} \leq r_3$. The resulting approximation error can be readily bounded as

$$\left|\left|\mathbf{X} - \widehat{\mathbf{X}}\right|\right|_F^2 \leq \sum_{\ell=r_1^{'}+1}^{r_1} ||\mathbf{G}(\ell,:,:)||_F^2 + \sum_{m=r_2^{'}+1}^{r_2} ||\mathbf{G}(:,m,:)||_F^2 + \sum_{n=r_3^{'}+1}^{r_3} ||\mathbf{G}(:,:,n)||_F^2,$$

where we use $\leq$ as opposed to $=$ because dropped elements may be counted up to three times (in particular, the lower-right-back ones). One can of course compute the exact error of such a truncation strategy, but this involves instantiating $\mathbf{X} - \widehat{\mathbf{X}}$.

Either way, such truncation in general *does not* yield the best approximation of $\mathbf{X}$ for the given $(r_1^{'}, r_2^{'}, r_3^{'})$. That is, there is no exact equivalent of the Eckart–Young theorem for tensors of order higher than two [52] – in fact, as we will see later, the best low multilinear rank approximation problem for tensors is NP-hard. Despite this "bummer", much of the beauty of matrix SVD remains in MLSVD, as explained next. In particular, the slabs of the core array $\mathbf{G}$ along each mode are orthogonal to each other, i.e., $(\text{vec}(\mathbf{G}(\ell,:,:)))^T\text{vec}(\mathbf{G}(\ell^{'},:,:)) = 0$ for $\ell^{'} \neq \ell$, and $||\mathbf{G}(\ell,:,:)||_F$ equals the ℓ-th singular value of $\mathbf{X}_1$; and similarly for the other modes (we will actually prove a more general result very soon). These orthogonality and Frobenius norm properties of the Tucker core array generalize a property of matrix SVD: namely, the "core matrix" of singular values $\mathbf{\Sigma}$ in matrix SVD is diagonal, which implies that its rows are orthogonal to each other, and the same is true for its columns. Diagonality thus implies orthogonality of one-lower-order slabs (sub-tensors of order one less than the original tensor), but the converse is not true, e.g., consider

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

We have seen that diagonality of the core is not possible in general for higher-order tensors, because it severely limits the degrees of freedom; but all-orthogonality of one-lower-order slabs of the core array, and the interpretation of their Frobenius norms as singular values of a certain matrix view of the tensor come without loss of generality (or optimality, as we will see in the proof of the next property). This intuitively pleasing result was pointed out by De Lathauwer [51], and it largely motivates the analogy to matrix SVD – albeit simply truncating slabs (or elements) of the full core will not give the best low multilinear rank approximation of $\mathbf{X}$ in the case of three- and higher-order tensors. The error bound above is actually the proper generalization of the Eckart–Young theorem. In the matrix case, because of diagonality there is only one summation and equality instead of inequality.

Simply truncating the MLSVD at sufficiently high $(r_1^{'}, r_2^{'}, r_3^{'})$ is often enough to obtain a good approximation in practice – we may control the error as we wish, so long as we pick high enough $(r_1^{'}, r_2^{'}, r_3^{'})$. The error $||\mathbf{X} - \widehat{\mathbf{X}}||_F^2$ is in fact at most 3 times higher than the minimal error (N times higher in the N-th order case) [16], [17]. If we are interested in the best possible approximation of $\mathbf{X}$ with mode ranks

(r_1', r_2', r_3'), however, then we need to consider the following, after dropping the $'$s for brevity:

Property 1. *[51], [53] Let* $\left(\widehat{\mathbf{U}}, \widehat{\mathbf{V}}, \widehat{\mathbf{W}}, \widehat{\mathbf{G}}_1\right)$ *be a solution to*

$$\begin{aligned} \min_{(\mathbf{U},\mathbf{V},\mathbf{W},\mathbf{G}_1)} \quad & ||\mathbf{X}_1 - (\mathbf{V}\otimes\mathbf{W})\mathbf{G}_1\mathbf{U}^T||_F^2 \\ \textit{such that:} \quad & \mathbf{U}: I \times r_1,\ r_1 \leq I,\ \mathbf{U}^T\mathbf{U} = \mathbf{I} \\ & \mathbf{V}: J \times r_2,\ r_2 \leq J,\ \mathbf{V}^T\mathbf{V} = \mathbf{I} \\ & \mathbf{W}: K \times r_3,\ r_3 \leq K,\ \mathbf{W}^T\mathbf{W} = \mathbf{I} \\ & \mathbf{G}_1: r_3 r_2 \times r_1 \end{aligned}$$

Then

- $\widehat{\mathbf{G}}_1 = (\widehat{\mathbf{V}} \otimes \widehat{\mathbf{W}})^T\mathbf{X}_1\widehat{\mathbf{U}}$*;*
- *Substituting the conditionally optimal* $\mathbf{G}_1$*, the problem can be recast in "concentrated" form as*

$$\begin{aligned} \max_{(\mathbf{U},\mathbf{V},\mathbf{W})} \quad & ||(\mathbf{V}\otimes\mathbf{W})^T\mathbf{X}_1\mathbf{U}||_F^2 \\ \textit{such that:} \quad & \mathbf{U}: I \times r_1,\ r_1 \leq I,\ \mathbf{U}^T\mathbf{U} = \mathbf{I} \\ & \mathbf{V}: J \times r_2,\ r_2 \leq J,\ \mathbf{V}^T\mathbf{V} = \mathbf{I} \\ & \mathbf{W}: K \times r_3,\ r_3 \leq K,\ \mathbf{W}^T\mathbf{W} = \mathbf{I} \end{aligned}$$

- $\widehat{\mathbf{U}}$ *= dominant* r_1*-dim. right subspace of* $(\widehat{\mathbf{V}} \otimes \widehat{\mathbf{W}})^T\mathbf{X}_1$*;*
- $\widehat{\mathbf{V}}$ *= dominant* r_2*-dim. right subspace of* $(\widehat{\mathbf{U}} \otimes \widehat{\mathbf{W}})^T\mathbf{X}_2$*;*
- $\widehat{\mathbf{W}}$ *= dominant* r_3*-dim. right subspace of* $(\widehat{\mathbf{U}} \otimes \widehat{\mathbf{V}})^T\mathbf{X}_3$*;*
- $\widehat{\mathbf{G}}_1$ *has orthogonal columns; and*
- $\left\{||\widehat{\mathbf{G}}_1(:,m)||_2^2\right\}_{m=1}^{r_1}$ *are the* r_1 *principal singular values of* $(\widehat{\mathbf{V}} \otimes \widehat{\mathbf{W}})^T\mathbf{X}_1$*. Note that each column of* $\widehat{\mathbf{G}}_1$ *is a vectorized slab of the core array* $\widehat{\mathbf{G}}$ *obtained by fixing the first reduced dimension index to some value.*

Proof. Note that $||\text{vec}(\mathbf{X}_1) - (\mathbf{U}\otimes\mathbf{V}\otimes\mathbf{W})\,\text{vec}(\mathbf{G}_1)||_2^2 = ||\mathbf{X}_1 - (\mathbf{V}\otimes\mathbf{W})\mathbf{G}_1\mathbf{U}^T||_F^2$, so conditioned on (orthonormal) $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$ the optimal $\mathbf{G}$ is given by $\text{vec}(\widehat{\mathbf{G}}_1) = (\mathbf{U}\otimes\mathbf{V}\otimes\mathbf{W})^T\,\text{vec}(\mathbf{X}_1)$, and therefore $\widehat{\mathbf{G}}_1 = (\mathbf{V}\otimes\mathbf{W})^T\mathbf{X}_1\mathbf{U}$.

Consider $||\mathbf{X}_1 - (\mathbf{V}\otimes\mathbf{W})\mathbf{G}_1\mathbf{U}^T||_F^2$, define $\widetilde{\mathbf{X}}_1 := (\mathbf{V}\otimes\mathbf{W})\mathbf{G}_1\mathbf{U}^T$, and use that $||\mathbf{X}_1 - \widetilde{\mathbf{X}}_1||_F^2 = \text{Tr}((\mathbf{X}_1 - \widetilde{\mathbf{X}}_1)^T(\mathbf{X}_1 - \widetilde{\mathbf{X}}_1)) = ||\mathbf{X}_1||_F^2 + ||\widetilde{\mathbf{X}}_1||_F^2 - 2\text{Tr}(\mathbf{X}_1^T\widetilde{\mathbf{X}}_1)$. By orthonormality of $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$, it follows that $||\widetilde{\mathbf{X}}_1||_F^2 = ||\mathbf{G}_1||_F^2$. Now, consider

$$-2\text{Tr}(\mathbf{X}_1^T\widetilde{\mathbf{X}}_1) = -2\text{Tr}(\mathbf{X}_1^T(\mathbf{V}\otimes\mathbf{W})\mathbf{G}_1\mathbf{U}^T),$$

and substitute $\mathbf{G}_1 = (\mathbf{V}\otimes\mathbf{W})^T\mathbf{X}_1\mathbf{U}$ to obtain

$$-2\text{Tr}(\mathbf{X}_1^T(\mathbf{V}\otimes\mathbf{W})(\mathbf{V}\otimes\mathbf{W})^T\mathbf{X}_1\mathbf{U}\mathbf{U}^T).$$

Using a property of the trace operator to bring the rightmost matrix to the left, we obtain

$$-2\text{Tr}(\mathbf{U}^T\mathbf{X}_1^T(\mathbf{V}\otimes\mathbf{W})(\mathbf{V}\otimes\mathbf{W})^T\mathbf{X}_1\mathbf{U}) =$$
$$-2\text{Tr}(\mathbf{G}_1^T\mathbf{G}_1) = -2||\mathbf{G}_1||_F^2.$$

It follows that $||\mathbf{X}_1 - (\mathbf{V}\otimes\mathbf{W})\mathbf{G}_1\mathbf{U}^T||_F^2 = ||\mathbf{X}_1||_F^2 - ||\mathbf{G}_1||_F^2$, so we may equivalently maximize $||\mathbf{G}_1||_F^2 = ||(\mathbf{V}\otimes\mathbf{W})^T\mathbf{X}_1\mathbf{U}||_F^2$. From this, it immediately follows that $\widehat{\mathbf{U}}$ is the dominant right subspace of $(\widehat{\mathbf{V}}\otimes\widehat{\mathbf{W}})^T\mathbf{X}_1$, so we can take it to be the r_1 principal right singular vectors of $(\widehat{\mathbf{V}}\otimes\widehat{\mathbf{W}})^T\mathbf{X}_1$. The respective results for $\widehat{\mathbf{V}}$ and $\widehat{\mathbf{W}}$ are obtained by appealing to role symmetry. Next, we show that $\widehat{\mathbf{G}}_1$ has orthogonal columns. To see this, let $\widehat{\mathbf{G}}_1 = [\widehat{\mathbf{g}}_{1,1}, \cdots, \widehat{\mathbf{g}}_{1,r_1}]$, and $\widehat{\mathbf{U}} = [\widehat{\mathbf{u}}_1, \cdots, \widehat{\mathbf{u}}_{r_1}]$. Consider

$$\widehat{\mathbf{g}}_{1,m_1}^T\widehat{\mathbf{g}}_{1,m_2} = \widehat{\mathbf{u}}_{m_1}^T\mathbf{X}_1^T(\widehat{\mathbf{V}}\otimes\widehat{\mathbf{W}})(\widehat{\mathbf{V}}\otimes\widehat{\mathbf{W}})^T\mathbf{X}_1\widehat{\mathbf{u}}_{m_2}.$$

Let $\mathbf{\Gamma\Sigma\tilde{U}}^T$ be the SVD of $(\widehat{\mathbf{V}}\otimes\widehat{\mathbf{W}})^T\mathbf{X}_1$. Then $\tilde{\mathbf{U}} = [\widehat{\mathbf{U}},\ \check{\mathbf{U}}]$, so

$$(\widehat{\mathbf{V}}\otimes\widehat{\mathbf{W}})^T\mathbf{X}_1\mathbf{u}_{m_2} = \boldsymbol{\gamma}_{m_2}\sigma_{m_2},$$

with obvious notation; and therefore

$$\widehat{\mathbf{g}}_{1,m_1}^T\widehat{\mathbf{g}}_{1,m_2} = \sigma_{m_1}\sigma_{m_2}\boldsymbol{\gamma}_{m_1}^T\boldsymbol{\gamma}_{m_2} = \sigma_{m_1}\sigma_{m_2}\delta(m_1 - m_2),$$

by virtue of orthonormality of left singular vectors of $(\widehat{\mathbf{V}}\otimes\widehat{\mathbf{W}})^T\mathbf{X}_1$ (here $\delta(\cdot)$ is the Kronecker delta). By role symmetry, it follows that the slabs of $\widehat{\mathbf{G}}$ along any mode are likewise orthogonal. It is worth mentioning that, as a byproduct of the last equation, $||\widehat{\mathbf{G}}(:,:,m)||_F^2 = ||\widehat{\mathbf{G}}_1(:,m)||_2^2 = ||\widehat{\mathbf{g}}_{1,m}||_2^2 = \sigma_m^2$; that is, *the Frobenius norms of the lateral core slabs are the* r_1 *principal singular values of* $(\widehat{\mathbf{V}}\otimes\widehat{\mathbf{W}})^T\mathbf{X}_1$. □

The best rank-1 tensor approximation problem over $\mathbb{R}$ is NP-hard [54, Theorem 1.13], so the best low multilinear rank approximation problem is also NP-hard (the best multilinear rank approximation with $(r_1, r_2, r_3) = (1, 1, 1)$ is the best rank-1 approximation). This is reflected in a key limitation of the characterization in Property 1, which gives explicit expressions that relate the sought $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$, and $\mathbf{G}$, but it does not provide an explicit solution for any of them. On the other hand, Property 1 naturally suggests the following alternating least squares scheme:

$\perp$-Tucker ALS

1) Initialize:
   - $\mathbf{U} = r_1$ principal right singular vectors of $\mathbf{X}_1$;
   - $\mathbf{V} = r_2$ principal right singular vectors of $\mathbf{X}_2$;
   - $\mathbf{W} = r_3$ principal right singular vectors of $\mathbf{X}_3$;
2) repeat:
   - $\mathbf{U} = r_1$ principal right sing. vec. of $(\mathbf{V}\otimes\mathbf{W})^T\mathbf{X}_1$;
   - $\mathbf{V} = r_2$ principal right sing. vec. of $(\mathbf{U}\otimes\mathbf{W})^T\mathbf{X}_2$;
   - $\mathbf{W} = r_3$ principal right sing. vec. of $(\mathbf{U}\otimes\mathbf{V})^T\mathbf{X}_3$;
   - until negligible change in $||(\mathbf{V}\otimes\mathbf{W})^T\mathbf{X}_1\mathbf{U}||_F^2$.
3) $\mathbf{G}_1 = (\mathbf{V}\otimes\mathbf{W})^T\mathbf{X}_1\mathbf{U}$.

The initialization in step 1) [together with step 3)] corresponds to (truncated) MLSVD. It is not necessarily optimal, as previously noted, but it does help as a very good initialization in most cases. The other point worth noting is that each variable update is optimal conditioned on the rest of the variables, so the reward $||(\mathbf{V}\otimes\mathbf{W})^T\mathbf{X}_1\mathbf{U}||_F^2$ is non-decreasing (equivalently, the cost $||\mathbf{X}_1 - (\mathbf{V}\otimes\mathbf{W})\mathbf{G}_1\mathbf{U}^T||_F^2$ is non-increasing) and bounded from above (resp. below), thus convergence of the reward (cost) sequence is guaranteed. Note the conceptual similarity of the above algorithm with ALS for CPD, which we discussed earlier. The first variant of Tucker-ALS goes back to the work of Kroonenberg and De Leeuw; see [55] and references therein.

Note that using MLSVD with somewhat higher (r_1, r_2, r_3) can be computationally preferable to ALS. In the case of big

data, even the computation of MLSVD may be prohibitive, and randomized projection approaches become more appealing [56], [57]. A drastically different approach is to draw the columns of $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$ from the columns, rows, fibers of $\mathbf{X}$ [58]–[60]. Although the idea is simple, it has sound algebraic foundations and error bounds are available [61]. Finally, we mention that for large-scale matrix problems Krylov subspace methods are one of the main classes of algorithms. They only need an implementation of the matrix-vector product to iteratively find subspaces on which to project. See [62] for Tucker-type extensions.

C. Compression as preprocessing

Consider a tensor $\mathbf{X}$ in vectorized form, and corresponding CPD and orthogonal Tucker ($\perp$-Tucker) models

$$\mathbf{x} = (\mathbf{A} \odot \mathbf{B} \odot \mathbf{C})\mathbf{1} = (\mathbf{U} \otimes \mathbf{V} \otimes \mathbf{W})\mathbf{g}.$$

Pre-multiplying with $(\mathbf{U} \otimes \mathbf{V} \otimes \mathbf{W})^T = (\mathbf{U}^T \otimes \mathbf{V}^T \otimes \mathbf{W}^T)$ and using the mixed-product rule for $\otimes$, $\odot$, we obtain

$$\mathbf{g} = \left((\mathbf{U}^T\mathbf{A}) \odot (\mathbf{V}^T\mathbf{B}) \odot (\mathbf{W}^T\mathbf{C})\right)\mathbf{1},$$

i.e., the Tucker core array $\mathbf{G}$ (shown above in vectorized form $\mathbf{g}$) admits a CPD decomposition of $\text{rank}(\mathbf{G}) \leq \text{rank}(\mathbf{X})$. Let $\left[\!\left[\tilde{\mathbf{A}}, \tilde{\mathbf{B}}, \tilde{\mathbf{C}}\right]\!\right]$ be a CPD of $\mathbf{G}$, i.e., $\mathbf{g} = (\tilde{\mathbf{A}} \odot \tilde{\mathbf{B}} \odot \tilde{\mathbf{C}})\mathbf{1}$. Then

$$\mathbf{x} = (\mathbf{U} \otimes \mathbf{V} \otimes \mathbf{W})\mathbf{g} = (\mathbf{U} \otimes \mathbf{V} \otimes \mathbf{W})(\tilde{\mathbf{A}} \odot \tilde{\mathbf{B}} \odot \tilde{\mathbf{C}})\mathbf{1} =$$
$$= \left((\mathbf{U}\tilde{\mathbf{A}}) \odot (\mathbf{V}\tilde{\mathbf{B}}) \odot (\mathbf{W}\tilde{\mathbf{C}})\right)\mathbf{1},$$

by the mixed product rule. Assuming that the CPD of $\mathbf{X}$ is essentially unique, it then follows that

$$\mathbf{A} = \mathbf{U}\tilde{\mathbf{A}}\mathbf{\Pi}\mathbf{\Lambda}_a,\ \mathbf{B} = \mathbf{V}\tilde{\mathbf{B}}\mathbf{\Pi}\mathbf{\Lambda}_b,\ \mathbf{C} = \mathbf{W}\tilde{\mathbf{C}}\mathbf{\Pi}\mathbf{\Lambda}_c,$$

where $\mathbf{\Pi}$ is a permutation matrix and $\mathbf{\Lambda}_a\mathbf{\Lambda}_b\mathbf{\Lambda}_c = \mathbf{I}$. It follows that

$$\mathbf{U}^T\mathbf{A} = \tilde{\mathbf{A}}\mathbf{\Pi}\mathbf{\Lambda}_a,\ \mathbf{V}^T\mathbf{B} = \tilde{\mathbf{B}}\mathbf{\Pi}\mathbf{\Lambda}_b,\ \mathbf{W}^T\mathbf{C} = \tilde{\mathbf{C}}\mathbf{\Pi}\mathbf{\Lambda}_c,$$

so that the CPD of $\mathbf{G}$ is essentially unique, and therefore $\text{rank}(\mathbf{G}) = \text{rank}(\mathbf{X})$.

Since the size of $\mathbf{G}$ is smaller than or equal to the size of $\mathbf{X}$, this suggests that an attractive way to compute the CPD of $\mathbf{X}$ is to first compress (using one of the orthogonal schemes in the previous subsection), compute the CPD of $\mathbf{G}$, and then "blow-up" the resulting factors, since $\mathbf{A} = \mathbf{U}\tilde{\mathbf{A}}$ (up to column permutation and scaling). It also shows that $\mathbf{A} = \mathbf{U}\mathbf{U}^T\mathbf{A}$, and likewise for the other two modes. The caveat is that the discussion above assumes *exact* CPD and $\perp$-Tucker models, whereas in reality we are interested in low-rank least-squares approximation – for this, we refer the reader to the *Candelinc* theorem of Carroll *et al.* [63]; see also Bro & Andersson [64].

