Low-Rank Tensor Networks for Dimensionality Reduction and Large-Scale Optimization Problems: Perspectives and Challenges PART 1

A. CICHOCKI *, N. LEE,

I.V. OSELEDETS, A-H. PHAN, Q. ZHAO, D. MANDIC

* RIKEN BSI, Japan and Polish Academy of Scoence, Poland,
e-mail: a.cichocki@riken.jp

Abstract

Machine learning and data mining algorithms are becoming increasingly important in analyzing large volume, multi-relational and multi–modal datasets, which are often conveniently represented as multiway arrays or tensors. It is therefore timely and valuable for the multidisciplinary research community to review tensor decompositions and tensor networks as emerging tools for large-scale data analysis and data mining. We provide the mathematical and graphical representations and interpretation of tensor networks, with the main focus on the Tucker and Tensor Train (TT) decompositions and their extensions or generalizations.

To make the material self-contained, we also address the concept of tensorization which allows for the creation of very high-order tensors from lower-order structured datasets represented by vectors or matrices. Then, in order to combat the curse of dimensionality and possibly obtain linear or even sub-linear complexity of storage and computation, we address super-compression of tensor data through low-rank tensor networks. Finally, we demonstrate how such approximations can be used to solve a wide class of huge-scale linear/multilinear dimensionality reduction and related optimization problems that are far from being tractable when using classical numerical methods.

The challenge for huge-scale optimization problems is therefore to develop methods which scale linearly or sub-linearly (i.e., logarithmic complexity) with the size of datasets, in order to benefit from the well– understood optimization frameworks for smaller size problems. However, most efficient optimization algorithms are convex and do not scale well with data volume, while linearly scalable algorithms typically only apply to very specific scenarios. In this review, we address this problem through

the concepts of low-rank tensor network approximations, distributed tensor networks, and the associated learning algorithms. We then elucidate how these concepts can be used to convert otherwise intractable huge-scale optimization problems into a set of much smaller linked and/or distributed sub-problems of affordable size and complexity. In doing so, we highlight the ability of tensor networks to account for the couplings between the multiple variables, and for multimodal, incomplete and noisy data.

The methods and approaches discussed in this work can be considered both as an alternative and a complement to emerging methods for huge-scale optimization, such as the random coordinate descent (RCD) scheme, subgradient methods, alternating direction method of multipliers (ADMM) methods, and proximal gradient descent methods This is PART1 which consists of Sections 1-4 [1].

Keywords: Tensor networks, Function-related tensors, CP decomposition, Tucker models, tensor train (TT) decompositions, matrix product states (MPS), matrix product operators (MPO), basic tensor operations, multiway component analysis, multilinear blind source separation, tensor completion, linear/multilinear dimensionality reduction, large-scale optimization problems, symmetric eigenvalue decomposition (EVD), PCA/SVD, huge systems of linear equations, pseudo-inverse of very large matrices, Lasso and Canonical Correlation Analysis (CCA).

1 Introduction and Motivation

This monograph aims to present a coherent account of ideas and methodologies for a wide class of huge-scale optimization problems, whereby for mathematical and computational tractability, the corresponding cost (loss) functions are approximately represented in highly compressed and distributed formats. This is achieved by virtue of the underlying tensor decompositions (TDs), which separate complex data tensors of exceedingly high dimensionality into their factor (component) matrices and a single core tensor, and tensor networks (TNs) which decompose higher-order tensors into sparsely interconnected low-scale factor matrices and/or low-order core tensors. These low-order core tensors are called "components", "blocks", "factors" or simply "cores".

In this monograph, the TDs and TNs are treated in a unified way by considering TDs as simple tensor networks or sub-networks; the terms "tensor decompositions" and "tensor networks" will therefore be used interchangeably. The resultant tensor networks are treated as special graph structures which break down high-order tensors into a set of sparsely interconnected low-order core tensors, thus allowing both enhanced interpretation and computational advantages. Such an approach is inspired by many application contexts which require the computation of the eigenvalues and corresponding eigenvectors of extremely high-dimensional linear or nonlinear operators which describe the coupling between the many degrees of freedom of real-world physical systems; such degrees of freedom are often only weakly coupled. Indeed, quantum physics provides evidence that couplings between multiple data channels usually do not occur among all the degrees of freedom but mostly locally, whereby "relevant" information, of relatively low-dimensionality, is embedded into very large-dimensional measurements [194, 201, 232, 267].