This does not work for a constrained CPD (e.g. one or more factor matrices nonnegative, monotonic, sparse, ...) since the orthogonal compression destroys the constraints. In the ALS approach we can still exploit multi-linearity, however, to update $\mathbf{U}$ by solving a *constrained* and/or regularized linear least squares problem, and similarly for $\mathbf{V}$ and $\mathbf{W}$, by role symmetry. For $\mathbf{G}$, we can use the vectorization property of the Kronecker product to bring it to the right, and then use a constrained or regularized linear least squares solver. By the mixed product rule, this last step entails pseudo-inversion of the $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$ matrices, instead of their (much larger) Kronecker product. This type of model is sometimes called *oblique* Tucker, to distinguish from *orthogonal* Tucker. More generally than in ALS (see the algorithms in Section VII), one can fit the constrained CPD in the uncompressed space, but with $\mathbf{X}$ replaced by its parameter-efficient factorized representation. The structure of the latter may then be exploited to reduce the per iteration complexity [65].

VI. Other decompositions

A. Compression

In Section V we have emphasized the use of $\perp$-Tucker/MLSVD for tensor approximation and compression. This use was in fact limited to tensors of moderate order. Let us consider the situation at order N and let us assume for simplicity that $r_1 = r_2 = \ldots = r_N = r > 1$. Then the core tensor $\mathbf{G}$ has r^N entries. The exponential dependence of the number of entries on the tensor order N is called the Curse of Dimensionality: in the case of large N (e.g. $N = 100$), r^N is large, even when r is small, and as a result $\perp$-Tucker/MLSVD cannot be used. In such cases one may resort to a Tensor Train (TT) representation or a hierarchical Tucker (hTucker) decomposition instead [16], [66]. A TT of an N-th order tensor $\mathbf{X}$ is of the form

$$\mathbf{X}(i_1, i_2, \ldots, i_N) = \sum_{r_1 r_2 \ldots r_{N-1}} u^{(1)}_{i_1 r_1} u^{(2)}_{r_1 i_2 r_2} u^{(3)}_{r_2 i_3 r_3} \ldots u^{(N)}_{i_N r_{N-1}}, \tag{5}$$

in which one can see $\mathbf{U}^{(1)}$ as the locomotive and the next factors as the carriages. Note that each carriage "transports" one tensor dimension, and that two consecutive carriages are connected through the summation over one common index. Since every index appears at most twice and since there are no index cycles, the TT-format is "matrix-like", i.e. a TT approximation can be computed using established techniques from numerical *linear* algebra, similarly to MLSVD. Like for MLSVD, fiber sampling schemes have been developed too. On the other hand, the number of entries is now $O(NIr^2)$, so the Curse of Dimensionality has been broken. hTucker is the extension in which the indices are organized in a binary tree.

B. Analysis

In Section IV we have emphasized the uniqueness of CPD under mild conditions as a profound advantage of tensors over matrices in the context of signal separation and data analysis – constraints such as orthogonality or triangularity are not necessary per se. An even more profound advantage is the possibility to have a unique decomposition in terms that are not even rank-1. Block Term Decompositions (BTD) write a tensor as a sum of terms that have low multilinear rank, i.e. the terms can be pictured as in Fig. 6 rather than as in Fig. 1 [67], [68]. Note that rank-1 structure of data components is indeed an assumption that needs to be justified.

As in CPD, uniqueness of a BTD is up to a permutation of the terms. The scaling/counterscaling ambiguities within

a rank-1 term generalize to the indeterminacies in a Tucker representation. Expanding the block terms into sums of rank-1 terms with repeated vectors as in (4) yields a form that is known as PARALIND [69]; see also more recent results in [70]–[72].

C. Fusion

Multiple data sets may be jointly analyzed by means of coupled decompositions of several matrices and/or tensors, possibly of different size [73], [74]. An early variant, in which coupling was imposed through a shared covariance matrix, is Harshman's PARAFAC2 [75]. In a coupled setting, particular decompositions may inherit uniqueness from other decompositions; in particular, the decomposition of a data matrix may become unique thanks to coupling [76].

VII. Algorithms

A. ALS: Computational aspects

1) CPD: We now return to the basic ALS algorithm for CPD, to discuss (efficient) computation issues. First note that the pseudo-inverse that comes into play in ALS updates is structured: in updating $\mathbf{C}$, for example

$$\mathbf{C} \leftarrow \arg\min_{\mathbf{C}} ||\mathbf{X}_3 - (\mathbf{B} \odot \mathbf{A})\mathbf{C}^T||_F^2,$$

the pseudo-inverse

$$(\mathbf{B} \odot \mathbf{A})^\dagger = \left[(\mathbf{B} \odot \mathbf{A})^T(\mathbf{B} \odot \mathbf{A})\right]^{-1}(\mathbf{B} \odot \mathbf{A})^T,$$

can be simplified. In particular,

$$(\mathbf{B} \odot \mathbf{A})^T(\mathbf{B} \odot \mathbf{A}) = \begin{bmatrix} \mathbf{A}\mathbf{D}_1(\mathbf{B}) \\ \vdots \\ \mathbf{A}\mathbf{D}_J(\mathbf{B}) \end{bmatrix}^T \begin{bmatrix} \mathbf{A}\mathbf{D}_1(\mathbf{B}) \\ \vdots \\ \mathbf{A}\mathbf{D}_J(\mathbf{B}) \end{bmatrix}$$

$$= \sum_{j=1}^{J} \mathbf{D}_j(\mathbf{B})\mathbf{A}^T\mathbf{A}\mathbf{D}_j(\mathbf{B}),$$

where we note that the result is $F \times F$, and element (f_1, f_2) of the result is element (f_1, f_2) of $\mathbf{A}^T\mathbf{A}$ times $\sum_{j=1}^{J} \mathbf{B}(j, f_1)\mathbf{B}(j, f_2)$. The latter is element (f_1, f_2) of $\mathbf{B}^T\mathbf{B}$. It follows that

$$(\mathbf{B} \odot \mathbf{A})^T(\mathbf{B} \odot \mathbf{A}) = (\mathbf{B}^T\mathbf{B}) * (\mathbf{A}^T\mathbf{A}),$$

which only involves the Hadamard product of $F \times F$ matrices, and is easy to invert for small ranks F (but note that in the case of big sparse data, small F may not be enough). Thus the update of $\mathbf{C}$ can be performed as

$$\mathbf{C}^T \leftarrow \left((\mathbf{B}^T\mathbf{B}) * (\mathbf{A}^T\mathbf{A})\right)^{-1}(\mathbf{B} \odot \mathbf{A})^T\mathbf{X}_3.$$

For small F, the bottleneck of this is actually the computation of $(\mathbf{B} \odot \mathbf{A})^T\mathbf{X}_3$ – notice that $\mathbf{B} \odot \mathbf{A}$ is $IJ \times F$, and $\mathbf{X}_3$ is $IJ \times K$. Brute-force computation of $(\mathbf{B} \odot \mathbf{A})^T\mathbf{X}_3$ thus demands IJF additional memory and flops to instantiate $\mathbf{B} \odot \mathbf{A}$, even though the result is only $F \times K$, and $IJKF$ flops to actually compute the product – but see [77]–[79]. If $\mathbf{X}$ (and therefore $\mathbf{X}_3$) is sparse, having NNZ($\mathbf{X}$) nonzero elements stored in a [i,j,k,value] list, then every nonzero element multiplies a column of $(\mathbf{B} \odot \mathbf{A})^T$, and the result should be added to column k. The specific column needed can be generated on-the-fly with $F+1$ flops, for an overall complexity of $(2F+1)$NNZ($\mathbf{X}$), without requiring any additional memory (other than that needed to store the running estimates of $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and the data $\mathbf{X}$). When $\mathbf{X}$ is dense, the number of flops is inevitably of order $IJKF$, but still no additional memory is needed this way. Furthermore, the computation can be parallelized in several ways – see [77], [80]–[83] for various resource-efficient algorithms for *matricized tensor times Khatri–Rao product* (MTTKRP) computations.

2) Tucker: For $\perp$-Tucker ALS, we need to compute products of type $(\mathbf{V} \otimes \mathbf{W})^T\mathbf{X}_1$ (and then compute the principal right singular vectors of the resulting $r_2r_3 \times I$ matrix). The column-generation idea can be used here as well to avoid intermediate memory explosion and exploit sparsity in $\mathbf{X}$ when computing $(\mathbf{V} \otimes \mathbf{W})^T\mathbf{X}_1$.

For oblique Tucker ALS we need to compute $\left((\mathbf{V} \otimes \mathbf{W})\mathbf{G}_1\right)^\dagger \mathbf{X}_1$ for updating $\mathbf{U}$, and $\left(\mathbf{U}^\dagger \otimes \mathbf{V}^\dagger \otimes \mathbf{W}^\dagger\right)\mathbf{x}$ for updating $\mathbf{g} \leftrightarrow \mathbf{G}$. The latter requires pseudo-inverses of relatively small matrices, but note that

$$\left((\mathbf{V} \otimes \mathbf{W})\mathbf{G}_1\right)^\dagger \neq \mathbf{G}_1^\dagger\left(\mathbf{V} \otimes \mathbf{W}\right)^\dagger,$$

in general. Equality holds if $\mathbf{V} \otimes \mathbf{W}$ is full column rank *and* $\mathbf{G}_1$ is full row rank, which requires $r_2r_3 \leq r_1$.

ALS is a special case of *block coordinate descent* (BCD), in which the subproblems take the form of linear LS estimation. As the musings in [84] make clear, understanding the convergence properties of ALS is highly nontrivial. ALS monotonically reduces the cost function, but it is not guaranteed to converge to a stationary point. A conceptually easy fix is to choose for the next update the parameter block that decreases the cost function the most – this *maximum block improvement* (MBI) variant is guaranteed to converge under some conditions [85]. However, in the case of third-order CPD MBI doubles the computation time as two possible updates have to be compared. At order N, the computation time increases by a factor $N-1$ – and in practice there is usually little difference between MBI and plain ALS. Another way to ensure convergence of ALS is to include proximal regularization terms and invoke the *block successive upper bound minimization* (BSUM) framework of [86], which also helps in ill-conditioned cases. In cases where ALS converges, it does so at a local linear rate (under some non-degeneracy condition), which makes it (locally) slower than some derivative-based algorithms [87], [88], see further. The same is true for MBI [85].

B. Gradient descent

Consider the squared loss

$$\mathcal{L}(\mathbf{A}, \mathbf{B}, \mathbf{C}) := ||\mathbf{X}_1 - (\mathbf{C} \odot \mathbf{B})\mathbf{A}^T||_F^2 =$$

$$\text{tr}\left((\mathbf{X}_1 - (\mathbf{C} \odot \mathbf{B})\mathbf{A}^T)^T(\mathbf{X}_1 - (\mathbf{C} \odot \mathbf{B})\mathbf{A}^T)\right) = ||\mathbf{X}_1||_F^2 -$$

$$2\ \text{tr}\left(\mathbf{X}_1^T(\mathbf{C} \odot \mathbf{B})\mathbf{A}^T\right) + \text{tr}\left(\mathbf{A}(\mathbf{C} \odot \mathbf{B})^T(\mathbf{C} \odot \mathbf{B})\mathbf{A}^T\right).$$

Recall that $(\mathbf{C} \odot \mathbf{B})^T(\mathbf{C} \odot \mathbf{B}) = (\mathbf{C}^T\mathbf{C}) * (\mathbf{B}^T\mathbf{B})$, so we may equivalently take the gradient of $-2\ \text{tr}\left(\mathbf{X}_1^T(\mathbf{C} \odot \mathbf{B})\mathbf{A}^T\right) +$

$\text{tr}\left(\mathbf{A}(\mathbf{C}^T\mathbf{C}) * (\mathbf{B}^T\mathbf{B})\mathbf{A}^T\right)$. Arranging the gradient in the same format[11] as $\mathbf{A}$, we have

$$\begin{aligned}\frac{\partial \mathcal{L}(\mathbf{A},\mathbf{B},\mathbf{C})}{\partial \mathbf{A}} &= -2\mathbf{X}_1^T(\mathbf{C}\odot\mathbf{B}) + 2\ \mathbf{A}\left[(\mathbf{C}^T\mathbf{C}) * (\mathbf{B}^T\mathbf{B})\right] \\ &= -2\left(\mathbf{X}_1^T - \mathbf{A}(\mathbf{C}\odot\mathbf{B})^T\right)(\mathbf{C}\odot\mathbf{B}),\end{aligned}$$

Appealing to role symmetry, we likewise obtain

$$\begin{aligned}\frac{\partial \mathcal{L}(\mathbf{A},\mathbf{B},\mathbf{C})}{\partial \mathbf{B}} &= -2\mathbf{X}_2^T(\mathbf{C}\odot\mathbf{A}) + 2\ \mathbf{B}\left[(\mathbf{C}^T\mathbf{C}) * (\mathbf{A}^T\mathbf{A})\right] \\ &= -2\left(\mathbf{X}_2^T - \mathbf{B}(\mathbf{C}\odot\mathbf{A})^T\right)(\mathbf{C}\odot\mathbf{A}), \\ \frac{\partial \mathcal{L}(\mathbf{A},\mathbf{B},\mathbf{C})}{\partial \mathbf{C}} &= -2\mathbf{X}_3^T(\mathbf{B}\odot\mathbf{A}) + 2\ \mathbf{C}\left[(\mathbf{B}^T\mathbf{B}) * (\mathbf{A}^T\mathbf{A})\right] \\ &= -2\left(\mathbf{X}_3^T - \mathbf{C}(\mathbf{B}\odot\mathbf{A})^T\right)(\mathbf{B}\odot\mathbf{A}).\end{aligned}$$

Remark 3. *The conditional least squares update for* $\mathbf{A}$ *is*

$$\mathbf{A} \leftarrow \left[(\mathbf{C}^T\mathbf{C}) * (\mathbf{B}^T\mathbf{B})\right]^{-1}(\mathbf{C}\odot\mathbf{B})^T\mathbf{X}_1,$$

So taking a gradient step or solving the least-squares subproblem to (conditional) optimality involves computing the same quantities: $(\mathbf{C}^T\mathbf{C}) * (\mathbf{B}^T\mathbf{B})$ *and* $(\mathbf{C}\odot\mathbf{B})^T\mathbf{X}_1$*. The only difference is that to take a gradient step you don't need to invert the* $F \times F$ *matrix* $(\mathbf{C}^T\mathbf{C}) * (\mathbf{B}^T\mathbf{B})$*. For small* F*, this inversion has negligible cost relative to the computation of the MTTKRP* $(\mathbf{C}\odot\mathbf{B})^T\mathbf{X}_1$*. Efficient algorithms for the MTTKRP can be used for gradient computations as well; but note that, for small* F*, each gradient step is essentially as expensive as an ALS step. Also note that, whereas it appears that keeping three different matricized copies of* $\mathbf{X}$ *is necessary for efficient gradient (and ALS) computations, only one is needed – see [81], [89].*

With these gradient expressions at hand, we can employ any gradient-based algorithm for model fitting.

C. Quasi-Newton and Nonlinear Least Squares

The well-known Newton descent algorithm uses a local quadratic approximation of the cost function $\mathcal{L}(\mathbf{A},\mathbf{B},\mathbf{C})$ to obtain a new step as the solution of the set of linear equations

$$\mathbf{H}\mathbf{p} = -\mathbf{g}, \tag{6}$$

in which $\mathbf{g}$ and $\mathbf{H}$ are the gradient and Hessian of $\mathcal{L}$, respectively. As computation of the Hessian may be prohibitively expensive, one may resort to an approximation, leading to quasi-Newton and Nonlinear Least Squares (NLS). Quasi-Newton methods such as Nonlinear Conjugate Gradients (NCG) and (limited memory) BFGS use a diagonal plus low-rank matrix approximation of the Hessian. In combination with line search or trust region globalization strategies for step size selection, quasi-Newton does guarantee convergence to a stationary point, contrary to plain ALS, and its convergence is superlinear [89], [90].

NLS methods such as Gauss–Newton and Levenberg–Marquardt start from a local linear approximation of the residual $\mathbf{X}_1 - (\mathbf{C}\odot\mathbf{B})\mathbf{A}^T$ to approximate the Hessian as $\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta})^T\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta})$, with $\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta})$ the Jacobian matrix of $\boldsymbol{\varphi}(\boldsymbol{\theta})$ (where $\boldsymbol{\theta}$ is the parameter vector; see section VIII for definitions of $\boldsymbol{\varphi}(\boldsymbol{\theta})$, and $\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta})$). The algebraic structure of $\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta})^T\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta})$ can be exploited to obtain a fast inexact NLS algorithm that has several favorable properties [89], [91]. Briefly, the inexact NLS algorithm uses a "parallel version" of one ALS iteration as a preconditioner for solving the linear system of equations (6). (In this parallel version the factor matrices are updated all together starting from the estimates in the previous iteration; note that the preconditioning can hence be parallelized.) After preconditioning, (6) is solved inexactly by a truncated conjugate gradient algorithm. That is, the set of equations is not solved exactly and neither is the matrix $\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta})^T\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta})$ computed or stored. Storage of $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{A}^T\mathbf{A}$, $\mathbf{B}^T\mathbf{B}$, $\mathbf{C}^T\mathbf{C}$ suffices for an efficient computation of the product of a vector with $\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta})^T\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta})$, exploiting the structure of the latter, and an approximate solution of (6) is obtained by a few such matrix-vector products. As a result, the conjugate gradient refinement adds little to the memory and computational cost, while it does yield the nice NLS-type convergence behavior. The algorithm has close to quadratic convergence, especially when the residuals are small. NLS has been observed to be more robust for difficult decompositions than plain ALS [89], [91]. The action of $\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta})^T\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta})$ can easily be split into smaller matrix-vector products (N^2 in the N-th order case), which makes inexact NLS overall well-suited for parallel implementation. Variants for low multilinear rank approximation are discussed in [92], [93] and references therein.

D. Exact line search

An important issue in numerical optimization is the choice of step-size. One approach that is sometimes used in multi-way analysis is the following [94], which exploits the multilinearity of the cost function. Suppose we have determined an update ("search") direction, say the negative gradient one. We seek to select the optimal step-size μ for the update

$$\begin{bmatrix}\mathbf{A}\\ \mathbf{B}\\ \mathbf{C}\end{bmatrix} \leftarrow \begin{bmatrix}\mathbf{A}\\ \mathbf{B}\\ \mathbf{C}\end{bmatrix} + \mu \begin{bmatrix}\boldsymbol{\Delta}_A\\ \boldsymbol{\Delta}_B\\ \boldsymbol{\Delta}_C\end{bmatrix},$$

and the goal is to

$$\min_{\mu} \left|\left|\mathbf{X}_1 - ((\mathbf{C}+\mu\boldsymbol{\Delta}_C)\odot(\mathbf{B}+\mu\boldsymbol{\Delta}_B))\,(\mathbf{A}+\mu\boldsymbol{\Delta}_A)^T\right|\right|_F^2.$$

Note that the above cost function is a polynomial of degree 6 in μ. We can determine the coefficients $c_0,\cdots,c_6$ of this polynomial by evaluating it for 7 different values of μ and solving

$$\begin{bmatrix}1 & \mu_1 & \mu_1^2 & \cdots & \mu_1^6\\ 1 & \mu_2 & \mu_2^2 & \cdots & \mu_2^6\\ & & \vdots & & \\ 1 & \mu_7 & \mu_7^2 & \cdots & \mu_7^6\end{bmatrix}\begin{bmatrix}c_0\\ c_1\\ \vdots\\ c_6\end{bmatrix} = \begin{bmatrix}\ell_1\\ \ell_2\\ \vdots\\ \ell_7\end{bmatrix},$$

where $\ell_1,\cdots,\ell_7$ are the corresponding loss values. Once the coefficients are determined, the derivative is the 5-th order polynomial $c_1 + 2c_2\mu + \cdots + 6c_6\mu^5$, and we can use numerical root finding to evaluate the loss at its roots and

[11] In some books, $\frac{\partial f(\mathbf{A})}{\partial \mathbf{A}}$ stands for the transpose of what we denote by $\frac{\partial f(\mathbf{A})}{\partial \mathbf{A}}$, i.e., for an $F \times I$ matrix instead of $I \times F$ in our case.

pick the best μ. The drawback of this is that it requires 11 evaluations of the loss function. We can work out the polynomial coefficients analytically, but this can only save about half of the computation. The bottom line is that optimal line search costs more than gradient computation *per se* (which roughly corresponds to 3 evaluations of the loss function, each requiring $IJKF$ flops for dense data). In practice, we typically use a small, or "good enough" μ, resorting to exact line search in more challenging cases, for instance where the algorithm encounters "swamps". Note that the possibility of exact line search is a profound implication of the multilinearity of the problem. More generally, the optimal update in a search plane (involving two search directions $(\mathbf{\Delta}_{A,1}^T, \mathbf{\Delta}_{B,1}^T, \mathbf{\Delta}_{C,1}^T)^T$ and $(\mathbf{\Delta}_{A,2}^T, \mathbf{\Delta}_{B,2}^T, \mathbf{\Delta}_{C,2}^T)^T$ and even in a three-dimensional search space (additionally involving scaling of $(\mathbf{A}^T, \mathbf{B}^T, \mathbf{C}^T)^T$) can be found via polynomial rooting [95].

E. Missing values

Consider the $\mathbf{C}$-update step in ALS, i.e., $\min_{\mathbf{C}} ||\mathbf{X}_3 - (\mathbf{B} \odot \mathbf{A})\mathbf{C}^T||_F^2$. If there are missing elements in $\mathbf{X}$ (and so in $\mathbf{X}_3$), define the weight tensor

$$\mathbf{W}(i,j,k) = \begin{cases} 1, & \mathbf{X}(i,j,k): \text{ available} \\ 0, & \text{otherwise.} \end{cases},$$

and consider $\min_{\mathbf{C}} ||\mathbf{W}_3 * (\mathbf{X}_3 - (\mathbf{B} \odot \mathbf{A})\mathbf{C}^T)||_F^2 \iff \min_{\mathbf{C}} ||\mathbf{W}_3 * \mathbf{X}_3 - \mathbf{W}_3 * ((\mathbf{B} \odot \mathbf{A})\mathbf{C}^T)||_F^2$, where matrix $\mathbf{W}_3$ is the matrix unfolding of tensor $\mathbf{W}$ obtained in the same way that matrix $\mathbf{X}_3$ is obtained by unfolding tensor $\mathbf{X}$. Notice that the Hadamard operation applies to the product $((\mathbf{B} \odot \mathbf{A})\mathbf{C}^T)$, not to $(\mathbf{B} \odot \mathbf{A})$ – and this complicates things. One may think of resorting to column-wise updates, but this does not work either. Instead, if we perform *row-wise* updates on $\mathbf{C}$, then we have to deal with minimizing over $\mathbf{C}(k,:)$ the squared norm of vector

$$\text{Diag}(\mathbf{W}_3(:,k))\mathbf{X}_3(:,k) - \text{Diag}(\mathbf{W}_3(:,k))(\mathbf{B} \odot \mathbf{A})(\mathbf{C}(k,:))^T,$$

which is a simple linear least squares problem.

There are two basic alternatives to the above strategy for handling missing data. One is to use derivative-based methods, such as (stochastic) gradient descent (see next two subsections) or Gauss-Newton – derivatives are easy to compute, even in the presence of $\mathbf{W}$. Stochastic gradient descent, in particular, computes gradient estimates by drawing only from the observed values. Effectively bypassing the element-wise multiplication by $\mathbf{W}$, stochastic gradient methods deal with missing data in a natural and effortless way. This point is well-known in the machine learning community, but seemingly under-appreciated in the signal processing community, which is more used to handling complete data.

The other alternative is to use a form of expectation-maximization to impute the missing values together with the estimation of the model parameters $\mathbf{A}, \mathbf{B}, \mathbf{C}$ [96]. One can initially impute misses with the average of the available entries (or any other reasonable estimate). More specifically, let $\mathbf{X}_a$ be a tensor that contains the available elements, and $\mathbf{X}_m$ the imputed ones. Then set $\mathbf{X}_c = \mathbf{W} * \mathbf{X}_a + (1 - \mathbf{W}) * \mathbf{X}_m$, and fit $\mathbf{A}, \mathbf{B}, \mathbf{C}$ to $\mathbf{X}_c$. Set $\mathbf{X}_m = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$, $\mathbf{X}_c = \mathbf{W} * \mathbf{X}_a + (1 - \mathbf{W}) * \mathbf{X}_m$, and repeat. It is easy to see that the above procedure amounts to alternating optimization over $\mathbf{A}, \mathbf{B}, \mathbf{C}, (1-\mathbf{W}) * \mathbf{X}_m$, and it thus decreases the cost function monotonically.

Whether it is best to ignore missing elements or impute them is dependent on the application; but we note that for very big and sparse data, imputation is very inefficient in terms of memory, and is thus avoided.

Note that, as a short-cut in large-scale applications, one may deliberately use only part of the available entries when estimating a decomposition [18] (see also the next section); entries that have not been selected, may be used for model cross-validation. If the data structure is sufficiently strong to allow such an approach, it can lead to a very significant speed-up and it should be considered as an alternative to full-scale parallel computation. As a matter of fact, in applications that involve tensors of high order, the latter is not an option due to the curse of dimensionality (i.e., the number of tensor entries depends exponentially on the order and hence quickly becomes astronomically high).

F. Stochastic gradient descent

Stochastic gradient descent (SGD) has become popular in the machine learning community for many types of convex and, very recently, non-convex optimization problems as well. In its simplest form, SGD randomly picks a data point $\mathbf{X}(i,j,k)$ from the available ones, and takes a gradient step only for those model parameters that have an effect on $\mathbf{X}(i,j,k)$; that is, only the i-th row of $\mathbf{A}$, the j-th row of $\mathbf{B}$ and the k-th row of $\mathbf{C}$. We have

$$\frac{\partial}{\partial \mathbf{A}(i,f)} \left(\mathbf{X}(i,j,k) - \sum_{f=1}^{F} \mathbf{A}(i,f)\mathbf{B}(j,f)\mathbf{C}(k,f) \right)^2 = -2 \left(\mathbf{X}(i,j,k) - \sum_{f'=1}^{F} \mathbf{A}(i,f')\mathbf{B}(j,f')\mathbf{C}(k,f') \right) \times \mathbf{B}(j,f)\mathbf{C}(k,f),$$

so that

$$\frac{\partial}{\partial \mathbf{A}(i,:)} = -2 \left(\mathbf{X}(i,j,k) - \sum_{f=1}^{F} \mathbf{A}(i,f)\mathbf{B}(j,f)\mathbf{C}(k,f) \right) \times (\mathbf{B}(j,:) * \mathbf{C}(k,:)).$$

Notice that the product $\mathbf{B}(j,:) * \mathbf{C}(k,:)$ is used once outside and once inside the parenthesis, so the number of multiplications needed is $2F$ for the update of $\mathbf{A}(i,:)$, and $6F$ for the (simultaneous) SGD update of $\mathbf{A}(i,:)$, $\mathbf{B}(j,:)$, $\mathbf{C}(k,:)$. This makes SGD updates very cheap, but the biggest gain is in terms of random access memory (we only need to load one $\mathbf{X}(i,j,k)$, and $\mathbf{A}(i,:)$, $\mathbf{B}(j,:)$, $\mathbf{C}(k,:)$ each time). There is one more inherent advantage to SGD: it can naturally deal with missing elements, as these are simply never "recalled" to execute an update. The drawback is that a truly random disk access pattern is a terrible idea (especially if the data is stored in rotating media) as the computation will inevitably

be bogged down from the disk I/O. For this reason, we prefer fetching blocks of data from secondary memory, and use intelligent caching strategies. To this end, note that SGD updates involving (stemming from) $\mathbf{X}(i,j,k)$ and $\mathbf{X}(i^{'},j^{'},k^{'})$ do not conflict with each other and can be executed in parallel, provided $i^{'} \neq i,\ j^{'} \neq j,\ k^{'} \neq k$ – where all three $\neq$ must hold simultaneously. This means that the maximum number of parallel SGD updates is $\min(I,J,K)$ in the three-way case, and $\min(\{I_n\}_{n=1}^{N})$ in the general N-way case. This limits the *relative* level of parallelization, especially for high N. Another disadvantage is that the convergence can be very slow (sublinear). See [97] for parallel SGD algorithms for CPD and coupled tensor decomposition. In [98] a block sampling variant is discussed that allows one to efficiently decompose TB-size tensors without resorting to parallel computation. The approach leverages CPD uniqueness of the sampled blocks to uniqueness of the CPD of the full tensor. For both SGD and the block sampling variant, the choice of step size is important for convergence. When chosen appropriately, the latter method often converges very fast. A randomized block-sampling approach for very sparse datasets was proposed in [99], building upon the idea of parallel CPD decomposition of multiple pseudo-randomly drawn sub-tensors, and combining the CPDs using *anchor rows*. Sampling is based on mode densities, and identifiability is guaranteed if the sub-tensors have unique CPD.

G. Constraints

In practice, we are often interested in imposing constraints on a CPD model. One may question the need for this – after all, CPD is essentially unique under relatively mild conditions. Constraints are nevertheless useful in

- Restoring identifiability in otherwise non-identifiable cases;
- Improving estimation accuracy in relatively challenging (low-SNR, and/or barely identifiable, and/or numerically ill-conditioned) cases;
- Ensuring interpretability of the results (e.g., power spectra cannot take negative values); and
- As a remedy against ill-posedness.

There are many types of constraints that are relevant in many applications, including those in the "laundry list" below.

- *Symmetry or Hermitian (conjugate) symmetry:* $\mathbf{B} = \mathbf{A}$, or $\mathbf{B} = \mathbf{A}^*$, leading to $\mathbf{X}(:,:,k) = \mathbf{A}\mathbf{D}_k(\mathbf{C})\mathbf{A}^T$ or $\mathbf{X}(:,:,k) = \mathbf{A}\mathbf{D}_k(\mathbf{C})\mathbf{A}^H$. This is actually only *partial symmetry*, with *full symmetry* (or simply *symmetry*) corresponding to $\mathbf{C} = \mathbf{B} = \mathbf{A}$. Partial symmetry corresponds to joint diagonalization of the frontal slabs, using a non-orthogonal and possibly fat diagonalizer $\mathbf{A}$ – that is, the inner dimension can exceed the outer one. Symmetric tensors (with possible conjugation in certain modes) arise when one considers higher-order statistics (HOS).
- *Real-valued parameters:* When $\mathbf{X} \in \mathbb{R}^{I\times J\times K}$, complex-valued $\mathbf{A},\mathbf{B},\mathbf{C}$ make little sense, but sometimes do arise because tensor rank is sensitive to the field over which the decomposition is taken. This is an issue in some applications, particularly in Chemistry and Psychology. Engineers are usually not annoyed by complex $\mathbf{A},\mathbf{B},\mathbf{C}$, but they may still need the following (stronger) constraint to model, e.g., power spectra.
- *Element-wise non-negativity:* $\mathbf{A} \geq 0$ and/or $\mathbf{B} \geq 0$, and/or $\mathbf{C} \geq 0$. When all three are in effect, the resulting problem is known as *non-negative tensor factorization* (NTF). More generally, bound constraints may apply. Non-negativity can help restore uniqueness, because even non-negative matrix factorization (NMF) is unique under certain conditions – these are much more restrictive than those for CPD uniqueness, but, taken together, low-rank CPD structure and non-negativity can restore identifiability. To appreciate this, note that when $k_{\mathbf{C}} = 1$ CPD alone cannot be unique, but if NMF of $\mathbf{X}(:,:,k) = \mathbf{A}\mathbf{D}_k(\mathbf{C})\mathbf{B}^T$ is unique (this requires $F < \min(I,J)$ and a certain level of sparsity in $\mathbf{A}$ and $\mathbf{B}$), then non-negativity can still ensure essential uniqueness of $\mathbf{A},\mathbf{B},\mathbf{C}$.
- *Orthogonality:* This may for instance be the result of prewhitening [49].
- *Probability simplex constraints:* $\mathbf{A}(i,:) \geq 0$, $\mathbf{A}(i,:)\mathbf{1} = 1$, $\forall\ i$, or $\mathbf{A}(:,f) \geq 0$, $\mathbf{1}^T\mathbf{A}(:,f) = 1$, $\forall\ f$, are useful when modeling allocations or probability distributions.
- *Linear constraints:* More general linear constraints on $\mathbf{A},\mathbf{B},\mathbf{C}$ are also broadly used. These can be column-wise, row-wise, or matrix-wise, such as $\text{tr}(\mathbf{W}\mathbf{A}) \leq b$.
- *Monotonicity and related constraints:* These are useful in cases where one deals with, e.g., concentrations that are known to be decaying, or spectra that are known to have a single or few peaks (unimodality, oligo-modality [100]).
- *Sparsity:* In many cases one knows (an upper bound on) the number of nonzero elements of $\mathbf{A},\mathbf{B},\mathbf{C}$, per column, row, or as a whole; or the number of nonzero columns or rows of $\mathbf{A},\mathbf{B},\mathbf{C}$ (group sparsity).
- *Smoothness:* Smoothness can be measured in different ways, but a simple one is in terms of convex quadratic inequalities such as

$$\left\| \begin{bmatrix} -1 & 1 & 0 & \cdots & & 0 \\ 0 & -1 & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & & \end{bmatrix} \mathbf{A} \right\|_F^2 .$$

- *Data model constraints:* All the above constraints apply to the model parameters. One may also be interested in constraints on the reconstructed model of the data, e.g.,

$$(\mathbf{A}\odot\mathbf{B}\odot\mathbf{C})\mathbf{1} \geq 0,\ \text{(element-wise)},\ \mathbf{1}^T(\mathbf{A}\odot\mathbf{B}\odot\mathbf{C})\mathbf{1} = 1,$$

if $\mathbf{X}$ models a joint probability distribution, or in $(\mathbf{A}\odot\mathbf{B}\odot\mathbf{C})\mathbf{1}$ being "smooth" in a suitable sense.
- *Parametric constraints:* All the above are *non-parametric* constraints. There are also important *parametric constraints* that often arise, particularly in signal processing applications. Examples include Vandermonde or Toeplitz structure imposed on $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$. Vandermonde matrices have columns that are (complex or real) exponentials, and Toeplitz matrices model linear time-invariant systems with memory (convolution). Factors may further be explicitly modeled as polynomial, sum-of-exponential, exponential polynomial, rational, or sigmoidal functions. This may for instance reduce the number of parameters needed and suppress noise. Various non-parametric constraints can be explicitly parametrized; e.g., non-negativity

can be parametrized as $\mathbf{A}(i,j) = \theta_{i,j}^2,\ \theta \in \mathbb{R}$, a magnitude constraint $|\mathbf{A}(i,j)| = 1$ as $\mathbf{A}(i,j) = e^{\sqrt{-1}\theta_{i,j}}$, and orthogonality may be parameterized via Jacobi rotations or Householder reflections. Smoothness, probability simplex, and linear constraints can be formulated as parametric constraints as well.

The main issue is then how do we go about enforcing these constraints. The answer depends on the type of constraint considered. Parametric constraints can be conveniently handled in the framework of derivative-based methods such as quasi-Newton and NLS, by using the chain rule in the computation of derivatives. In this way, the proven convergence properties of these methods can be extended to constrained and possibly coupled decompositions [73].

Linear equality and inequality constraints (including monotonicity) on individual loading matrices $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ can be handled in ALS, but change each conditional update from linear least-squares (solving a system of linear equations in the least-squares sense) to quadratic programming. E.g., non-negative least-squares is much slower than unconstrained least squares, and it requires specialized algorithms. This slows down the outer ALS loop considerably; but see the insert entitled *Constrained least squares using ADMM*. Direct application of ADMM for fitting the (nonconvex) CPD model has been considered in [101], using non-negative tensor factorization as the working example. This approach has certain advantages: it can outperform standard approaches in certain scenarios, and it can incorporate various constraints beyond non-negativity, including constraints that couple some of the factor matrices.

The drawback is that direct ADMM requires sophisticated parameter tuning, and even then it is not guaranteed to converge – in contrast to the AO-ADMM hybrid approach of [102] that soon followed [101]. A nice contribution of [101] is that it showed how to parallelize ADMM (and, as a special case, plain ALS) for high-performance computing environments.

Also note that linear constraints on the reconstructed data model, such as $(\mathbf{A} \odot \mathbf{B} \odot \mathbf{C})\mathbf{1} \geq 0$, are nonlinear in $\mathbf{A}, \mathbf{B}, \mathbf{C}$, but become conditionally linear in $\mathbf{A}$ when we fix $\mathbf{B}$ and $\mathbf{C}$, so they too can be handled in the same fashion. Smoothness constraints such as the one above are convex, and can be dualized when solving the conditional mode loading updates, so they can also be handled in ALS, using quadratic programming. Symmetry constraints are more challenging, but one easy way of approximately handling them is to introduce a quadratic penalty term, such as $||\mathbf{B} - \mathbf{A}||_F^2$, which has the benefit of keeping the conditional updates in ALS simple. Sparsity can often be handled using ℓ_1 regularization, which turns linear least squares conditional updates to LASSO-type updates, but one should beware of latent scaling issues, see [107].

Many other constraints though, such as hard ℓ_0 sparsity, unimodality, or finite-alphabet constraints are very hard to handle. A general tool that often comes handy under such circumstances is the following. Consider $\left|\left|\mathbf{X}_3 - (\mathbf{B} \odot \mathbf{A})\mathbf{C}^T\right|\right|_F^2$, and let $\mathbf{M} := (\mathbf{B} \odot \mathbf{A})$. Then $\left|\left|\mathbf{X}_3 - (\mathbf{B} \odot \mathbf{A})\mathbf{C}^T\right|\right|_F^2 = \left|\left|\mathbf{X}_3 - \sum_{f=1}^F \mathbf{M}(:,f)(\mathbf{C}(:,f))^T\right|\right|_F^2$. Fix all the columns of

Constrained least squares using ADMM: Consider $\min_{\mathbf{C}\in\mathcal{C}} \left|\left|\mathbf{X}_3 - \mathbf{M}\mathbf{C}^T\right|\right|_F^2$, where $\mathcal{C}$ is a convex constraint set, and in the context of ALS for CPD $\mathbf{M} := (\mathbf{B} \odot \mathbf{A})$. Introduce an auxiliary variable $\widetilde{\mathbf{C}}$ and the function

$$f_{\mathcal{C}}(\mathbf{C}) := \begin{cases} 0, & \mathbf{C} \in \mathcal{C} \\ \infty, & \text{otherwise.} \end{cases}$$

Then we may equivalently consider

$$\begin{aligned} \min_{\mathbf{C},\widetilde{\mathbf{C}}} \quad & \tfrac{1}{2}\left|\left|\mathbf{X}_3 - \mathbf{M}\mathbf{C}^T\right|\right|_F^2 + f_{\mathcal{C}}(\widetilde{\mathbf{C}}) \\ \text{subject to:} \quad & \widetilde{\mathbf{C}} = \mathbf{C}. \end{aligned}$$

This reformulation falls under the class of problems that can be solved using the alternating direction method of multipliers (ADMM) – see [103] for a recent tutorial. The ADMM iterates for this problem are

$$\begin{aligned} \mathbf{C}^T &\leftarrow (\mathbf{M}^T\mathbf{M} + \rho\mathbf{I})^{-1}(\mathbf{M}^T\mathbf{X}_3 + \rho(\widetilde{\mathbf{C}} + \mathbf{U})^T), \\ \widetilde{\mathbf{C}} &\leftarrow \arg\min_{\widetilde{\mathbf{C}}} f_{\mathcal{C}}(\widetilde{\mathbf{C}}) + \frac{\rho}{2}\|\widetilde{\mathbf{C}} - \mathbf{C} + \mathbf{U}\|_F^2, \\ \mathbf{U} &\leftarrow \mathbf{U} + \widetilde{\mathbf{C}} - \mathbf{C}. \end{aligned}$$

Note that $\mathbf{M}^T\mathbf{X}_3$ and $(\mathbf{M}^T\mathbf{M} + \rho\mathbf{I})^{-1}$ remain fixed throughout the ADMM iterations. We can therefore compute $\mathbf{M}^T\mathbf{X}_3$ (or $\mathbf{X}_3^T\mathbf{M}$: a MTTKRP computation) and the Cholesky decomposition of $(\mathbf{M}^T\mathbf{M} + \rho\mathbf{I}) = \mathbf{L}\mathbf{L}^T$, where $\mathbf{L}$ is lower triangular, and amortize the cost throughout the iterations. Then each update of $\mathbf{C}$ can be performed using one forward and one backward substitution, at much lower complexity. The update of $\widetilde{\mathbf{C}}$ is the so-called *proximity operator* of the function $(1/\rho)f_{\mathcal{C}}(\cdot)$, which is easy to compute in many cases of practical interest [104], including (but not limited to)

- Non-negativity. In this case, the update simply projects onto the non-negative orthant. Element-wise bounds can be handled in the same way.
- Sparsity via ℓ_1-regularization. The update is the well-known *soft-thresholding* operator.
- Simplex constraint. See [105].
- Smoothness regularization. See [102].