Tensor networks offer the theoretical and computational framework for computationally prohibitive large volumes of data, by "dissecting" such data into the "relevant" and "irrelevant" information, both of lower dimensionality. In this way, tensor network representations often allow for super-compression of datasets as large as 10^{50} entries, down to the affordable levels of 10^7 entries or even less [28, 85, 86, 138, 140, 155, 173, 209, 269].

With the emergence of the big data paradigm, it is therefore both timely and important to provide the multidisciplinary machine learning and data analytic communities with a comprehensive overview of tensor networks, together with an example-rich guidance on their application in contents-specific optimization problems for huge-scale structured data. In this way, our aim is also to unify the terminology, notation, and algorithms for tensor decompositions and tensor networks which are increasingly used not only in machine learning, signal processing, numerical analysis and scientific computing, but also in quantum physics/chemistry for the representation of e.g., quantum many-body systems.

1.1 Challenges in Big Data Processing

The sheer volume and structural complexity of modern datasets may be very high, to an extent which renders standard analysis methods and algorithms inadequate. Apart from Volume, the "V" features which characterize big data include Veracity, Variety and Velocity. Each of the V features represents a research challenge on its own, for example, high volume implies the need for algorithms that are scalable; high Velocity requires the processing of big data streams in near real-time; high Veracity calls for robust and predictive algorithms for noisy, incomplete and/or inconsistent data; high Variety demands the fusion of different data types, e.g., continuous, discrete, binary, time series, images, video, text, probabilistic or multi-view (see Figure 1 (a) and (b)). In some applications, additional V challenges may arise, such as Visualization, Variability and Value. The Value is particularly interesting and refers to the extraction of high quality and consistent data, capable of yielding meaningful and interpretable results.

Owing to the increasingly affordable recording devices and extreme-scale data volumes and variety provided by remote sensing, multidimensional data is becoming ubiquitous across the science and engineering disciplines. In the case of multimedia (speech, video) and medical/biological data, the analysis also requires a paradigm shift in order to efficiently process massive datasets within tolerable time (velocity). Such massive datasets may have billions of entries and are typically represented in the form of huge block matrices and/or tensors. As a result, this has spurred a renewed interest in the development of matrix/tensor algorithms that are suitable for very large-scale datasets. We show that, tensor networks provide a natural sparse and distributed representation for big data and address both established and emerging methodologies for tensor-based representations and optimization. These include low-rank tensor network representations, which allow for huge data tensors to be approximated (compressed) by interconnected low-order core tensors.

1.2 Tensor Notations and Graphical Representations

Tensors are multi-dimensional generalizations of matrices; a matrix (2nd-order tensor) has two modes, rows and columns, while an Nth-order tensor has N modes (see Figures 2 – 7). For example, a 3rd-order tensor (with three-modes) looks like a cube (see Figure 2). Subtensors are formed when a subset of tensor indices is fixed. Of particular interest are *fibers* which are vectors defined by fixing every index but one, and *matrix slices* which are two-dimensional sections (matrices) of a tensor, obtained by fixing all the tensor indices but two. It should also be noted that block matrices can be represented by tensors, as illustrated in Figure 3 for 4th-order tensors.

We adopt the notation in which tensors (for $N \geqslant 3$) are denoted by bold underlined capital letters, e.g., $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. For simplicity, we assume that all tensors are real-valued, but it is, of course, possible to define tensors as complex-valued or over arbitrary fields. Matrices are denoted by bold-