In the context of CPD fitting, the above ADMM loop is embedded within the outer Alternating Optimization (AO) loop that alternates over the matrix variables $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$. After several outer iterations, one can use previous iterates to warm-start the inner ADMM loop. This ensures that the inner loop eventually does very few iterations; and, due to computation caching / amortization, each inner loop costs as much as solving an unconstrained linear least squares problem. The net result is that *we can perform constrained ALS at roughly the cost of unconstrained ALS*, for a wide variety of constraints, in a mix-and-match, plug-and-play fashion, so long as the proximity operator involved is easy to compute. This is the main attraction of the AO-ADMM approach of [102], which can also deal with more general loss functions and missing elements, while maintaining the monotone decrease of the cost and conceptual simplicity of ALS. The AO-ADMM framework has been recently extended to handle robust tensor factorization problems where some slabs are grossly corrupted [106].

$\mathbf{C}$ except column f_0. Define $\tilde{\mathbf{X}}_3 = \mathbf{X}_3 - \sum_{f=1,\ f\neq f_0}^{F} \mathbf{M}(:,f)(\mathbf{C}(:,f))^T$, $\mathbf{c} := \mathbf{C}(:,f_0)$, and $\mathbf{m} := \mathbf{M}(:,f_0)$, and consider

$$\min_{\mathbf{c}\in\mathcal{C}} \left|\left| \tilde{\mathbf{X}}_3 - \mathbf{m}\mathbf{c}^T \right|\right|_F^2,$$

where $\mathcal{C}$ is a column-wise constraint.

This corresponds to performing ALS over each column of $\mathbf{C}$, i.e., further breaking down the variable blocks into smaller pieces.

Lemma 1. *For* any *column-wise constraint set $\mathcal{C}$ it holds that*

$$\min_{\mathbf{c}\in\mathcal{C}} \left|\left| \tilde{\mathbf{X}}_3 - \mathbf{m}\mathbf{c}^T \right|\right|_F^2 \Longleftrightarrow \min_{\mathbf{c}\in\mathcal{C}} ||\tilde{\mathbf{c}} - \mathbf{c}||_F^2,$$

where $\tilde{\mathbf{c}} := \left(\frac{\mathbf{m}^T}{||\mathbf{m}||_2^2}\tilde{\mathbf{X}}_3\right)^T$*, i.e., the optimal solution of the constrained least-squares problem is simply the projection of the unconstrained least-squares solution onto the constraint set $\mathcal{C}$. This is known as the* Optimal Scaling Lemma*; see [100] and references therein.*

Armed with Lemma 1, we can easily enforce a wide variety of *column-wise* (but not row-wise) constraints. For example,

- if $\mathcal{C} = \left\{\mathbf{c}\in\mathbb{R}^K \mid w(\mathbf{c}) = s\right\}$, where $w(\cdot)$ counts the number of nonzeros (Hamming weight) of its argument, then $\mathbf{c}_{\text{opt}}$ is obtained by zeroing out the $K-s$ smallest elements of $\tilde{\mathbf{c}}$.
- If $\mathcal{C}$ is the set of complex exponentials, then $\mathbf{c}_{\text{opt}}$ is obtained by peak-picking the magnitude of the Fourier transform of $\tilde{\mathbf{c}}$.
- If $\mathcal{C}$ is the set of non-decreasing sequences, then $\mathbf{c}_{\text{opt}}$ is obtained via monotone regression of $\tilde{\mathbf{c}}$.

The drawback of using Lemma 1 is that it splits the optimization variables into smaller blocks, which usually slows down convergence. Variable splitting can also be used to tackle problems that involve constraints that couple the loading matrices. An example for the partial symmetry constraint ($\mathbf{B} = \mathbf{A}$) is provided in the supplementary material.

VIII. Cramér-Rao Bound

The Cramér-Rao bound is the most commonly used performance benchmarking tool in statistical signal processing. It is a lower bound on the variance of any *unbiased* estimator (and thus on mean square error of unbiased estimators), which is expressed in terms of the (pseudo-)inverse of the *Fisher information matrix*. In many cases it is hard to prove that an estimator is unbiased, but if the empirical bias is small, the Cramér-Rao bound is still used as a benchmark. See the supplementary material for general references and more motivation and details about the Cramér-Rao bound. We derive the Fisher information matrix and the corresponding Cramér-Rao bound for the CPD model

$$\text{vec}\left(\mathbf{X}\right) = (\mathbf{C}\odot\mathbf{B}\odot\mathbf{A})\mathbf{1} + \text{vec}\left(\mathbf{N}\right). \tag{7}$$

Assuming the elements of $\mathbf{N}$ come from an i.i.d. Gaussian distribution with variance σ^2, it has been shown that the FIM can be derived in a simpler way without resorting to taking expectations. Denote $\boldsymbol{\theta}$ as the long vector obtained by stacking all the unknowns,

$$\boldsymbol{\theta} = \left[\begin{array}{ccc} \text{vec}\left(\mathbf{A}\right)^T & \text{vec}\left(\mathbf{B}\right)^T & \text{vec}\left(\mathbf{C}\right)^T \end{array}\right]^T,$$

and define the nonlinear function

$$\boldsymbol{\varphi}(\boldsymbol{\theta}) = (\mathbf{C}\odot\mathbf{B}\odot\mathbf{A})\mathbf{1},$$

then the FIM is simply given by [108] (cf. Proposition 2 in the supplementary material)

$$\boldsymbol{\Phi} = \frac{1}{\sigma^2}\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta})^T\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta}),$$

where $\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta})$ is the Jacobian matrix of $\boldsymbol{\varphi}(\boldsymbol{\theta})$, which can be partitioned into three blocks

$$\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta}) = \left[\mathcal{D}_{\mathbf{A}}\boldsymbol{\varphi}(\boldsymbol{\theta}) \quad \mathcal{D}_{\mathbf{B}}\boldsymbol{\varphi}(\boldsymbol{\theta}) \quad \mathcal{D}_{\mathbf{C}}\boldsymbol{\varphi}(\boldsymbol{\theta})\right].$$

Rewrite $\boldsymbol{\varphi}(\boldsymbol{\theta})$ as follows

$$\begin{aligned}
\boldsymbol{\varphi}(\boldsymbol{\theta}) &= (\mathbf{C}\odot\mathbf{B}\odot\mathbf{A})\mathbf{1} = \text{vec}\left(\mathbf{A}(\mathbf{C}\odot\mathbf{B})^T\right) \\
&= ((\mathbf{C}\odot\mathbf{B})\otimes\mathbf{I}_I)\text{vec}\left(\mathbf{A}\right) \qquad (8)\\
&= \mathbf{K}_{JK,I}(\mathbf{A}\odot\mathbf{C}\odot\mathbf{B})\mathbf{1} = \mathbf{K}_{JK,I}\text{vec}\left(\mathbf{B}(\mathbf{A}\odot\mathbf{C})^T\right) \\
&= \mathbf{K}_{JK,I}((\mathbf{A}\odot\mathbf{C})\otimes\mathbf{I}_J)\text{vec}\left(\mathbf{B}\right) \qquad (9)\\
&= \mathbf{K}_{K,IJ}(\mathbf{B}\odot\mathbf{A}\odot\mathbf{C})\mathbf{1} = \mathbf{K}_{K,IJ}\text{vec}\left(\mathbf{C}(\mathbf{B}\odot\mathbf{A})^T\right) \\
&= \mathbf{K}_{K,IJ}((\mathbf{B}\odot\mathbf{A})\otimes\mathbf{I}_K)\text{vec}\left(\mathbf{C}\right), \qquad (10)
\end{aligned}$$

where $\mathbf{K}_{m,n}$ represents the commutation matrix [109] of size $mn\times mn$. The commutation matrix is a permutation matrix that has the following properties:

1) $\mathbf{K}_{m,n}\text{vec}\left(\mathbf{S}\right) = \text{vec}\left(\mathbf{S}^T\right)$, where $\mathbf{S}$ is $m\times n$;
2) $\mathbf{K}_{p,m}(\mathbf{S}\otimes\mathbf{T}) = (\mathbf{T}\otimes\mathbf{S})\mathbf{K}_{q,n}$, where $\mathbf{T}$ is $p\times q$;
3) $\mathbf{K}_{p,m}(\mathbf{S}\odot\mathbf{T}) = \mathbf{T}\odot\mathbf{S}$;
4) $\mathbf{K}_{n,m} = \mathbf{K}_{m,n}^T = \mathbf{K}_{m,n}^{-1}$;
5) $\mathbf{K}_{mp,n}\mathbf{K}_{mn,p} = \mathbf{K}_{m,np}$.

From (8)-(10), it is easy to see that

$$\begin{aligned}
\mathcal{D}_{\mathbf{A}}\boldsymbol{\varphi}(\boldsymbol{\theta}) &= ((\mathbf{C}\odot\mathbf{B})\otimes\mathbf{I}_I), \\
\mathcal{D}_{\mathbf{B}}\boldsymbol{\varphi}(\boldsymbol{\theta}) &= \mathbf{K}_{JK,I}((\mathbf{A}\odot\mathbf{C})\otimes\mathbf{I}_J), \\
\mathcal{D}_{\mathbf{C}}\boldsymbol{\varphi}(\boldsymbol{\theta}) &= \mathbf{K}_{K,IJ}((\mathbf{B}\odot\mathbf{A})\otimes\mathbf{I}_K).
\end{aligned}$$

Similarly, we can partition the FIM into nine blocks

$$\boldsymbol{\Phi} = \frac{1}{\sigma^2}\boldsymbol{\Psi} = \frac{1}{\sigma^2}\begin{bmatrix} \boldsymbol{\Psi}_{\mathbf{A},\mathbf{A}} & \boldsymbol{\Psi}_{\mathbf{A},\mathbf{B}} & \boldsymbol{\Psi}_{\mathbf{A},\mathbf{C}} \\ \boldsymbol{\Psi}_{\mathbf{B},\mathbf{A}} & \boldsymbol{\Psi}_{\mathbf{B},\mathbf{B}} & \boldsymbol{\Psi}_{\mathbf{B},\mathbf{C}} \\ \boldsymbol{\Psi}_{\mathbf{C},\mathbf{A}} & \boldsymbol{\Psi}_{\mathbf{C},\mathbf{B}} & \boldsymbol{\Psi}_{\mathbf{C},\mathbf{C}} \end{bmatrix}.$$

For the diagonal blocks, using the properties of commutation matrices

$$\begin{aligned}
\boldsymbol{\Psi}_{\mathbf{A},\mathbf{A}} &= ((\mathbf{C}\odot\mathbf{B})\otimes\mathbf{I}_I)^T((\mathbf{C}\odot\mathbf{B})\otimes\mathbf{I}_I) \\
&= (\mathbf{C}^T\mathbf{C} * \mathbf{B}^T\mathbf{B})\otimes\mathbf{I}_I, \\
\boldsymbol{\Psi}_{\mathbf{B},\mathbf{B}} &= (\mathbf{A}^T\mathbf{A} * \mathbf{C}^T\mathbf{C})\otimes\mathbf{I}_J, \\
\boldsymbol{\Psi}_{\mathbf{C},\mathbf{C}} &= (\mathbf{B}^T\mathbf{B} * \mathbf{A}^T\mathbf{A})\otimes\mathbf{I}_K.
\end{aligned}$$

For the off-diagonal blocks, we derive $\boldsymbol{\Psi}_{\mathbf{B},\mathbf{C}}$ here for tutorial purposes

$$\begin{aligned}
\boldsymbol{\Psi}_{\mathbf{B},\mathbf{C}} &= \mathcal{D}_{\mathbf{B}}\boldsymbol{\varphi}(\boldsymbol{\theta})^T\mathcal{D}_{\mathbf{C}}\boldsymbol{\varphi}(\boldsymbol{\theta}) \\
&= ((\mathbf{A}\odot\mathbf{C})\otimes\mathbf{I}_J)^T\mathbf{K}_{JK,I}^T\mathbf{K}_{K,IJ}((\mathbf{B}\odot\mathbf{A})\otimes\mathbf{I}_K) \\
&= ((\mathbf{A}\odot\mathbf{C})\otimes\mathbf{I}_J)^T\mathbf{K}_{IK,J}((\mathbf{B}\odot\mathbf{A})\otimes\mathbf{I}_K).
\end{aligned}$$

To further simplify the expression, let us consider the product of $\mathbf{\Psi}_{\mathbf{B},\mathbf{C}}$ and vec $\left(\tilde{\mathbf{C}}\right)$, where $\tilde{\mathbf{C}}$ is an arbitrary $K \times F$ matrix

$$\begin{aligned}
&\mathbf{\Psi}_{\mathbf{B},\mathbf{C}}\text{vec}\left(\tilde{\mathbf{C}}\right)\\
=&((\mathbf{A}\odot\mathbf{C})\otimes\mathbf{I}_J)^T\mathbf{K}_{IK,J}((\mathbf{B}\odot\mathbf{A})\otimes\mathbf{I}_K)\text{vec}\left(\tilde{\mathbf{C}}\right)\\
=&((\mathbf{A}\odot\mathbf{C})\otimes\mathbf{I}_J)^T\mathbf{K}_{IK,J}\text{vec}\left(\tilde{\mathbf{C}}(\mathbf{B}\odot\mathbf{A})^T\right)\\
=&\mathbf{K}_{F,J}(\mathbf{I}_J\otimes(\mathbf{A}\odot\mathbf{C})^T)(\mathbf{B}\odot\mathbf{A}\odot\tilde{\mathbf{C}})\mathbf{1}\\
=&\mathbf{K}_{F,J}(\mathbf{B}\odot(\mathbf{A}^T\mathbf{A}*\mathbf{C}^T\tilde{\mathbf{C}}))\mathbf{1}\\
=&((\mathbf{A}^T\mathbf{A}*\mathbf{C}^T\tilde{\mathbf{C}})\odot\mathbf{B})\mathbf{1}\\
=&\text{vec}\left(\mathbf{B}(\mathbf{A}^T\mathbf{A}*\mathbf{C}^T\tilde{\mathbf{C}})^T\right)\\
=&(\mathbf{I}_F\otimes\mathbf{B})\text{Diag}\left\{\text{vec}\left(\mathbf{A}^T\mathbf{A}\right)\right\}\text{vec}\left(\tilde{\mathbf{C}}^T\mathbf{C}\right)\\
=&(\mathbf{I}_F\otimes\mathbf{B})\text{Diag}\left\{\text{vec}\left(\mathbf{A}^T\mathbf{A}\right)\right\}\mathbf{K}_{F,F}(\mathbf{I}_F\otimes\mathbf{C}^T)\text{vec}\left(\tilde{\mathbf{C}}\right).
\end{aligned}$$

This holds for all possible $\tilde{\mathbf{C}}\in\mathbb{R}^{K\times F}$, implying

$$\mathbf{\Psi}_{\mathbf{B},\mathbf{C}}=(\mathbf{I}_F\otimes\mathbf{B})\text{Diag}\left\{\text{vec}\left(\mathbf{A}^T\mathbf{A}\right)\right\}\mathbf{K}_{F,F}(\mathbf{I}_F\otimes\mathbf{C}^T).$$

Similarly, we can derive the expression for $\mathbf{\Psi}_{\mathbf{A},\mathbf{B}}$ and $\mathbf{\Psi}_{\mathbf{A},\mathbf{C}}$. The entire expression for the FIM $\mathbf{\Phi}=(1/\sigma^2)\mathbf{\Psi}$ is given in (11).

Formulae for the Jacobian matrix and FIM have appeared in [110]–[114], but the derivation is not as clear and straightforward as the one given here. Furthermore, we show below that $\mathbf{\Psi}$ is rank deficient with deficiency at least $2F$, and identify the associated null subspace $\mathbf{\Psi}$. When the FIM is singular, it has been shown that we can simply take its pseudo-inverse as CRB [115], albeit this bound might be far from attainable, even in theory. When the size of the tensor is large, it may be computationally intractable to take the pseudo-inverse of $\mathbf{\Psi}$ directly, which takes $O((I+J+K)^3F^3)$ flops. Instead of taking this route, we explain how we can compute $\mathbf{\Psi}^\dagger$ efficiently when the deficiency is exactly $2F$. Simulations suggest that the deficiency is exactly $2F$ when the model is identifiable.

Proposition 1. *The rank of the* $(I+J+K)F\times(I+J+K)F$ *FIM* $\mathbf{\Phi}$ *defined in* (11) *is at most* $(I+J+K)F-2F$.

Proof: Please refer to the proof of Proposition 4 for the more general N-way tensor case, in the supplementary material. ∎

When the rank deficiency is equal to $2F$ (which appears to be true almost surely when the model is identifiable, based on our simulations) then we can compute the pseudo-inverse of $\mathbf{\Psi}$ efficiently, invoking the following lemma proven in the supplementary material of [116].

Lemma 2. *Let matrix* $\mathbf{M}$ *be symmetric and singular, and the matrix* $\mathbf{L}$ *satisfying range* $\{\mathbf{L}\}=$ *null* $\{\mathbf{M}\}$*, then*

$$\mathbf{M}^\dagger=(\mathbf{M}+\mathbf{L}\mathbf{L}^T)^{-1}-(\mathbf{L}^\dagger)^T\mathbf{L}^\dagger. \tag{12}$$

A matrix $\mathbf{L}$ that spans the null-space of $\mathbf{\Psi}$ can be written as

$$\mathbf{L}=\mathbf{\Upsilon}\mathbf{E},$$

where

$$\mathbf{E}=\begin{bmatrix}\mathbf{I}_F\odot\mathbf{I}_F & \mathbf{I}_F\odot\mathbf{I}_F\\ -\mathbf{I}_F\odot\mathbf{I}_F & 0\\ 0 & -\mathbf{I}_F\odot\mathbf{I}_F\end{bmatrix}.$$

Since $\mathbf{L}$ has full column rank, its pseudo-inverse is

$$\mathbf{L}^\dagger=\left(\mathbf{L}^T\mathbf{L}\right)^{-1}\mathbf{L}^T=\left(\mathbf{E}^T\mathbf{\Upsilon}^T\mathbf{\Upsilon}\mathbf{E}\right)^{-1}\mathbf{E}^T\mathbf{\Upsilon}^T.$$

Next, we define $\mathbf{\Omega}$ by "completing" the range-space of $\mathbf{\Psi}$

$$\begin{aligned}\mathbf{\Omega}&=\mathbf{\Psi}+\mathbf{L}\mathbf{L}^T\\&=\mathbf{\Delta}+\mathbf{\Upsilon}\mathbf{F}\mathbf{\Upsilon}^T+\mathbf{\Upsilon}\mathbf{E}\mathbf{E}^T\mathbf{\Upsilon}^T\\&=\mathbf{\Delta}+\mathbf{\Upsilon}\left(\mathbf{F}+\mathbf{E}\mathbf{E}^T\right)\mathbf{\Upsilon}^T,\end{aligned}$$

where the definitions of $\mathbf{\Delta}$, $\mathbf{F}$, and $\mathbf{\Upsilon}$ can be found in (11). If $\mathbf{F}+\mathbf{E}\mathbf{E}^T$ is invertible, applying matrix inversion lemma to $\mathbf{\Omega}$ leads to

$$\mathbf{\Omega}^{-1}=\mathbf{\Delta}^{-1}-\mathbf{\Delta}^{-1}\mathbf{\Upsilon}\left(\left(\mathbf{F}+\mathbf{E}\mathbf{E}^T\right)^{-1}+\mathbf{\Upsilon}\mathbf{\Delta}^{-1}\mathbf{\Upsilon}\right)^{-1}\mathbf{\Upsilon}^T\mathbf{\Delta}^{-1}.$$

Notice that $\mathbf{\Delta}$ can be efficiently inverted, because it is block diagonal, and each of its diagonal blocks is a Kronecker product. The most expensive step is to compute $\left(\mathbf{F}+\mathbf{E}\mathbf{E}^T\right)^{-1}$ and $\left(\left(\mathbf{F}+\mathbf{E}\mathbf{E}^T\right)^{-1}+\mathbf{\Upsilon}\mathbf{\Delta}^{-1}\mathbf{\Upsilon}\right)^{-1}$, each taking $O(F^6)$ flops. However, this is still a huge improvement when $F\ll\min(I,J,K)$, compared to directly inverting $\mathbf{\Delta}$ with $O((I+J+K)^3F^3)$ flops. Finally, according to Lemma 2, $\mathbf{\Psi}^\dagger$ can be computed as

$$\begin{aligned}\mathbf{\Psi}^\dagger&=\mathbf{\Delta}^{-1}-(\mathbf{L}^\dagger)^T\mathbf{L}^\dagger\\&=\mathbf{\Delta}^{-1}-\mathbf{\Delta}^{-1}\mathbf{\Upsilon}\left(\left(\mathbf{F}+\mathbf{E}\mathbf{E}^T\right)^{-1}+\mathbf{\Upsilon}\mathbf{\Delta}^{-1}\mathbf{\Upsilon}\right)^{-1}\mathbf{\Upsilon}^T\mathbf{\Delta}^{-1}\\&\quad-\mathbf{\Upsilon}\mathbf{E}\left(\mathbf{E}^T\mathbf{\Upsilon}^T\mathbf{\Upsilon}\mathbf{E}\right)^{-2}\mathbf{E}^T\mathbf{\Upsilon}^T,\end{aligned}$$

and the CRB is simply

$$\mathbf{\Phi}^\dagger=\sigma^2\mathbf{\Psi}^\dagger.$$

IX. Applications

A. Blind Multiuser CDMA

In direct-sequence code-division multiple access (DS-CDMA) communications, a transmitter sends logical digits $d\in\{0,1\}$ by transmitting one of two analog waveforms $bp(t)$, where $b:=(-1)^d\in\{+1,-1\}$, $p(t)=\sum_{\ell=1}^{L}\mathbf{s}(\ell)c(t-\ell T_c)$, $t\in\mathbb{R}$ is the bit signaling pulse, the $L\times 1$ vector $\mathbf{s}$ is the transmitter's *spreading code*, L is the *spreading gain*, T_c is the *chip period*, and $c(\cdot)$ is the *chip pulse*. In practice, the transmitter sends a sequence of digits $\{\mathbf{d}(n)\}_{n=1}^N$ by transmitting $\sum_{n=1}^N\mathbf{b}(n)p(t-nT)$, and the receiver performs matched filtering with respect to the chip pulse and outputs a vector of chip-rate samples that, under ideal conditions (no multipath, perfect synchronization, no noise), reproduce $\{\mathbf{y}_n=\mathbf{s}\mathbf{b}(n)\}_{n=1}^N$ at the other end of the communication link. Collecting these output vectors in an $L\times N$ matrix

$$\mathbf{Y}:=[\mathbf{y}_1,\mathbf{y}_2,\cdots,\mathbf{y}_N]=\mathbf{s}\mathbf{b}^T,$$

we notice that this is rank-1, and we can therefore recover $\mathbf{s}$ and $\mathbf{b}$ from $\mathbf{Y}$ up to an inherently unresolvable sign

$$\sigma^2\mathbf{\Phi} = \mathbf{\Psi} = \begin{bmatrix} \left(\mathbf{B}^T\mathbf{B} * \mathbf{C}^T\mathbf{C}\right) \otimes \mathbf{I}_I & (\mathbf{I}_F \otimes \mathbf{A})\mathbf{F}_\mathbf{C}(\mathbf{I}_F \otimes \mathbf{B})^T & (\mathbf{I}_F \otimes \mathbf{A})\mathbf{F}_\mathbf{B}(\mathbf{I}_F \otimes \mathbf{C})^T \\ (\mathbf{I}_F \otimes \mathbf{B})\mathbf{F}_\mathbf{C}(\mathbf{I}_F \otimes \mathbf{A})^T & \left(\mathbf{A}^T\mathbf{A} * \mathbf{C}^T\mathbf{C}\right) \otimes \mathbf{I}_J & (\mathbf{I}_F \otimes \mathbf{B})\mathbf{F}_\mathbf{A}(\mathbf{I}_F \otimes \mathbf{C})^T \\ (\mathbf{I}_F \otimes \mathbf{C})\mathbf{F}_\mathbf{B}(\mathbf{I}_F \otimes \mathbf{A})^T & (\mathbf{I}_F \otimes \mathbf{C})\mathbf{F}_\mathbf{A}(\mathbf{I}_F \otimes \mathbf{B})^T & \left(\mathbf{A}^T\mathbf{A} * \mathbf{B}^T\mathbf{B}\right) \otimes \mathbf{I}_K \end{bmatrix} = \mathbf{\Delta} + \mathbf{\Upsilon}\mathbf{F}\mathbf{\Upsilon}^T \tag{11}$$

$$\begin{matrix} \mathbf{F}_\mathbf{A} = \text{Diag}\left\{\text{vec}\left(\mathbf{A}^T\mathbf{A}\right)\right\}\mathbf{K}_{F,F} \\ \mathbf{F}_\mathbf{B} = \text{Diag}\left\{\text{vec}\left(\mathbf{B}^T\mathbf{B}\right)\right\}\mathbf{K}_{F,F} \\ \mathbf{F}_\mathbf{C} = \text{Diag}\left\{\text{vec}\left(\mathbf{C}^T\mathbf{C}\right)\right\}\mathbf{K}_{F,F} \end{matrix} \qquad \mathbf{F} = \begin{bmatrix} 0 & \mathbf{F}_\mathbf{B} & \mathbf{F}_\mathbf{C} \\ \mathbf{F}_\mathbf{A} & 0 & \mathbf{F}_\mathbf{C} \\ \mathbf{F}_\mathbf{A} & \mathbf{F}_\mathbf{B} & 0 \end{bmatrix}$$

$$\mathbf{\Delta} = \begin{bmatrix} \left(\mathbf{B}^T\mathbf{B} * \mathbf{C}^T\mathbf{C}\right) \otimes \mathbf{I}_I & 0 & 0 \\ 0 & \left(\mathbf{A}^T\mathbf{A} * \mathbf{C}^T\mathbf{C}\right) \otimes \mathbf{I}_J & 0 \\ 0 & 0 & \left(\mathbf{A}^T\mathbf{A} * \mathbf{B}^T\mathbf{B}\right) \otimes \mathbf{I}_K \end{bmatrix}, \mathbf{\Upsilon} = \begin{bmatrix} \mathbf{I}_F \otimes \mathbf{A} & 0 & 0 \\ 0 & \mathbf{I}_F \otimes \mathbf{B} & 0 \\ 0 & 0 & \mathbf{I}_F \otimes \mathbf{C} \end{bmatrix}$$

ambiguity in this case, simply by reading out a column and row of $\mathbf{Y}$, respectively. In practice there will be noise and other imperfections, so we will in fact extract the principal component of $\mathbf{Y}$ instead, using SVD. This says that we can in fact decode the transmitted sequence even if we do not know the user's "secret" spreading code, up to a sign ambiguity.

However, DS-CDMA is very wasteful in terms of bandwidth when used in single-user mode; in fact it is a multiuser multiplexing modality that is used as an alternative to classical frequency- or time-division multiplexing. When there are two co-channel transmitters sending information simultaneously, then (again under idealized conditions)

$$\mathbf{Y} = \mathbf{s}_1\mathbf{b}_1^T + \mathbf{s}_2\mathbf{b}_2^T = [\mathbf{s}_1,\ \mathbf{s}_2]\,[\mathbf{b}_1,\ \mathbf{b}_2]^T = \mathbf{S}\mathbf{B}^T,$$

In this case, if we know $\mathbf{s}_1$ and $\mathbf{s}_1 \perp \mathbf{s}_2$ (i.e., $\mathbf{s}_1^T\mathbf{s}_2 = 0$), then

$$\mathbf{s}_1^T\mathbf{Y} = \mathbf{s}_1^T\mathbf{s}_1\mathbf{b}_1^T + \mathbf{s}_1^T\mathbf{s}_2\mathbf{b}_2^T = ||\mathbf{s}_1||_2^2\mathbf{b}_1^T,$$

and thus perfect interference cancelation and recovery of the transmitted bits of both users is possible. Even if $\mathbf{s}_1^T\mathbf{s}_2 \neq 0$, so long as they are linearly independent, we can instead use the so-called *zero-forcing* (interference nulling) equalizer

$$\mathbf{S}^\dagger\mathbf{Y} = (\mathbf{S}^T\mathbf{S})^{-1}\mathbf{S}^T\mathbf{Y} = (\mathbf{S}^T\mathbf{S})^{-1}\mathbf{S}^T\mathbf{S}\mathbf{B}^T = \mathbf{B}^T,$$

and recover both streams of bits. This is the basic idea behind DS-CDMA: it is possible to unmix the user transmissions when we know the spreading codes and these are linearly independent (this requires $L \geq$ the number of users/transmitters). However, unmixing is doomed to fail when we do not know $\mathbf{S}$, because rank-two or higher matrix factorization is not unique in general.

The first application of CPD to (communication) signal processing was exactly in bypassing this seemingly insurmountable problem, which is of interest in non-cooperative communications [5]. Suppose that we have two (or more) receivers simultaneously monitoring the same band of interest from different locations. Let $\mathbf{H}(i,f)$ denote the path loss from the f-th transmitter to the i-th receiver. Then

$$\mathbf{Y}_1 = [\mathbf{s}_1,\ \mathbf{s}_2]\begin{bmatrix} \mathbf{H}(1,1) & 0 \\ 0 & \mathbf{H}(1,2) \end{bmatrix}[\mathbf{b}_1,\ \mathbf{b}_2]^T = \mathbf{S}\mathbf{D}_1(\mathbf{H})\mathbf{B}^T,$$

$$\mathbf{Y}_2 = [\mathbf{s}_1,\ \mathbf{s}_2]\begin{bmatrix} \mathbf{H}(2,1) & 0 \\ 0 & \mathbf{H}(2,2) \end{bmatrix}[\mathbf{b}_1,\ \mathbf{b}_2]^T = \mathbf{S}\mathbf{D}_2(\mathbf{H})\mathbf{B}^T,$$

or

$$\mathbf{Y}(i,j,k) := \mathbf{Y}_i(j,k) = \sum_{f=1}^{2}\mathbf{S}(j,f)\mathbf{H}(i,f)\mathbf{B}(k,f),$$

a CPD model of rank $F = 2$. When there are more users, we obtain a CPD model of higher rank. The key point here is that the link to CPD allows recovering everything (spreading codes, information bits, and path losses) for all transmitters, up to the inherent (user) permutation and scaling / counter-scaling ambiguities of CPD. This is true even if there are more co-channel transmitters than the length of the spreading codes, L, so long as one of the CPD identifiability conditions is satisfied. The conceptual development presented here hides practical concerns, such as multipath, noise, imperfect synchronization, etc. There has been considerable follow-up work to address some of these issues, such as [117].

B. Blind source separation

Let us consider the model $\mathbf{y}_n = \mathbf{A}\mathbf{s}_n$, $n \in \{1, 2, \cdots\}$, where $\mathbf{y}_n$ is $I \times 1$, $\mathbf{s}_n$ is $F \times 1$, and we ignore additive noise for simplicity of exposition. We adopt *one* of the following two assumptions.

Assumption A1): Uncorrelated sources of time-varying powers. In this case,

$$\mathbf{R}_n := E\left[\mathbf{y}_n\mathbf{y}_n^T\right] = \mathbf{A}E\left[\mathbf{s}_n\mathbf{s}_n^T\right]\mathbf{A}^T =$$

$$\mathbf{A}\begin{bmatrix} E[(\mathbf{s}_n(1))^2] & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & E[(\mathbf{s}_n(F))^2] \end{bmatrix}\mathbf{A}^T = \mathbf{A}\mathbf{D}_n\mathbf{A}^T.$$

If we assume that the average source powers remain approximately constant over "dwells", and then switch to a different "state", then we can estimate

$$\widehat{\mathbf{R}}_n := \frac{1}{N}\sum_{m=N(n-1)+1}^{Nn}\mathbf{y}_m\mathbf{y}_m^T,$$

yielding a partially symmetric CPD model, which is particularly well-suited for speech signal separation using an array of microphones [4]. Given $\mathbf{A}$, one can use its pseudo-inverse to recover the sources if $\mathbf{A}$ is tall, else it is possible to mitigate crosstalk using various beamforming strategies. When the speakers are moving, we can even track $\mathbf{A}$ over time using adaptive CPD approaches [118].

Assumption A2): Uncorrelated jointly WSS sources having different power spectra. In this case, we rely instead on correlations at different lags, i.e.,

$$\mathbf{R}_{n,\ell} := E\left[\mathbf{y}_n\mathbf{y}_{n-\ell}^T\right] = \mathbf{A}E\left[\mathbf{s}_n\mathbf{s}_{n-\ell}^T\right]\mathbf{A}^T =$$

(since different sources are uncorrelated)

$$\mathbf{A}\begin{bmatrix} E[\mathbf{s}_n(1)\mathbf{s}_{n-\ell}(1)] & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & E[\mathbf{s}_n(F)\mathbf{s}_{n-\ell}(F)] \end{bmatrix}\mathbf{A}^T =$$

(since each source is wide-sense stationary (WSS))

$$\mathbf{A}\begin{bmatrix} r_1(\ell) & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & r_F(\ell)] \end{bmatrix}\mathbf{A}^T = \mathbf{A}\mathbf{D}_\ell(\mathbf{R})\mathbf{A}^T,$$

where $\mathbf{R}$ is the $L \times F$ (lags considered $\times$ sources) matrix holding the autocorrelation vector for all the sources. The power spectrum is the Fourier transform of the autocorrelation vector, hence different spectra are associated with different autocorrelation vectors, providing the necessary diversity to distinguish the sources and estimate $\mathbf{A}$. In practice we use $\widehat{\mathbf{R}}_\ell := \frac{1}{N}\sum_{n=\ell+1}^{N+\ell}\mathbf{y}_n\mathbf{y}_{n-\ell}^T$. This approach to source separation was pioneered by Belouchrani [119]. It took some years to recognize that this is a special (partially symmetric) case of CPD. These two approaches are second-order variants of Independent Component Analysis [120]. Other assumptions are possible, and may involve higher-order statistics, which are by themselves higher-order tensors.

Tensor factorization can also be applied to source separation in the power spectrum domain [121], which has applications in radio astronomy and dynamic spectrum access.

C. Harmonics

Consider the harmonic mixture $\mathbf{Y} = \mathbf{V}\mathbf{S}^T$, where $\mathbf{V}$ is Vandermonde, i.e.,

$$\mathbf{V} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ a_1 & a_2 & \cdots & a_F \\ a_1^2 & a_2^2 & \cdots & a_F^2 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix},$$

where the *generators* in the second row can be real or complex. The model $\mathbf{Y} = \mathbf{V}\mathbf{S}^T$ seems unrelated to tensors, however, upon defining

$$\mathbf{Y}_1 := \mathbf{Y}(1:end-1,:) \quad \text{(all rows except last)}$$

$$\mathbf{Y}_2 := \mathbf{Y}(2:end,:) \quad \text{(all rows except first)}$$

$\mathbf{V}_1 := \mathbf{V}(1:end-1,:)$, and $\mathbf{D} := \text{Diag}([a_1,\ a_2,\ \cdots,\ a_F])$, it is easy to see that

$$\mathbf{Y}_1 = \mathbf{V}_1\mathbf{S}^T; \quad \mathbf{Y}_2 = \mathbf{V}_1\mathbf{D}\mathbf{S}^T,$$

i.e., a two-slab CPD model with Vandermonde structure in one mode. If we instead take $\mathbf{Y}_1 := \mathbf{Y}(1:end-2,:)$, $\mathbf{Y}_2 := \mathbf{Y}(2:end-1,:)$, $\mathbf{Y}_3 := \mathbf{Y}(3:end,:)$, then we obtain a three-slab CPD model with Vandermonde structure in two modes. If $\mathbf{S}$ is also Vandermonde, then we get Vandermonde (harmonic) stucture in all three modes, leading to a *multidimensional harmonic retrieval* (MDHR) problem. There are deep connections between CPD and MDHR (and direction finding using linear and rectangular arrays), originally revealed in [122], [123]; see also [6].

D. Collaborative filtering - based recommender systems

Switching gears, consider a users $\times$ movies ratings matrix $\mathbf{R}$ of size $I \times J$, and the bilinear model $\mathbf{R} \approx \mathbf{U}\mathbf{V}^T$, where $\mathbf{U}$ is $I \times F$, with $F \ll \min(i, J)$, and its i-th row contains a reduced-dimension latent description of user i, $\mathbf{V}$ is $J \times F$ and its j-th row contains a reduced-dimension latent description of movie j, and the model $\mathbf{R} = \mathbf{U}\mathbf{V}^T$ implies that user i's rating of movie j is approximated as $\mathbf{R}(i,j) \approx \mathbf{U}(i,:)(\mathbf{V}(j,:))^T$, i.e., the inner product of the latent descriptions of the i-th user and the j-th movie. The premise of this type of modeling is that every user is a linear combination of F (few) user "types" (e.g., child, college student, ... - these correspond to rows of $\mathbf{V}^T$ / columns of $\mathbf{V}$); and every movie is a linear combination of few movie types (e.g., comedy, drama, documentary, ... - these correspond to columns of $\mathbf{U}$). Typically, only a very small percentage of the entries of $\mathbf{R}$ is available – between 1 per thousand and 1 per 10^5 in practice. Recommender systems try to predict a user's missing ratings using not only that user's past ratings but also the ratings of all other users – hence the term *collaborative filtering*. Notice that, if we can find $\mathbf{U}$ and $\mathbf{V}$ from the available ratings, then we can impute the missing ones using inner products of columns of $\mathbf{U}$ and $\mathbf{V}$. This suggests using the following formulation.

$$\min_{\mathbf{U},\ \mathbf{V}} \left|\left|\mathbf{W} * \left(\mathbf{R} - \mathbf{U}\mathbf{V}^T\right)\right|\right|_F^2,$$

where $\mathbf{W}(i,j) = 1$ if $\mathbf{R}(i,j)$ is available, 0 otherwise. In practice it is unclear what would be a good F, so we typically over-estimate it and then use a rank penalty to control over-fitting and improve generalization ability. The rank of $\mathbf{X}$ is equal to the number of nonzero singular values of $\mathbf{X}$, and the nuclear norm $||\mathbf{X}||_*$ (sum of singular values) is a commonly used convex surrogate for rank ($||\cdot||_1$ vs. $||\cdot||_0$ of the vector of singular values). It has been shown [124] that

$$||\mathbf{X}||_* = \min_{\mathbf{U},\mathbf{V}\ |\ \mathbf{X}=\mathbf{U}\mathbf{V}^T} \frac{1}{2}\left(||\mathbf{U}||_F^2 + ||\mathbf{V}||_F^2\right),$$

giving rise to the following formulation

$$\min_{\mathbf{U},\ \mathbf{V}} \left|\left|\mathbf{W} * \left(\mathbf{R} - \mathbf{U}\mathbf{V}^T\right)\right|\right|_F^2 + \frac{\lambda}{2}\left(||\mathbf{U}||_F^2 + ||\mathbf{V}||_F^2\right).$$

The above "flattened" matrix view of ratings hides the fact that additional information on the context in which the ratings were given is often available. This may range from time stamps to detailed information regarding social context, etc. Every different type of context can be treated as one additional mode, giving rise to (very sparse) higher-order tensors. Taking time stamp (rating time) as an example, consider the user $\times$ movie $\times$ time tensor $\mathbf{R}$ with elements $\mathbf{R}(i,j,k)$. This can be modeled using CPD as

$$\min_{\mathbf{A},\ \mathbf{B},\ \mathbf{C}} \sum_{k=1}^{K} \left|\left|\mathbf{W}(:,:,k) * \left(\mathbf{R}(:,:,k) - \mathbf{A}\mathbf{D}_k(\mathbf{C})\mathbf{B}^T\right)\right|\right|_F^2,$$

where we have switched variables to the familiar ones for CPD. We can use similar rank regularization surrogates in the tensor case as well. We may also impose smoothness in the temporal mode (columns of $\mathbf{C}$) to reflect our expectation that user preferences change slowly over time. These have been