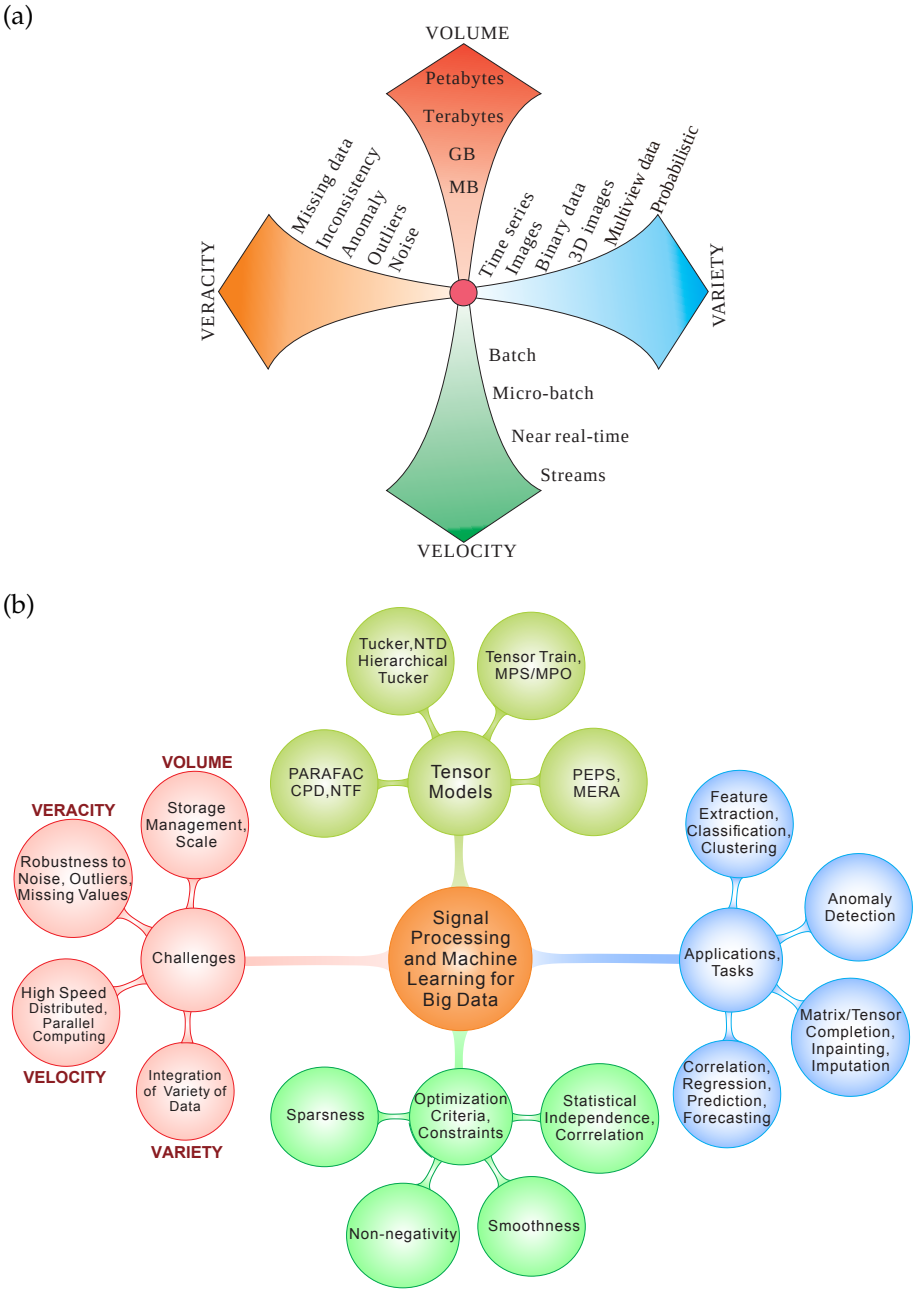


Figure 1: A framework for extremely large-scale data analysis. (a) The 4V challenges for big data. (b) Tensor networks for the 4V challenges and their potential applications.