Multilinear maps for classification: In support vector machine (SVM) classification, we use a linear mapping $\mathbf{w}^T\mathbf{x} = \sum_i \mathbf{w}(i)\mathbf{x}(i)$ to discriminate between vectors belonging to two different classes. When the classes are not linearly separable, one can employ a bilinear mapping $\mathbf{x}^T\mathbf{W}\mathbf{x} = \sum_{i,j}\mathbf{W}(i,j)\mathbf{x}(i)\mathbf{x}(j) = \text{vec}\left(\mathbf{x}^T\mathbf{W}\mathbf{x}\right) = \left(\mathbf{x}^T\otimes\mathbf{x}^T\right)\text{vec}(\mathbf{W}) = \left(\mathbf{x}\otimes\mathbf{x}\right)^T\text{vec}(\mathbf{W})$, or even a multilinear one $\left(\mathbf{x}\otimes\mathbf{x}\otimes\mathbf{x}\right)^T\text{vec}(\mathbf{W}) = \sum_{i,j,k}\mathbf{W}(i,j,k)\mathbf{x}(i)\mathbf{x}(j)\mathbf{x}(k)$. Notice that by augmenting $\mathbf{x}$ with a unit as last element (i.e., replacing $\mathbf{x}$ by $[\mathbf{x},1]^T$), higher-order mappings include lower-order ones, hence it suffices to consider the highest order. In such cases, the classifier design problem boils down to designing a suitable matrix or tensor $\mathbf{W}$ of weights. In order to keep the number of model parameters low relative to the number of training samples (to enable statistically meaningful learning and generalization), a low-rank tensor model such as CPD [130] or low multilinear rank one such as Tucker can be employed, and the model parameters can be learned using a measure of classification error as the cost function. A simple optimization solution is to use SGD, drawing samples from the training set at random.

considered by Xiong *et al.* in [125] from a Bayesian point of view, which also proposed a probabilistic model for the hyper-parameters coupled with Markov Chain Monte-Carlo (MCMC) techniques for automated parameter tuning. At about the same time, Karatzoglou *et al.* [126] used age as the third mode, binned into three groups: under 18, 18-50, and over 50. More generally, they proposed adding a new mode for every piece of contextual information provided. The problem with this is that the more modes one adds, the sparser the resulting tensor, and very sparse tensors require high rank to model (recall that a diagonal matrix with nonzero entries on the diagonal is full rank). Karatzoglou *et al.* proposed using a non-orthogonal Tucker model instead of CPD. In particular, they proposed

$$\min_{\mathbf{U},\mathbf{V},\mathbf{W},\mathbf{G}} ||\mathbf{\Gamma} * (\mathbf{R} - (\mathbf{U},\mathbf{V},\mathbf{W},\mathbf{G}))||_F^2 + \lambda\left(||\mathbf{U}||_F^2 + ||\mathbf{V}||_F^2 + ||\mathbf{W}||_F^2\right) + \mu||\mathbf{G}||_F^2,$$

where $\mathbf{\Gamma}(i,j,k) = 1$ if $\mathbf{R}(i,j,k)$ is available, 0 otherwise; $(\mathbf{U},\mathbf{V},\mathbf{W},\mathbf{G})$ stands for the Tucker model that is generated by $\mathbf{U},\mathbf{V},\mathbf{W},\mathbf{G}$; and $||\mathbf{X}||_F^2$ is the sum of squared elements of tensor $\mathbf{X}$. Note that orthogonality is not imposed on $\mathbf{U},\mathbf{V},\mathbf{W}$ *because it is desired to penalize the model's rank* – so constraints of type $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ cannot be imposed (recall that $||\mathbf{U}||_F^2 = \text{Tr}(\mathbf{U}^T\mathbf{U})$).

In recommender systems one may additionally want to exploit side information such as a user similarity matrix that cannot simply be considered as an extra slice of the tensor $\mathbf{R}$. In such cases, *coupled* matrix-tensor decomposition (possibly involving several matrices and tensors) can be used. There are many possibilities and design choices in this direction; we refer the reader to [127]–[129], and [73] which introduces a domain specific language for fast prototyping.

E. Gaussian mixture parameter estimation

Consider F Gaussians $\mathcal{N}(\boldsymbol{\mu}_f,\sigma_f^2\mathbf{I})$, where $\boldsymbol{\mu}_f \in \mathbb{R}^{I\times 1}$ is the mean vector and σ_f^2 is the variance of the elements of the f-th Gaussian. Let $\boldsymbol{\pi} = [\pi_1,\cdots,\pi_F]^T$ be a prior distribution, and consider the following experiment: first draw $f_m \sim \boldsymbol{\pi}$; then draw $\mathbf{x}_m \sim \mathcal{N}(\boldsymbol{\mu}_{f_m},\sigma_{f_m}^2\mathbf{I})$. The distribution of $\mathbf{x}_m$ is then a mixture of the F Gaussians, i.e., $\sum_{f=1}^F \pi_f\mathcal{N}(\boldsymbol{\mu}_f,\sigma_f^2\mathbf{I})$. Now run M independent trials of the above experiment to create $\{\mathbf{x}_m\}_{m=1}^M$. Given $\{\mathbf{x}_m\}_{m=1}^M$, the problem of interest is to estimate the mixture parameters $\{\boldsymbol{\mu}_f,\sigma_f^2,\pi_f\}_{f=1}^F$. Note that it is possible to estimate F from $\{\mathbf{x}_m\}_{m=1}^M$, but we will assume it given for the purposes of our discussion. Note the conceptual similarity of this problem and k-means (here: F-means) clustering or vector quantization (VQ): the main difference is that here we make an additional modeling assumption that the "point clouds" are isotropic Gaussian about their means. Let us consider

$$E[\mathbf{x}_m] = \sum_{f_m=1}^F E[\mathbf{x}_m|f_m]\pi_{f_m} = \sum_{f=1}^F \boldsymbol{\mu}_f\pi_f = \mathbf{M}\boldsymbol{\pi},$$

where $\mathbf{M} := [\boldsymbol{\mu}_1,\cdots,\boldsymbol{\mu}_F]$ $(I\times F)$. Next, consider

$$E[\mathbf{x}_m\mathbf{x}_m^T] = \sum_{f_m=1}^F E[\mathbf{x}_m\mathbf{x}_m^T|f_m]\pi_{f_m} = \sum_{f=1}^F\left(\boldsymbol{\mu}_f\boldsymbol{\mu}_f^T + \sigma_f^2\mathbf{I}\right)\pi_f$$
$$= \mathbf{M}\text{Diag}(\boldsymbol{\pi})\mathbf{M}^T + \bar{\sigma}^2\mathbf{I},\ \bar{\sigma}^2 := \sum_{f=1}^F \sigma_f^2\pi_f.$$

It is tempting to consider third-order moments, which are easier to write out in scalar form

$$E[\mathbf{x}_m(i)\mathbf{x}_m(j)\mathbf{x}_m(k)] = \sum_{f=1}^F E[\mathbf{x}_m(i)\mathbf{x}_m(j)\mathbf{x}_m(k)|f]\pi_f.$$

Conditioned on f, $\mathbf{x}_m(i) = \boldsymbol{\mu}_f(i) + \mathbf{z}_m(i)$, where $\mathbf{z}_m \sim \mathcal{N}(\mathbf{0},\sigma_f^2\mathbf{I})$, and likewise for $\mathbf{x}_m(j)$ and $\mathbf{x}_m(k)$. Plugging these back into the above expression, and using that

- If two out of three indices i,j,k are equal, then $E[\mathbf{z}_m(i)\mathbf{z}_m(j)\mathbf{z}_m(k)|f] = 0$, due to zero mean and independence of the third; and
- If all three indices are equal, then $E[\mathbf{z}_m(i)\mathbf{z}_m(j)\mathbf{z}_m(k)|f] = 0$ because the third moment of a zero-mean Gaussian is zero,

we obtain

$$E[\mathbf{x}_m(i)\mathbf{x}_m(j)\mathbf{x}_m(k)|f] = \boldsymbol{\mu}_f(i)\boldsymbol{\mu}_f(j)\boldsymbol{\mu}_f(k) + \sigma_f^2\left(\boldsymbol{\mu}_f(i)\delta(j-k) + \boldsymbol{\mu}_f(j)\delta(i-k) + \boldsymbol{\mu}_f(k)\delta(i-j)\right),$$

where $\delta(\cdot)$ is the Kronecker delta. Averaging over π_f,

$$\mathbf{R}(i,j,k) := E[\mathbf{x}_m(i)\mathbf{x}_m(j)\mathbf{x}_m(k)] = \sum_{f=1}^F \pi_f\boldsymbol{\mu}_f(i)\boldsymbol{\mu}_f(j)\boldsymbol{\mu}_f(k) + \sum_{f=1}^F \pi_f\sigma_f^2\left(\boldsymbol{\mu}_f(i)\delta(j-k) + \boldsymbol{\mu}_f(j)\delta(i-k) + \boldsymbol{\mu}_f(k)\delta(i-j)\right).$$

At this point, let us further assume, for simplicity, that $\sigma_f^2 = \sigma^2$, $\forall f$, and σ^2 is known. Then $\sum_{f=1}^F \pi_f\boldsymbol{\mu}_f(i) = E[\mathbf{x}_m(i)]$ can be easily estimated. So we may pre-compute the second term in the above equation, call it $\mathbf{\Gamma}(i,j,k)$, and form

$$\mathbf{R}(i,j,k) - \mathbf{\Gamma}(i,j,k) = \sum_{f=1}^F \pi_f\boldsymbol{\mu}_f(i)\boldsymbol{\mu}_f(j)\boldsymbol{\mu}_f(k),$$

which is evidently a symmetric CPD model of rank (at most) F. Note that, due to symmetry and the fact that $\pi_f \geq 0$, there is no ambiguity regarding the sign of $\boldsymbol{\mu}_f$; but we can still set e.g., $\boldsymbol{\mu}_1^{'} = \rho^{\frac{1}{3}}\boldsymbol{\mu}_1$, $\pi_1^{'} = \frac{1}{\rho}\pi_1$, $\pi_2^{'} = \pi_1 + \pi_2 - \pi_1^{'} = \frac{\rho-1}{\rho}\pi_1 + \pi_2$, $\frac{1}{\gamma} = \frac{\pi_2^{'}}{\pi_2}$, and $\boldsymbol{\mu}_2^{'} = \gamma^{\frac{1}{3}}\boldsymbol{\mu}_2$, for some $\rho > 0$. However, we must further ensure that $\pi_2^{'} > 0$, and $\pi_1^{'} < \pi_1 + \pi_2$; both require $\rho > \frac{\pi_1}{\pi_1+\pi_2}$. We see that scaling ambiguity remains, and is important to resolve it here, otherwise we will obtain the wrong means and mixture probabilities. Towards this end, consider lower-order statistics, namely $E[\mathbf{x}_m\mathbf{x}_m^T]$ and $E[\mathbf{x}_m]$. Note that,

$$(\mathbf{M}\odot\mathbf{M}\odot\mathbf{M})\boldsymbol{\pi} = ((\mathbf{M}\mathbf{D}^{1/3})\odot(\mathbf{M}\mathbf{D}^{1/3})\odot(\mathbf{M}\mathbf{D}^{1/3}))\mathbf{D}^{-1}\boldsymbol{\pi}$$

but

$$E[\mathbf{x}_m] = \mathbf{M}\boldsymbol{\pi} \neq (\mathbf{M}\mathbf{D}^{1/3})\mathbf{D}^{-1}\boldsymbol{\pi},$$
$$E[\mathbf{x}_m\mathbf{x}_m^T] - \bar{\sigma}^2\mathbf{I} = \mathbf{M}\mathrm{Diag}(\boldsymbol{\pi})\mathbf{M}^T$$
$$\overset{\mathrm{vec}(\cdot)}{\rightarrow} (\mathbf{M}\odot\mathbf{M})\boldsymbol{\pi} \neq ((\mathbf{M}\mathbf{D}^{1/3})\odot(\mathbf{M}\mathbf{D}^{1/3}))\mathbf{D}^{-1}\boldsymbol{\pi}.$$

This shows that no scaling ambiguity remains when we jointly fit third and second (or third and first) order statistics. For the general case, when the variances $\left\{\sigma_f^2\right\}_{f=1}^F$ are unknown and possibly different, see [131]. A simpler work-around is to treat "diagonal slabs" (e.g., corresponding to $j=k$) as missing, fit the model, then use it to estimate $\left\{\sigma_f^2\right\}_{f=1}^F$ and repeat.

F. Topic modeling

Given a dictionary $\mathcal{D} = \{w_1, \cdots, w_I\}$ comprising I possible words, a *topic* is a probability mass function (pmf) over $\mathcal{D}$. Assume there are F topics overall, let $\mathbf{p}_f := \Pr(w_i|f)$ be the pmf associated with topic f, π_f be the probability that one may encounter a document associated with topic f, and $\boldsymbol{\pi} := [\pi_1, \cdots, \pi_f]^T$. Here we begin our discussion of topic modeling by assuming that each document is related to one and only one topic (or, document "type"). Consider the following experiment:

1) Draw a document at random;
2) Sample m words from it, independently, and at random (with replacement – and the order in which words are drawn does not matter);
3) Repeat (until you collect "enough samples" – to be qualified later).

Assume for the moment that F is known. Your objective is to estimate $\{\mathbf{p}_f, \pi_f\}_{f=1}^F$. Clearly, $\Pr(w_i) = \sum_{f=1}^F \Pr(w_i|f)\pi_f$; furthermore, the word co-occurrence probabilities $\Pr(w_i, w_j)$:=Pr(word i and word j are drawn from the same document) satisfy

$$\Pr(w_i, w_j) = \sum_{f=1}^F \Pr(w_i, w_j|f)\pi_f = \sum_{f=1}^F \mathbf{p}_f(i)\mathbf{p}_f(j)\pi_f,$$

since the words are independently drawn from the document. Define the matrix of word co-occurrence probabilities $\mathbf{P}^{(2)}$ with elements $\mathbf{P}^{(2)}(i,j) := \Pr(w_i, w_j)$, and the matrix of conditional pmfs $\mathbf{C} := [\mathbf{p}_1, \cdots, \mathbf{p}_F]$. Then

$$\mathbf{P}^{(2)} = \mathbf{C}\mathrm{Diag}(\boldsymbol{\pi})\mathbf{C}^T.$$

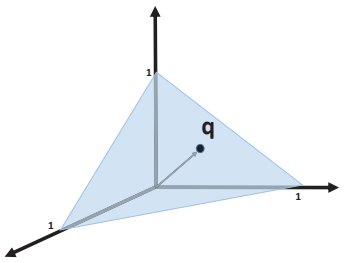

Fig. 7: 2-D probability simplex in 3-D space.

Next, consider "trigrams" – i.e., probabilities of triples of words being drawn from the same document

$$\Pr(w_i,w_j,w_k) = \sum_{f=1}^F \Pr(w_i,w_j,w_k|f)\pi_f = \sum_{f=1}^F \mathbf{p}_f(i)\mathbf{p}_f(j)\mathbf{p}_f(k)\pi_f.$$

Define tensor $\mathbf{P}^{(3)}$ with elements $\mathbf{P}^{(3)}(i,j,k) := \Pr(w_i,w_j,w_k)$. Then $\mathbf{P}^{(3)}$ admits a symmetric non-negative CPD model of rank (at most) F:

$$\mathbf{P}^{(3)} = (\mathbf{C}\odot\mathbf{C}\odot\mathbf{C})\boldsymbol{\pi}.$$

Similar to[12] Gaussian mixture parameter estimation, we can estimate $\mathbf{C}$ and $\boldsymbol{\pi}$ from the tensor $\mathbf{P}^{(3)}$ and the matrix $\mathbf{P}^{(2)}$. In reality, we will use empirical word co-occurrence counts to estimate $\mathbf{P}^{(3)}$ and $\mathbf{P}^{(2)}$, and for this we need to sample enough triples ("enough samples"). Once we have $\mathbf{C}$, we can classify any document by estimating (part of) its conditional word pmf and comparing it to the columns of $\mathbf{C}$.

Next, consider the more realistic situation where each document is a mixture of topics, modeled by a pmf $\mathbf{q}$ $(F\times 1)$ that is itself drawn from a distribution $\delta(\cdot)$ over the $(F-1)$-dimensional *probability simplex* – see Fig. 7. Our working experiment is now modified as follows.

1) For every document we sample, we draw $\mathbf{q} \sim \delta(\cdot)$;
2) For every word we sample from the given document, we first draw a topic t from $\mathbf{q}$ – i.e., topic f is selected with probability $\mathbf{q}(f)$;
3) Next, we draw a word $\sim \mathbf{p}_t$;
4) Goto 2, until you have sampled the desired number of words (e.g., 3) from the given document;
5) Goto 1, until you have collected enough samples (e.g., enough triples).

Then,

$$\Pr(w_i, w_j|t_1, t_2, \mathbf{q}) = \mathbf{p}_{t_1}(i)\mathbf{p}_{t_2}(j) \Longrightarrow$$
$$\Pr(w_i, w_j|\mathbf{q}) = \sum_{t_1=1}^F\sum_{t_2=1}^F \mathbf{p}_{t_1}(i)\mathbf{p}_{t_2}(j)\mathbf{q}(t_1)\mathbf{q}(t_2) \Longrightarrow$$
$$\Pr(w_i, w_j) = \sum_{t_1=1}^F\sum_{t_2=1}^F \mathbf{p}_{t_1}(i)\mathbf{p}_{t_2}(j)E\left[\mathbf{q}(t_1)\mathbf{q}(t_2)\right],$$

[12]But in fact simpler from, since here, due to sampling with replacement, the same expression holds even if two or three indices i, j, k are the same.

where we notice that what comes into play is the second-order statistics $E\left[\mathbf{q}(t_1)\mathbf{q}(t_2)\right]$ (the correlation) of $\delta(\cdot)$. Likewise, it follows that, for the trigrams, $\Pr(w_i, w_j, w_k) =$

$$\sum_{t_1=1}^{F}\sum_{t_2=1}^{F}\sum_{t_3=1}^{F}\mathbf{p}_{t_1}(i)\mathbf{p}_{t_2}(j)\mathbf{p}_{t_3}(k)E\left[\mathbf{q}(t_1)\mathbf{q}(t_2)\mathbf{q}(t_3)\right],$$

which involves the third-order statistics tensor $\mathbf{G}$ of $\delta(\cdot)$ with elements $\mathbf{G}(i,j,k) := E\left[\mathbf{q}(t_1)\mathbf{q}(t_2)\mathbf{q}(t_3)\right]$. Defining the $I \times I \times I$ tensor $\mathbf{P}$ with elements $\mathbf{P}(i,j,k) := \Pr(w_i, w_j, w_k)$, it follows that $\mathbf{P}$ admits a symmetric Tucker decomposition, $\mathbf{P} = \text{Tucker}(\mathbf{C}, \mathbf{C}, \mathbf{C}, \mathbf{G})$, with $\mathbf{C} = [\mathbf{p}_1, \cdots, \mathbf{p}_F]$. Note that $\mathbf{C}$ is element-wise non-negative, but in principle $\mathbf{G}$ may have negative elements. As we know, Tucker models are not identifiable in general – there is linear transformation freedom. This can be alleviated when one can assume sparsity in $\mathbf{C}$ [132], $\mathbf{G}$, or both (intuitively, this is because linear transformations generally do not preserve sparsity).