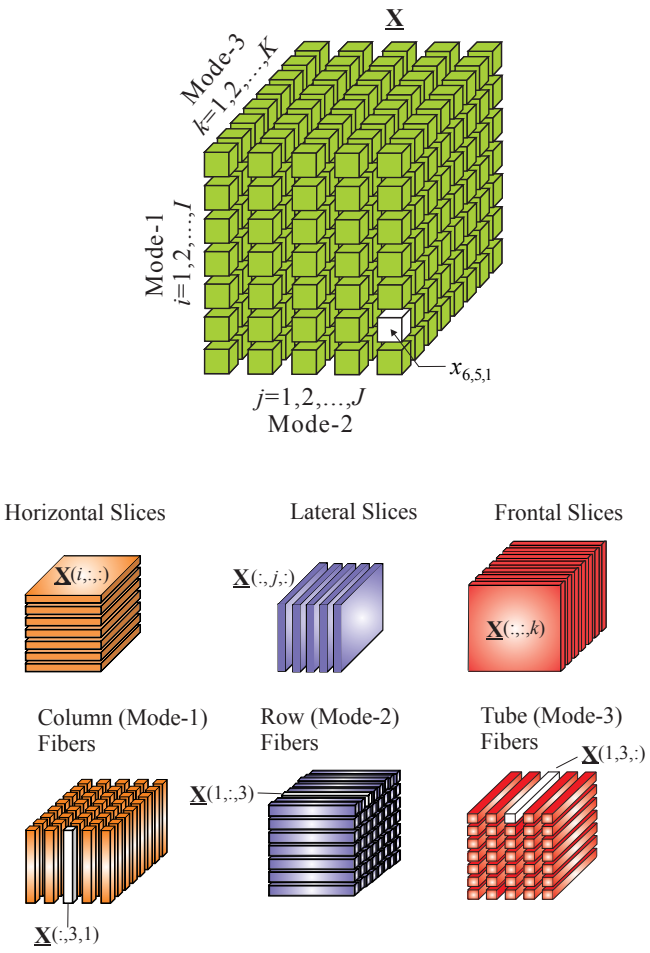


Figure 2: A 3rd-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$, with entries $x_{i,j,k} = \underline{\mathbf{X}}(i,j,k)$ and its sub-tensors: slices (middle) and fibers (bottom). All fibers are treated as column vectors.

face capital letters, e.g., $\mathbf{X} \in \mathbb{R}^{I \times J}$, and vectors (1st-order tensors) by boldface lowercase letters e.g., $\mathbf{x} \in \mathbb{R}^{J}$; for instance, the columns of the matrix $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_R] \in \mathbb{R}^{I \times R}$ are the vectors denoted by $\mathbf{a}_r \in \mathbb{R}^{I}$, and the elements of a matrix (scalars) are denoted by lowercase letters, e.g., $a_{ir} = \mathbf{A}(i,r)$ (see Table 1).

A specific entry of an Nth-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is denoted by $x_{i_1,i_2,\ldots,i_N} = \underline{\mathbf{X}}(i_1, i_2, \ldots, i_N) \in \mathbb{R}$. The term "size" stands for the number of values that an index can take in a particular mode. For example, the tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is of order N and size I_n in each of its n modes $(n = 1, 2, \ldots, N)$.

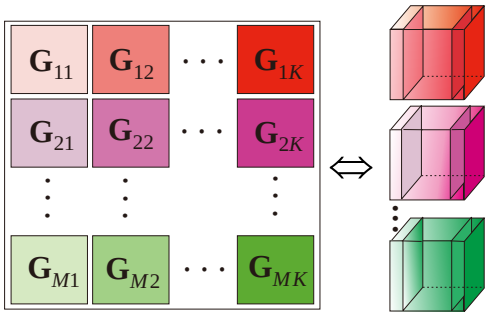


Figure 3: A block matrix and its representation as a 4th-order tensor, created by reshaping or projection of row blocks as lateral slices of 3-rd-order tensors.

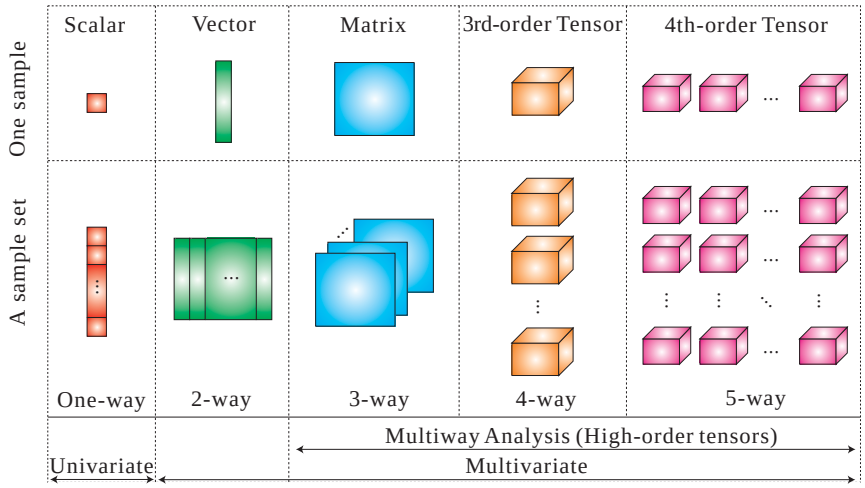


Figure 4: Graphical representation of multiway array (tensor) data of increasing complexity, both for a single sample and for a set of samples (see [200] for more detail).

Lower-case letters e.g, $i, j, \ldots$ are used for the subscripts in running indices and capital letters $I, J, \ldots$ denote the upper bound of an index, i.e., $i = 1, 2, \ldots, I$ and $j = 1, 2, \ldots, J$. For a positive integer n, the shorthand notation $< n >$ is used to denote the set of indices $\{1, 2, \ldots, n\}$. To summarize, the order of a tensor is the number of its "modes", "ways" or "dimensions", which can include space, time, frequency, trials, classes, and dictionaries.

Notations and terminology used for tensors and tensors networks differ across the scientific communities (see Table 2); to this end we employ a unifying notation particularly suitable for machine learning and signal processing

Table 1: Basic matrix/tensor notation and symbols.

$\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$	Nth-order tensor of size $I_1 \times \cdots \times I_N$
$x_{i_1,i_2\ldots,i_N} = \underline{\mathbf{X}}(i_1, i_2 \ldots, i_N)$	$(i_1, i_2 \ldots, i_N)$th entry of $\underline{\mathbf{X}}$
x, $\mathbf{x}$, $\mathbf{X}$	scalar, vector and matrix
$\underline{\mathbf{G}}$, $\underline{\mathbf{S}}$, $\underline{\mathbf{G}}^{(n)}$, $\underline{\mathbf{X}}^{(n)}$	core tensors
$\underline{\mathbf{\Lambda}} \in \mathbb{R}^{R \times R \times \cdots \times R}$	Nth-order diagonal core tensor with nonzero entries λ_r on the main diagonal
$\mathbf{A}^T$, $\mathbf{A}^{-1}$, $\mathbf{A}^\dagger$	transpose, inverse and Moore-Penrose pseudo-inverse of a matrix $\mathbf{A}$
$\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_R] \in \mathbb{R}^{I \times R}$	matrix with column vectors $\mathbf{a}_r \in \mathbb{R}^I$ and entries a_{ir}
$\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{A}^{(n)}$, $\mathbf{B}^{(n)}$, $\mathbf{U}^{(n)}$	component (factor) matrices
$\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times I_1 \cdots I_{n-1} I_{n+1} \cdots I_N}$	mode-n matricization of $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$
$\mathbf{X}_{<n>} \in \mathbb{R}^{I_1 I_2 \cdots I_n \times I_{n+1} \cdots I_N}$	$(1, \ldots, n)$th matricization of $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$
$\underline{\mathbf{X}}(:, i_2, i_3, \ldots, i_N) \in \mathbb{R}^{I_1}$	mode-1 fiber of a tensor $\underline{\mathbf{X}}$ obtained by fixing all indices but one (a vector)
$\underline{\mathbf{X}}(:, :, i_3, \ldots, i_N) \in \mathbb{R}^{I_1 \times I_2}$	slice of a tensor $\underline{\mathbf{X}}$ obtained by fixing all indices but two (a matrix)
$\underline{\mathbf{X}}(:, :, :, i_4, \ldots, i_N)$	subtensor of $\underline{\mathbf{X}}$, obtained by fixing several indices
R, $(R_1, \ldots, R_N)$	tensor rank R and multilinear rank
$\circ$, $\odot$, $\otimes$	outer, Khatri-Rao, Kronecker products
$\otimes_L$, $\boxtimes$	Left Kronecker, strong Kronecker products
$\mathbf{x} = \text{vec}(\underline{\mathbf{X}})$	vectorization of $\underline{\mathbf{X}}$
$\text{tr}(\bullet)$	trace of a square matrix
$\text{diag}(\bullet)$	diagonal matrix

Table 2: Terminology for tensor networks across the machine learning/scientific computing and Quantum Physics/Chemistry communities.

Machine Learning	Quantum Physics
Nth-order tensor	rank-N tensor
high/low-order tensor	tensor of high/low dimension
ranks of TNs	bond dimensions of TNs
unfolding, matricization	grouping of indices
tensorization	splitting of indices
core tensors	particles
core	site
physical variables	open indices
ALS Algorithm	one-site DMRG or DMRG1
MALS Algorithm	two-site DMRG or DMRG2
column vector $\mathbf{x} \in \mathbb{R}^{I \times 1}$	ket $\|\Psi\rangle$
row vector $\mathbf{x}^T \in \mathbb{R}^{1 \times I}$	bra $\langle\Psi\|$
inner product $\langle \mathbf{x}, \mathbf{x} \rangle = \mathbf{x}^T \mathbf{x}$	$\langle\Psi\|\Psi\rangle$
Tensor Train (TT)	Matrix product states (MPS) with Open Boundary Conditions (OPC)
Tensor Chain (TC)	MPS with Periodic Boundary Conditions (PBC)
Matrix TT	Matrix Product Operators with OPB
Hierarchical Tucker (HT)	Tensor Tree Network (TTN) with rank-3 tensors