G. Multilinear discriminative subspace learning

Consider the following *discriminative subspace learning* problem: given $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_M]$ $(N \times M)$ and associated class labels $\mathbf{z} = [z_1, \cdots, z_M]$ $(1 \times M)$ for the columns of $\mathbf{X}$, find a dimensionality-reducing linear transformation $\mathbf{U}$ of size $N \times F$, $F < N$ (usually $F \ll N$) such that

$$\min_{\mathbf{U}|\mathbf{U}^T\mathbf{U}=\mathbf{I}} \sum_{m=1}^{M}\left\{(1-\lambda)\sum_{\ell=1|z_\ell=z_m}^{M}||\mathbf{U}^T\mathbf{x}_m - \mathbf{U}^T\mathbf{x}_\ell||_2^2 - \lambda\sum_{\ell=1|z_\ell\neq z_m}^{M}||\mathbf{U}^T\mathbf{x}_m - \mathbf{U}^T\mathbf{x}_\ell||_2^2\right\},$$

where the first (second) term measures the within-class (across-class) distance in reduced dimension space. We are therefore trying to find a dimensionality-reducing transformation that will map points close in terms of Euclidean distance if they have the same class label, far otherwise. Another way to look at it is that we are trying to find a subspace to project onto where we can easily visualize (if $F = 2$ or 3) the point clouds of the different classes. Upon defining

$$w_{m,\ell} := (1-\lambda)^{1(z_\ell=z_m)}(-\lambda)^{1-1(z_\ell=z_m)},$$

where $1(z_\ell = z_m) = 1$ if $z_\ell = z_m$, 0 otherwise, we can compactly write the problem as follows

$$\min_{\mathbf{U}|\mathbf{U}^T\mathbf{U}=\mathbf{I}} \sum_{m=1}^{M}\sum_{\ell=1}^{M}||\mathbf{U}^T\mathbf{x}_m - \mathbf{U}^T\mathbf{x}_\ell||_2^2 w_{m,\ell}.$$

Expanding the squared norm and using properties of $\text{Tr}(\cdot)$, we can write the cost function as

$$\sum_{m=1}^{M}\sum_{\ell=1}^{M}||\mathbf{U}^T\mathbf{x}_m - \mathbf{U}^T\mathbf{x}_\ell||_2^2 w_{m,\ell} = \text{Tr}(\mathbf{U}\mathbf{U}^T\mathbf{Y}),$$

where

$$\mathbf{Y} := \sum_{m=1}^{M}\sum_{\ell=1}^{M} w_{m,\ell}(\mathbf{x}_m - \mathbf{x}_\ell)(\mathbf{x}_m - \mathbf{x}_\ell)^T.$$

Notice that $w_{m,\ell} = w_{\ell,m}$ by definition, and $\mathbf{Y}$ is symmetric. Let $\mathbf{Y} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$ be the eigendecomposition of $\mathbf{Y}$, and note that $\text{Tr}(\mathbf{U}\mathbf{U}^T\mathbf{Y}) = \text{Tr}(\mathbf{U}^T\mathbf{Y}\mathbf{U})$. Clearly, $\mathbf{U}_{\text{opt}} = F$ minor eigenvectors of $\mathbf{Y}$ (columns of $\mathbf{V}$ corresponding to the F smallest elements on the diagonal of $\mathbf{\Lambda}$).

Now, suppose that the columns in $\mathbf{X}$ are in fact vectorized tensors. As an example, suppose that there exist *common* bases $\mathbf{U}$ $(I \times r_1)$, $\mathbf{V}$ $(J \times r_2)$, $\mathbf{W}$ $(K \times r_3)$, such that

$$\mathbf{x}_m \approx (\mathbf{U} \otimes \mathbf{V} \otimes \mathbf{W})\mathbf{g}_m, \quad \forall m \in \{1, \cdots, M\},$$

i.e., each $\mathbf{x}_m$ can be modeled using a $\perp$-Tucker model with common mode bases, but different cores for different m. We can think of $(\mathbf{U}\otimes\mathbf{V}\otimes\mathbf{W})^T$ $(r_1r_2r_3 \times IJK)$ as a (Kronecker) structured dimensionality reducing transformation, and the vectorized core array $\mathbf{g}_m$ as the low-dimensional $(r_1r_2r_3 \times 1)$ representation of $\mathbf{x}_m$. We want to find $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$ such that the $\mathbf{g}$'s corresponding to $\mathbf{x}$'s in the same (different) class are close (far) from each other. Following the same development as before, using

$$\widehat{\mathbf{g}}_m = (\mathbf{U} \otimes \mathbf{V} \otimes \mathbf{W})^T\mathbf{x}_m$$

as the projection of $\mathbf{x}_m$ in reduced-dimension space, we arrive at

$$\min_{\mathbf{U},\mathbf{V},\mathbf{W}} \text{Tr}\left((\mathbf{U} \otimes \mathbf{V} \otimes \mathbf{W})(\mathbf{U} \otimes \mathbf{V} \otimes \mathbf{W})^T\mathbf{Y}\right),$$

$$\text{subject to: } \mathbf{U}^T\mathbf{U} = \mathbf{I},\ \mathbf{V}^T\mathbf{V} = \mathbf{I},\ \mathbf{W}^T\mathbf{W} = \mathbf{I},$$

or, equivalently,

$$\min_{\mathbf{U},\mathbf{V},\mathbf{W}} \text{Tr}\left(((\mathbf{U}\mathbf{U}^T) \otimes (\mathbf{V}\mathbf{V}^T) \otimes (\mathbf{W}\mathbf{W}^T))\mathbf{Y}\right),$$

$$\text{subject to: } \mathbf{U}^T\mathbf{U} = \mathbf{I},\ \mathbf{V}^T\mathbf{V} = \mathbf{I},\ \mathbf{W}^T\mathbf{W} = \mathbf{I},$$

from which it is clear that, conditioned on, say, $\mathbf{U}$ and $\mathbf{V}$, the update with respect to $\mathbf{W}$ boils down to

$$\min_{\mathbf{W}|\mathbf{W}^T\mathbf{W}=\mathbf{I}} \text{Tr}\left(\mathbf{W}\mathbf{W}^T\mathbf{Z}\right),$$

for some matrix $\mathbf{Z}$ that depends on the values of $\mathbf{U}$ and $\mathbf{V}$. See [133], [134] for more on the topic of multilinear subspace learning.

The applications that we reviewed in some depth are by no means exhaustive – there are a lot more success stories using tensors for data mining and machine learning, e.g., for higher-order web link analysis [135], and spotting misbehaviors in location-based social networks [136]; see also [99].

X. Software, demos, history, and what lies ahead

As we wrap up this admittedly long article, we would like to point out some widely available resources that can help bring the reader up to speed experimenting with tensors in minutes. Matlab provides native support for tensor objects, but working with tensors is facilitated by these freely available toolboxes:

1) The `n-way toolbox` http://www.models.life.ku.dk/nwaytoolbox by Bro *et al.* [137], based on ALS (with Gauss-Newton, line-search and other methods as an option) incorporates many *non-parametric* types of constraints, such as non-negativity;

2) The `tensor toolbox` http://www.sandia.gov/~tgkolda/TensorToolbox/index-2.6.html by Kolda *et al.* [138], [139] was the first to provide support for sparse, dense, and factored tensor classes, alongside standard ALS and all-at-once algorithms for CPD, MLSVD, and other factorizations;
3) `Tensorlab` http://www.tensorlab.net/ by De Lathauwer *et al.* [140], builds upon the complex optimization framework and offers numerical algorithms for computing CPD, MLSVD and more general block term decompositions. It includes a library of constraints and regularization penalties and offers the possibility to combine and jointly factorize dense, sparse, structured and incomplete tensors. It provides special routines for large-scale problems and visualization.
4) `SPLATT` http://glaros.dtc.umn.edu/gkhome/splatt/overview by Smith *et al.* is a high-performance computing software toolkit for parallel sparse tensor factorization. It contains memory- and operation-efficient algorithms that allows it to compute PARAFAC decompositions of very large sparse datasets. SPLATT is written in C and OpenMP.
5) The `TensorPackage` http://www.gipsa-lab.fr/~pierre.comon/TensorPackage/tensorPackage.html by Comon *et al.*, which includes various algorithms for CPD and employs enhanced line search [94].

While these toolboxes are great to get you going and for rapid prototyping, when it comes to really understanding what you're doing with tensors, there is nothing as valuable as programming ALS for CPD and $\perp$-Tucker yourself, and trying them on real data. Towards this end, we have produced educational "plain-vanilla" programs (CPD-ALS, MLSVD, $\perp$-Tucker-ALS, CPD-GD, CPD-SGD), and simple but instructive demos (multichannel speech separation, and faces tensor compression) which are provided as supplementary material together with this article.

Tensor decomposition has come a long way since Hitchcock '27, [141], Cattell '44 [142], and later Harshman '70-'72 [31], [32], Carroll and Chang [143], and Kruskal's '77 [35] seminal papers. It is now a vibrant field that is well-represented in major IEEE, ACM, SIAM, and other mainstream conferences. The cross-disciplinary community that nurtured tensor decomposition research during the years that it was a niche area has two dedicated workshops that usually happen every three years: the TRICAP (Three-way methods In Chemistry and Psychology) workshop, which was last organized in 2015 at Pecol – Val di Zoldo (Belluno), Italy http://people.ece.umn.edu/~nikos/TRICAP_home.html; and the TDA (Tensor Decompositions and Applications) workshop, which was last organized in 2016 at Leuven, Belgium http://www.esat.kuleuven.be/stadius/TDA2016/.

In terms of opportunities and challenges ahead, we see the need for more effective and tractable tensor rank detection criteria, and flexible and scalable algorithms that can handle very big datasets while offering identifiability, convergence, and parameter RMSE performance guarantees – at least under certain reasonable conditions. Data fusion, in the form of coupled decomposition of several related tensors and matrices is a very promising direction with numerous applications. More broadly, we believe that machine learning offers a wide range of potential applications, and this is one of the reasons why we wrote this article. Tensor decomposition in higher rank blocks is another interesting but relatively under-explored area with many applications. Finally, using multilinear models such as tensor trains as "universal approximants" has been gaining traction and will continue to grow in the foreseeable future, as a way to get away from the "curse of dimensionality".

REFERENCES

[1] T. G. Kolda, B. W. Bader, and J. P. Kenny, "Higher-order web link analysis using multilinear algebra," in *Fifth IEEE International Conference on Data Mining (ICDM'05)*, Nov 2005, pp. 8 pp.–.

[2] E. Acar, S.A. Camtepe, M.S. Krishnamoorthy, and B. Yener, *Modeling and Multiway Analysis of Chatroom Tensors*, pp. 256–268, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[3] E. Acar and B. Yener, "Unsupervised multiway data analysis: A literature survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 1, pp. 6–20, Jan 2009.

[4] D. Nion, K.N. Mokios, N.D. Sidiropoulos, and A. Potamianos, "Batch and adaptive PARAFAC-based blind separation of convolutive speech mixtures," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1193–1207, 2010.

[5] N.D. Sidiropoulos, G.B. Giannakis, and R. Bro, "Blind PARAFAC receivers for DS-CDMA systems," *IEEE Transactions on Signal Processing*, vol. 48, no. 3, pp. 810–823, 2000.

[6] D. Nion and N.D. Sidiropoulos, "Tensor algebra and multidimensional harmonic retrieval in signal processing for MIMO radar," *IEEE Transactions on Signal Processing*, vol. 58, no. 11, pp. 5693–5705, Nov 2010.

[7] N. D. Sidiropoulos, R. Bro, and G. B. Giannakis, "Parallel factor analysis in sensor array processing," *IEEE Trans. Signal Process.*, vol. 48, no. 8, pp. 2377–2388, Aug. 2000.

[8] A.-J. van der Veen and A. Paulraj, "An analytical constant modulus algorithm," *IEEE Transactions on Signal Processing*, vol. 44, no. 5, pp. 1136–1155, May 1996.

[9] E. Kofidis and P. Regalia, "Tensor approximation and signal processing applications," *Contemporary Mathematics*, vol. 280, pp. 103–134, 2001.

[10] M. Vasilescu and D. Terzopoulos, *Multilinear Analysis of Image Ensembles: TensorFaces*, pp. 447–460, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[11] A. Cichocki, D. Mandic, L. De Lathauwer, Zhou Q., Zhao Q., C. Caiafa, and A.H. Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *Signal Processing Magazine, IEEE*, vol. 32, no. 2, pp. 145–163, March 2015.

[12] E.E. Papalexakis, C. Faloutsos, and N.D. Sidiropoulos, "Parcube: Sparse parallelizable tensor decompositions.," in *ECML/PKDD (1)*, P.A. Flach, T.D. Bie, and N. Cristianini, Eds. 2012, vol. 7523 of *Lecture Notes in Computer Science*, pp. 521–536, Springer.

[13] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky, "Tensor decompositions for learning latent variable models," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 2773–2832, Jan. 2014.

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

[15] P. Comon, "Tensors : A brief introduction," *Signal Processing Magazine, IEEE*, vol. 31, no. 3, pp. 44–53, May 2014.

[16] 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.

[17] W. Hackbusch, *Tensor spaces and numerical tensor calculus*, vol. 42, Springer Science & Business Media, 2012.

[18] N. Vervliet, O. Debals, L. Sorber, and L De Lathauwer, "Breaking the Curse of Dimensionality Using Decompositions of Incomplete Tensors: Tensor-based scientific computing in big data analysis," *Signal Processing Magazine, IEEE*, vol. 31, no. 5, pp. 71–79, Sept. 2014.

[19] R. Bro, "PARAFAC. Tutorial and applications," *Chemometrics and Intelligent Laboratory Systems*, vol. 38, pp. 149–171, 1997.

[20] A.K. Smilde, R. Bro, and P. Geladi, *Multi-way analysis with applications in the chemical sciences*, John Wiley & Sons, 2004.

[21] E.E. Papalexakis, C. Faloutsos, N.D. Sidiropoulos, and T. Kolda, "Tensors for data mining and data fusion: Models, applications, and scalable algorithms," *ACM Trans. on Intelligent Systems and Technology*, vol. 8, no. 2, pp. 16:1–16:44, Nov. 2016.

[22] G. Bergqvist, "Exact probabilities for typical ranks of $2 \times 2 \times 2$ and $3 \times 3 \times 2$ tensors," *Linear Algebra and its Applications*, vol. 438, no. 2, pp. 663 – 667, 2013.

[23] P. Paatero, "Construction and analysis of degenerate parafac models," *Journal of Chemometrics*, vol. 14, no. 3, pp. 285–299, 2000.

[24] L.-H. Lim and P. Comon, "Nonnegative approximations of nonnegative tensors," *Journal of Chemometrics*, vol. 23, no. 7-8, pp. 432–441, 2009.

[25] Y. Qi, P. Comon, and L. H. Lim, "Uniqueness of nonnegative tensor approximations," *IEEE Transactions on Information Theory*, vol. 62, no. 4, pp. 2170–2183, April 2016.

[26] W. Krijnen, T. Dijkstra, and A. Stegeman, "On the non-existence of optimal solutions and the occurrence of degeneracy in the candecomp/parafac model," *Psychometrika*, vol. 73, no. 3, pp. 431–439, 2008.

[27] J. Landsberg, *Tensors: geometry and applications*, vol. 128, American Mathematical Soc., 2012.

[28] Nick Vannieuwenhoven, Raf Vandebril, and Karl Meerbergen, "A randomized algorithm for testing nonsingularity of structured matrices with an application to asserting nondefectivity of segre varieties," *IMA Journal of Numerical Analysis*, p. drt069, 2014.

[29] James Alexander and André Hirschowitz, "Polynomial interpolation in several variables," *Journal of Algebraic Geometry*, vol. 4, no. 2, pp. 201–222, 1995.

[30] J. Håstad, "Tensor rank is np-complete," *J. Algorithms*, vol. 11, no. 4, pp. 644–654, Dec. 1990.

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

[32] R.A. Harshman, "Determination and proof of minimum uniqueness conditions for PARAFAC-1," *UCLA Working Papers in Phonetics*, vol. 22, pp. 111–117, 1972.

[33] R. Roy, A. Paulraj, and T. Kailath, "ESPRIT–a subspace rotation approach to estimation of parameters of cisoids in noise," *IEEE Trans. Acoustics, Speech and Signal Process.*, vol. 34, no. 5, pp. 1340 – 1342, oct 1986.

[34] Jos M. F. Ten Berge and Jorge N. Tendeiro, "The link between sufficient conditions by harshman and by kruskal for uniqueness in candecomp/parafac," *Journal of Chemometrics*, vol. 23, no. 7-8, pp. 321–323, 2009.

[35] J.B. Kruskal, "Three-way arrays: Rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics," *Linear Algebra and its Applications*, vol. 18, no. 2, pp. 95–138, 1977.

[36] T. Jiang and N.D. Sidiropoulos, "Kruskal's permutation lemma and the identification of CANDECOMP/PARAFAC and bilinear models with constant modulus constraints," *IEEE Transactions on Signal Processing*, vol. 52, no. 9, pp. 2625–2636, 2004.

[37] L. De Lathauwer, "A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization," *SIAM journal on Matrix Analysis and Applications*, vol. 28, no. 3, pp. 642–666, 2006.

[38] L. Chiantini and G. Ottaviani, "On generic identifiability of 3-tensors of small rank," *SIAM. J. Matrix Anal. & Appl.*, vol. 33, no. 3, pp. 1018–1037, 2012.

[39] I. Domanov and L. De Lathauwer, "Generic uniqueness conditions for the canonical polyadic decomposition and indscal," *SIAM Journal on Matrix Analysis and Applications*, vol. 36, no. 4, pp. 1567–1589, 2015.

[40] V. Strassen, "Rank and optimal computation of generic tensors," *Linear algebra and its applications*, vol. 52, pp. 645–685, 1983.

[41] I. Domanov and L. De Lathauwer, "Canonical polyadic decomposition of third-order tensors: Relaxed uniqueness conditions and algebraic algorithm," *Linear Algebra and its Applications*, vol. 513, no. 15, pp. 342–375, Jan 2017.

[42] I. Domanov and L. De Lathauwer, "On the uniqueness of the canonical polyadic decomposition of third-order tensors — part ii: Uniqueness of the overall decomposition," *SIAM Journal on Matrix Analysis and Applications (SIMAX)*, vol. 34, no. 3, pp. 876–903, 2013.

[43] I. Domanov and L. De Lathauwer, "Canonical polyadic decomposition of third-order tensors: reduction to generalized eigenvalue decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 35, no. 2, pp. 636–660, 2014.

[44] H. Derksen, "Kruskals uniqueness inequality is sharp," *Linear Algebra Appl*, vol. 438, no. 2, pp. 708–712, 2013.

[45] A. Stegeman and N.D. Sidiropoulos, "On Kruskal's uniqueness condition for the CANDECOMP/PARAFAC decomposition," *Linear Algebra and its Applications*, vol. 420, no. 2-3, pp. 540–552, 2007.

[46] L. Chiantini, G. Ottaviani, and N. Vannieuwenhoven, "On generic identifiability of symmetric tensors of subgeneric rank," *Trans. Amer. Math. Soc.*, 2016.

[47] L. Chiantini, G. Ottaviani, and N. Vannieuwenhoven, "An algorithm for generic and low-rank specific identifiability of complex tensors," *SIAM Journal on Matrix Analysis and Applications*, vol. 35, no. 4, pp. 1265–1287, 2014.

[48] M. Sørensen and L. De Lathauwer, "New uniqueness conditions for the canonical polyadic decomposition of third-order tensors," *SIAM Journal on Matrix Analysis and Applications*, vol. 36, no. 4, pp. 1381–1403, 2015.

[49] M. Sorensen, L. De Lathauwer, P. Comon, S. Icart, and L. Deneire, "Canonical Polyadic Decomposition with a Columnwise Orthonormal Factor Matrix," *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 4, Oct.-Dec., pp. 1190–1213, 2012.

[50] N.D. Sidiropoulos and R. Bro, "On the uniqueness of multilinear decomposition of N-way arrays," *Journal of chemometrics*, vol. 14, no. 3, pp. 229–239, 2000.

[51] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2000.

[52] T. Kolda, "A counterexample to the possibility of an extension of the eckart–young low-rank approximation theorem for the orthogonal rank tensor decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 24, no. 3, pp. 762–767, 2003.

[53] 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 Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1324–1342, 2000.

[54] C. J. Hillar and L.-H. Lim, "Most tensor problems are NP-hard," *Journal of the ACM*, vol. 60, no. 6, pp. 45, 2013.

[55] P.M. Kroonenberg, *Applied multiway data analysis*, Wiley, 2008.

[56] N.D. Sidiropoulos and A. Kyrillidis, "Multi-way compressed sensing for sparse low-rank tensors," *Signal Processing Letters, IEEE*, vol. 19, no. 11, pp. 757–760, Nov 2012.

[57] N. D. Sidiropoulos, E. E. Papalexakis, and C. Faloutsos, "Parallel randomly compressed cubes : A scalable distributed architecture for big tensor decomposition," *IEEE Signal Process. Mag.*, vol. 31, no. 5, pp. 57–70, Sep. 2014.