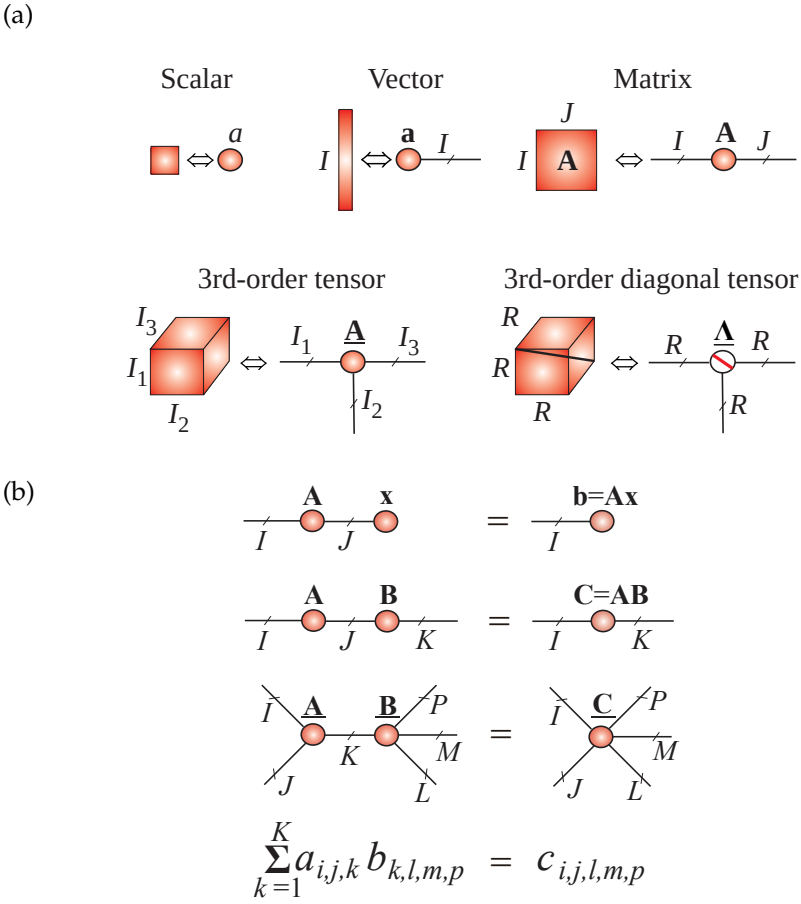


Figure 5: (a) Basic building blocks for tensor network diagrams. (b) Tensor network diagrams for matrix-vector multiplication, matrix by matrix multiplication and contraction of two tensors. The order of reading of indices is anti-clockwise, from the left (west) position.

research, summarized in Table 1.

Even with the above notation conventions, a precise description of tensors and tensor operations is often tedious and cumbersome, given the multitude of indices involved. To this end, we grossly simplify the description of tensors and their mathematical operations through diagrammatic representations borrowed from physics and quantum chemistry [201]. In this way, tensors are represented graphically by nodes of any geometrical shapes (e.g., circles, squares, dots), while each outgoing line ("edge", "leg","arm") from a node represents the indices of a specific mode (see Figure 5) (a). Therefore, each scalar (zero-order tensor), vector (first-order tensor), matrix (2nd-order tensor) or 3rd-order

tensor is represented by a circle, while the order of a tensor is determined by the number of lines (edges) connected to it. According to this notation, an Nth-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is represented by a circle (or any shape) with N branches each of size I_n, $(n = 1, 2, \ldots, N)$ (see Section 2). An interconnection between two circles represents a contraction of tensors, which is a summation of products over a common index (see Figure 5 (b) and Section 2).

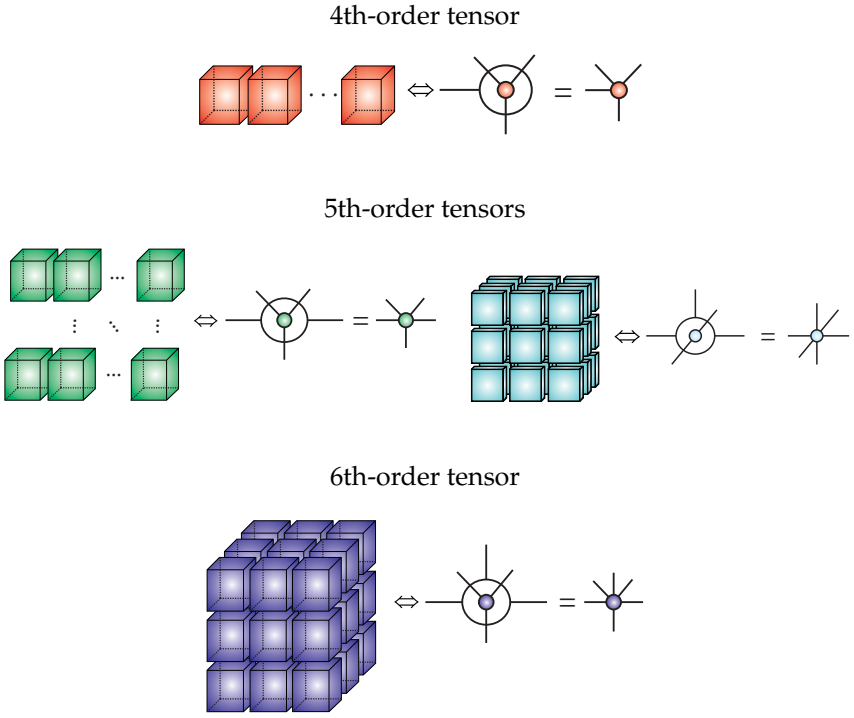


Figure 6: Graphical representations and symbols for higher-order block tensors. Each block represents either a 3rd-order tensor or a 2nd-order tensor. The outer circle indicates a global structure of the block tensor (e.g. a vector, a matrix, a 3rd-order block tensor), while the inner circle reflects the structure of each element within the block tensor.

Block tensors, where each entry (e.g., of a matrix, or a vector) is an individual sub-tensor, can be represented in a similar graphical form, as illustrated in Figure 6. Hierarchical (multilevel block) matrices are also naturally represented by tensors and vice versa, as illustrated in Figure 7, for 4th-, 5th- and 6th-order tensors. All mathematical operations on tensors can be therefore equally performed on block matrices.

In this monograph, we make extensive use of tensor network diagrams as an intuitive and visual way to efficiently represent tensor decompositions. Such graphical notations are of great help in studying and implementing so-

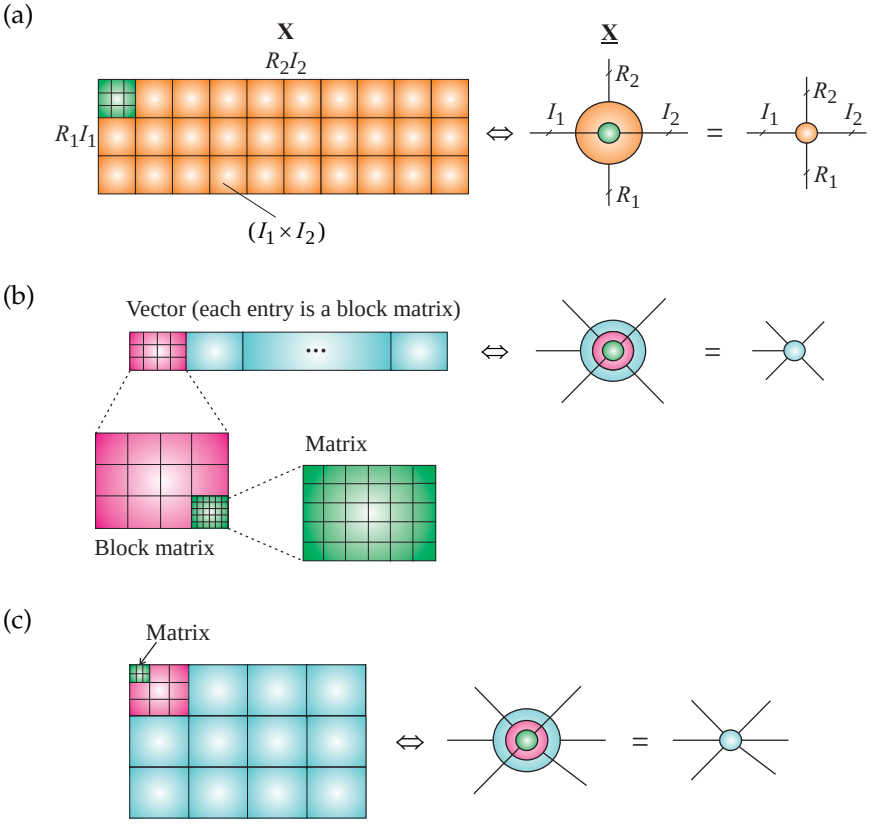


Figure 7: Hierarchical matrices and their symbolic representation as tensors. (a) a 4th-order tensor for a block matrix $\mathbf{X} \in \mathbb{R}^{R_1 I_1 \times R_2 I_2}$, which comprises block matrices $\mathbf{X}_{r_1,r_2} \in \mathbb{R}^{I_1 \times I_2}$. (b) A 5th-order tensor. (c) A 6th-order tensor.