[58] I. Oseledets, D. Savostianov, and E. Tyrtyshnikov, "Tucker dimensionality reduction of three-dimensional arrays in linear time," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 3, pp. 939–956, 2008.

[59] M. Mahoney, M. Maggioni, and P. Drineas, "Tensor-cur decompositions for tensor-based data," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 3, pp. 957–987, 2008.

[60] C. Caiafa and A. Cichocki, "Generalizing the column–row matrix decomposition to multi-way arrays," *Linear Algebra and its Applications*, vol. 433, no. 3, pp. 557–573, 2010.

[61] S. Goreinov, E. Tyrtyshnikov, and N. Zamarashkin, "A theory of pseudoskeleton approximations," *Linear Algebra and its Applications*, vol. 261, no. 1, pp. 1–21, 1997.

[62] B. Savas and L. Eldén, "Krylov-type methods for tensor computations i," *Linear Algebra and its Applications*, vol. 438, no. 2, pp. 891–918, 2013.

[63] J. Douglas 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, no. 1, pp. 3–24, 1980.

[64] R. Bro and C. Andersson, "Improving the speed of multiway algorithms: Part ii: Compression," *Chemometrics and Intelligent Laboratory Systems*, vol. 42, no. 12, pp. 105 – 113, 1998.

[65] N. Vervliet, O. Debals, and L. De Lathauwer, "Tensorlab 3.0 — numerical optimization strategies for large-scale constrained and coupled matrix/tensor factorization," in *2016 Conference Record of the 50th Asilomar Conference on Signals, Systems and Computers*. IEEE, 2016.

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

[67] L. De Lathauwer, "Decompositions of a higher-order tensor in block terms-part ii: Definitions and uniqueness," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 3, pp. 1033–1066, 2008.

[68] L. De Lathauwer, "Blind separation of exponential polynomials and the decomposition of a tensor in rank-(l_r,l_r,1) terms," *SIAM Journal*

on Matrix Analysis and Applications, vol. 32, no. 4, pp. 1451–1474, 2011.

[69] R. Bro, R. Harshman, N. Sidiropoulos, and M. Lundy, "Modeling multi-way data with linearly dependent loadings," *Journal of Chemometrics*, vol. 23, no. 7-8, pp. 324–340, 2009.

[70] A. Stegeman and A. de Almeida, "Uniqueness conditions for constrained three-way factor decompositions with linearly dependent loadings," *SIAM Journal on Matrix Analysis and Applications*, vol. 31, no. 3, pp. 1469–1490, 2010.

[71] A. Stegeman and T. Lam, "Improved uniqueness conditions for canonical tensor decompositions with linearly dependent loadings," *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 4, pp. 1250–1271, 2012.

[72] X. Guo, S. Miron, D. Brie, and A. Stegeman, "Uni-mode and partial uniqueness conditions for candecomp/parafac of three-way arrays with linearly dependent loadings," *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 1, pp. 111–129, 2012.

[73] L. Sorber, M. Van Barel, and L. De Lathauwer, "Structured data fusion," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 4, pp. 586–600, June 2015.

[74] D. Lahat, T. Adali, and C. Jutten, "Multimodal data fusion: an overview of methods, challenges, and prospects," *Proceedings of the IEEE*, vol. 103, no. 9, pp. 1449–1477, 2015.

[75] R. Harshman, "Parafac2: Mathematical and technical notes," *UCLA working papers in phonetics*, vol. 22, no. 3044, pp. 122215, 1972.

[76] M. Sørensen and L. De Lathauwer, "Coupled canonical polyadic decompositions and (coupled) decompositions in multilinear rank-(l_r,n,l_r,n,1) terms—part i: Uniqueness," *SIAM Journal on Matrix Analysis and Applications*, vol. 36, no. 2, pp. 496–522, 2015.

[77] B. W. Bader and T. G. Kolda, "Efficient MATLAB computations with sparse and factored tensors," *SIAM Journal on Scientific Computing*, vol. 30, no. 1, pp. 205–231, December 2007.

[78] N. Vannieuwenhoven, K. Meerbergen, and R Vandebril, "Computing the Gradient in Optimization Algorithms for the CP Decomposition in Constant Memory through Tensor Blocking," *SIAM Journal on Scientific Computing*, vol. 37, no. 3, pp. C415–C438, 2015.

[79] A. H. Phan, P. Tichavsky, and A. Cichocki, "Fast Alternating LS Algorithms for High Order CANDECOMP/PARAFAC Tensor Factorizations," *IEEE Transactions on Signal Processing*, vol. 61, no. 19, pp. 4834–4846, Oct. 2013, doi: 10.1109/TSP.2013.2269903.

[80] U Kang, E. Papalexakis, A. Harpale, and C. Faloutsos, "Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 316–324.

[81] N. Ravindran, N.D. Sidiropoulos, S. Smith, and G. Karypis, "Memory-efficient parallel computation of tensor and matrix products for big tensor decomposition," in *Asilomar Conference on Signals, Systems, and Computers*, 2014, pp. 581–585.

[82] J. H. Choi and S. V. N. Vishwanathan, "DFacTo: Distributed factorization of tensors," in *Advances in Neural Information Processing Systems*, 2014, pp. 1296–1304.

[83] S. Smith, N. Ravindran, N. D. Sidiropoulos, and G. Karypis, "SPLATT: Efficient and parallel sparse tensor-matrix multiplication," in *IEEE International Parallel and Distributed Processing Symposium*, 2015.

[84] M. J. Mohlenkamp, "Musings on multilinear fitting," *Linear Algebra and its Applications*, vol. 438, no. 2, pp. 834–852, 2013.

[85] L. Zhening, A. Uschmajew, and S. Zhang, "On convergence of the maximum block improvement method," *SIAM Journal on Optimization*, vol. 25, no. 1, pp. 210–233, 2015.

[86] M. Razaviyayn, M. Hong, and Z.-Q. Luo, "A unified convergence analysis of block successive minimization methods for nonsmooth optimization," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 1126–1153, 2013.

[87] A. Uschmajew, "Local convergence of the alternating least squares algorithm for canonical tensor approximation," *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 2, pp. 639–652, 2012.

[88] W. Espig, M. Hackbusch and A. Khachatryan, "On the convergence of alternating least squares optimisation in tensor format representations," arXiv preprint arXiv:1506.00062, 2015.

[89] L. Sorber, M. Van Barel, and L. De Lathauwer, "Optimization-based algorithms for tensor decompositions: canonical polyadic decomposition, decomposition in rank-(Lr,Lr,1) terms and a new generalization," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 695–720, Apr. 2013.

[90] J. Nocedal and S. Wright, *Numerical optimization*, Springer Science & Business Media, 2006.

[91] G. Tomasi and R. Bro, "A comparison of algorithms for fitting the PARAFAC model," *Computational Statistics and Data Analysis*, vol. 50, no. 7, pp. 1700–1734, 2006.

[92] M. Ishteva, P.-A. Absil, S. Van Huffel, and L. De Lathauwer, "Best low multilinear rank approximation of higher-order tensors, based on the riemannian trust-region scheme," *SIAM Journal on Matrix Analysis and Applications*, vol. 32, no. 1, pp. 115–135, 2011.

[93] B. Savas and L.-H. Lim, "Quasi-newton methods on grassmannians and multilinear approximations of tensors," *SIAM Journal on Scientific Computing*, vol. 32, no. 6, pp. 3352–3393, 2010.

[94] M. Rajih, P. Comon, and R. Harshman, "Enhanced line search: A novel method to accelerate parafac," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 3, pp. 1128–1147, 2008.

[95] L. Sorber, I. Domanov, M. Van Barel, and L. De Lathauwer, "Exact Line and Plane Search for Tensor Optimization," *Computational Optimization and Applications*, vol. 63, no. 1, pp. 121–142, Jan. 2016.

[96] G. Tomasi and R. Bro, "PARAFAC and missing values," *Chemometrics and Intelligent Laboratory Systems*, vol. 75, no. 2, pp. 163–180, 2005.

[97] A. Beutel, P. Talukdar, A. Kumar, C. Faloutsos, E. Papalexakis, and E. Xing, *FlexiFaCT: Scalable Flexible Factorization of Coupled Tensors on Hadoop*, chapter 13, pp. 109–117.

[98] N. Vervliet and L. De Lathauwer, "A randomized block sampling approach to canonical polyadic decomposition of large-scale tensors," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 2, pp. 284–295.

[99] C. Papalexakis, E.and Faloutsos and N. Sidiropoulos, "Parcube: Sparse parallelizable tensor decompositions," in *Proceedings of the 2012 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part I*, Berlin, Heidelberg, 2012, ECML PKDD'12, pp. 521–536, Springer-Verlag.

[100] R. Bro and N.D. Sidiropoulos, "Least squares regression under unimodality and non-negativity constraints," *Journal of Chemometrics*, vol. 12, pp. 223–247, 1998.

[101] A. P. Liavas and N. D. Sidiropoulos, "Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers," *IEEE Transactions on Signal Processing*, vol. 63, no. 20, pp. 5450–5463, Oct 2015.

[102] K. Huang, N. D. Sidiropoulos, and A. P. Liavas, "A flexible and efficient algorithmic framework for constrained matrix and tensor factorization," *IEEE Transactions on Signal Processing*, vol. 64, no. 19, pp. 5052 – 5065, Oct. 2016.

[103] S. P. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[104] N. Parikh and S. P. Boyd, "Proximal algorithms," *Foundations and Trends® in Optimization*, vol. 1, no. 3, pp. 123–231, 2014.

[105] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, "Efficient projections onto the l_1-ball for learning in high dimensions," in *Proc. ACM ICML*, 2008, pp. 272–279.

[106] X. Fu, K. Huang, W. K. Ma, N. D. Sidiropoulos, and R. Bro, "Joint tensor factorization and outlying slab suppression with applications," *IEEE Transactions on Signal Processing*, vol. 63, no. 23, pp. 6315–6328, Dec 2015.

[107] E.E. Papalexakis, N.D. Sidiropoulos, and R. Bro, "From k -means to higher-way co-clustering: Multilinear decomposition with sparse latent factors," *IEEE Trans. on Signal Processing*, vol. 61, no. 2, pp. 493–506, 2013.

[108] S. Basu and Y. Bresler, "The stability of nonlinear least squares problems and the Cramér-Rao bound," *IEEE Transactions on Signal Processing*, vol. 48, no. 12, pp. 3426–3436, 2000.

[109] J. R. Magnus and H. Neudecker, "The commutation matrix: some properties and applications," *The Annals of Statistics*, pp. 381–394, 1979.

[110] X. Liu and N. D. Sidiropoulos, "Cramér-Rao lower bounds for low-rank decomposition of multidimensional arrays," *IEEE Transactions on Signal Processing*, vol. 49, no. 9, pp. 2074–2086, 2001.

[111] S. A. Vorobyov, Y. Rong, N. D. Sidiropoulos, and A. B. Gershman, "Robust iterative fitting of multilinear models," *IEEE Transactions on Signal Processing*, vol. 53, no. 8, pp. 2678–2689, 2005.

[112] G. Tomasi, *Practical and computational aspects in chemometric data analysis*, Ph.D. thesis, 2006.

[113] A.-H. Phan, P. Tichavsky, and A. Cichocki, "Low complexity damped Gauss–Newton algorithms for CANDECOMP/PARAFAC," *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 1, pp. 126–147, 2013.

[114] P. Tichavsky, A.-H. Phan, and Z. Koldovsky, "Cramér-Rao-induced bounds for CANDECOMP/PARAFAC tensor decomposition," *IEEE Transactions on Signal Processing*, vol. 61, no. 8, pp. 1986–1997, 2013.

[115] P. Stoica and T. L. Marzetta, "Parameter estimation problems with singular information matrices," *IEEE Transactions on Signal Processing*, vol. 49, no. 1, pp. 87–90, 2001.

[116] K. Huang and N. D. Sidiropoulos, "Putting nonnegative matrix factorization to the test: a tutorial derivation of pertinent Cramér-Rao bounds and performance benchmarking," *IEEE Signal Processing Magazine*, vol. 31, no. 3, pp. 76–86, May 2014.

[117] N. D. Sidiropoulos and G. Z. Dimic, "Blind multiuser detection in W-CDMA systems with large delay spread," *IEEE Signal Processing Letters*, vol. 8, no. 3, pp. 87–89, March 2001.

[118] D. Nion and N.D. Sidiropoulos, "Adaptive algorithms to track the PARAFAC decomposition of a third-order tensor," *IEEE Transactions on Signal Processing*, vol. 57, no. 6, pp. 2299–2310, 2009.

[119] A. Belouchrani, K. Abed-Meraim, J. F. Cardoso, and E. Moulines, "A blind source separation technique using second-order statistics," *IEEE Transactions on Signal Processing*, vol. 45, no. 2, pp. 434–444, Feb 1997.

[120] P. Comon, , and C. Jutten, Eds., *Handbook of Blind Source Separation: Independent Component Analysis and Applications*, Academic Press, Oxford, 2010.

[121] X. Fu, N. D. Sidiropoulos, J. H. Tranter, and W. K. Ma, "A factor analysis framework for power spectra separation and multiple emitter localization," *IEEE Transactions on Signal Processing*, vol. 63, no. 24, pp. 6581–6594, Dec 2015.

[122] N. D. Sidiropoulos, "Generalizing caratheodory's uniqueness of harmonic parameterization to n dimensions," *IEEE Transactions on Information Theory*, vol. 47, no. 4, pp. 1687–1690, May 2001.

[123] T. Jiang, N.D. Sidiropoulos, and J.M.F. ten Berge, "Almost sure identifiability of multidimensional harmonic retrieval," *IEEE Transactions on Signal Processing*, vol. 49, no. 9, pp. 1849–1859, 2001.

[124] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM Review*, vol. 52, no. 3, pp. 471–501, 2010.

[125] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. Carbonell, *Temporal Collaborative Filtering with Bayesian Probabilistic Tensor Factorization*, chapter 18, pp. 211–222.

[126] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver, "Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering," in *Proceedings of the Fourth ACM Conference on Recommender Systems*, New York, NY, USA, 2010, RecSys '10, pp. 79–86, ACM.

[127] E. Acar, T. Kolda, and D. Dunlavy, "All-at-once optimization for coupled matrix and tensor factorizations," in *MLG'11: Proceedings of Mining and Learning with Graphs*, August 2011.

[128] E. Acar, M. Rasmussen, F. Savorani, T. Ns, and R. Bro, "Understanding data fusion within the framework of coupled matrix and tensor factorizations," *Chemometrics and Intelligent Laboratory Systems*, vol. 129, pp. 53 – 63, 2013, Multiway and Multiset Methods.

[129] E. E. Papalexakis, T. Mitchell, N. D. Sidiropoulos, C. Faloutsos, P. P. Talukdar, and B. Murphy, "Turbo-SMT: Accelerating Coupled Sparse Matrix-Tensor Factorizations by 200x," in *Proc. SIAM Conference on Data Mining (SDM)*, April 24-26 2014.

[130] S. Rendle, "Factorization machines," in *2010 IEEE International Conference on Data Mining*, Dec 2010, pp. 995–1000.

[131] D. Hsu and S. Kakade, "Learning mixtures of spherical gaussians: Moment methods and spectral decompositions," in *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, New York, NY, USA, 2013, ITCS '13, pp. 11–20, ACM.

[132] A. Anandkumar, D. Hsu, M. Janzamin, and S. Kakade, "When are overcomplete topic models identifiable? uniqueness of tensor tucker decompositions with structured sparsity," *CoRR*, vol. abs/1308.2853, 2013.

[133] S. Yan, D. Xu, Q. Yang, L. Zhang, X. Tang, and H. J. Zhang, "Multilinear discriminant analysis for face recognition," *IEEE Transactions on Image Processing*, vol. 16, no. 1, pp. 212–220, Jan 2007.

[134] H. Lu, K. Plataniotis, and A. Venetsanopoulos, "A survey of multilinear subspace learning for tensor data," *Pattern Recogn.*, vol. 44, no. 7, pp. 1540–1551, July 2011.

[135] T. Kolda and B. Bader, "The TOPHITS model for higher-order web link analysis," in *Proceedings of Link Analysis, Counterterrorism and Security 2006*, 2006.

[136] E. Papalexakis, K. Pelechrinis, and C. Faloutsos, "Spotting misbehaviors in location-based social networks using tensors," in *Proceedings of the 23rd International Conference on World Wide Web*, New York, NY, USA, 2014, WWW '14 Companion, pp. 551–552, ACM.

[137] C. A Andersson and R. Bro, "The N-way toolbox for matlab," *Chemometrics and Intelligent Laboratory Systems*, vol. 52, no. 1, pp. 1–4, 2000.

[138] B. W. Bader and T. G. Kolda, "Efficient MATLAB computations with sparse and factored tensors," *SIAM Journal on Scientific Computing*, vol. 30, no. 1, pp. 205–231, 2007.

[139] B. W. Bader, T. G. Kolda, et al., "Matlab tensor toolbox version 2.6," http://www.sandia.gov/~tgkolda/TensorToolbox/, February 2015.

[140] N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer, "Tensorlab v3.0," http://www.tensorlab.net/, Mar. 2016.

[141] F. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *Journal of Mathematics and Physics*, vol. 6, no. 1-4, pp. 164–189, 1927.

[142] R. Cattell, ""parallel proportional profiles" and other principles for determining the choice of factors by rotation," *Psychometrika*, vol. 9, no. 4, pp. 267–283, 1944.

[143] J.D. Carroll and J.J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of Eckart-Young decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.

[144] A. Yeredor, "Non-orthogonal joint diagonalization in the least-squares sense with application in blind source separation," *IEEE Transactions on Signal Processing*, vol. 50, no. 7, pp. 1545–1553, Jul 2002.

[145] S. M. Kay, *Fundamentals of Statistical Signal Processing, Volume I: Estimation Theory*, Prentice-Hall, 1993.

[146] J. D. Gorman and A. O. Hero, "Lower bounds for parametric estimation with constraints," *IEEE Transactions on Information Theory*, vol. 36, no. 6, pp. 1285–1301, 1990.

[147] P. Stoica and B. C. Ng, "On the Cramér-Rao bound under parametric constraints," *IEEE Signal Processing Letters*, vol. 5, no. 7, pp. 177–179, 1998.

[148] Z. Ben-Haim and Y. C. Eldar, "On the constrained Cramér-Rao bound with a singular Fisher information matrix," *IEEE Signal Processing Letters*, vol. 16, no. 6, pp. 453–456, 2009.

[149] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge University Press, 2004.

[150] C. Qian, N. D. Sidiropoulos, K. Huang, L. Huang, and H.-C. So, "Least squares phase retrieval using feasible point pursuit," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2016.

[151] C. Qian, N. D. Sidiropoulos, K. Huang, L. Huang, and H.-C. So, "Phase retrieval using feasible point pursuit: Algorithms and Cramér-Rao bound," *IEEE Transactions on Signal Processing*, vol. 64, no. 20, pp. 5282–5296, Oct. 2016.

[152] K. Huang, Y. C. Eldar, and N. D. Sidiropoulos, "On Convexity and Identifiability in 1-D Fourier Phase Retrieval," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2016.

[153] K. Huang, Y. C. Eldar, and N. D. Sidiropoulos, "Phase Retrieval from 1D Fourier Measurements: Convexity, Uniqueness, and Algorithms," *IEEE Transactions on Signal Processing*, vol. 64, no. 23, pp. 6105–6117, Dec. 2016.

[154] D. P. Bertsekas, *Nonlinear programming*, Athena Scientific, 2nd edition, 1999.

[155] A. Swami, "Cramér-Rao bounds for deterministic signals in additive and multiplicative noise," *Signal Processing*, vol. 53, no. 2, pp. 231–244, 1996.

[156] A. Swami and B. M. Sadler, "On some detection and estimation problems in heavy-tailed noise," *Signal Processing*, vol. 82, no. 12, pp. 1829–1846, 2002.

[157] K.B. Petersen and M. S. Pedersen, *The matrix cookbook*, Technical University of Denmark, 2006.

[158] K. Huang, N. D. Sidiropoulos, and A. Swami, "Non-negative matrix factorization revisited: Uniqueness and algorithm for symmetric decomposition," *IEEE Trans. on Signal Processing*, vol. 62, no. 1, pp. 211–224, Jan 2014.