phisticated tensor operations; we highlight the significant advantages of such diagrammatic notations for the description of tensor manipulations, and show that most tensor operations can be visualized by changes in the architecture of a tensor network diagram.

1.3 Curse of Dimensionality and Generalized Separation of Variables for Multivariate Functions

1.3.1 Curse of Dimensionality

The term *curse of dimensionality* was coined by [23] to indicate that the number of samples needed to estimate an arbitrary function with a given level of accuracy grows exponentially with the number of variables (i.e., dimensions) that

the function comprises. In a general context of machine learning and the related optimization problems, the "curse of dimensionality" may also refer to an exponentially increasing number of parameters or an extremely large number of degrees of freedom needed to describe the data/system. The term "curse of dimensionality", in the context of tensors, refers to the phenomenon whereby the number of elements, I^N, of an Nth-order tensor of size $(I \times I \times \cdots \times I)$ grows exponentially with the tensor order, N. Tensor volume can therefore easily become prohibitively big for multiway arrays for which the number of dimensions ("ways", or "modes") is very high, thus increasing the computational and memory requirements to process such data. The understanding and handling of the inherent dependencies among those excessive degrees of freedom both creates difficulties to solve problems and fascinating new opportunities, but comes at a price of reduced accuracy, through the various approximations involved.

We show that the curse of dimensionality can be alleviated or even eliminated through tensor network representations which cater for the excessive volume, veracity and variety of data (see Figure 1), and are supported by efficient tensor decomposition algorithms based on relatively simple mathematical operations (see e.g., [201]).

Another desirable aspect of tensor networks is their relatively small-scale *core tensors*, which act as "building blocks" to the tensor networks. These core tensors, which are easy to handle and visualize, enable super-compression of the raw, incomplete, and noisy huge-scale datasets. This also suggests a solution to a more general quest for new technologies for processing of exceedingly large datasets within affordable computation times [10, 131, 138, 219].

In this work, in order to address the curse of dimensionality, we will mostly focus on approximative low-rank representations of tensors, the so called low-rank tensor approximations (LRTA) or low-rank tensor network decompositions.

1.3.2 Separation of Variables and Tensor Formats

A tensor is said to be in a *full format* when it is represented as an original (raw) multidimensional array [151], however, a distributed storage and processing of high-order tensors in a full format is infeasible due to the curse of dimensionality. The *sparse format* is a variant of the full tensor format which stores only the nonzero entries of a tensor, and is used extensively in the Tensor Toolbox [14] and in the sparse grid approach [112].

As already mentioned, the problem of huge dimensionality can be alleviated by various tensor formats enabled by low-rank tensor network approximations. The underpinning idea is that instead of working with the full format, by virtue of distributed and compressed tensor network formats, both computational costs and storage requirements may be dramatically reduced through distributed storage and computing resources. It is important to note that, except for very special data structures, a tensor cannot be compressed without incurring some compression error, since a low-rank tensor representation is

only an approximation of the original tensor.

The concept of compression of multidimensional large-scale data by tensor network decompositions can be intuitively explained as follows. Consider the approximation of an N-variate function $f(\mathbf{x}) = f(x_1, x_2, \ldots, x_N)$ by a finite sum of products of individual functions, each depending on only one variable or on a few variables [22,44,84,258]. In the simplest scenario, the function $f(\mathbf{x})$ can be (approximately) represented in the following separable form

$$f(x_1, x_2, \ldots, x_N) \cong f^{(1)}(x_1) f^{(2)}(x_2) \cdots f^{(N)}(x_N). \tag{1}$$

In practice, function $f(\mathbf{x})$ is described by its discretized values, and becomes an N-order array, or a tensor. Obviously, such an approximation then corresponds to the representation by rank-1 tensors, also called elementary tensors (see Section 2). Observe that with I_n, $(n = 1, 2, \ldots, N)$, denoting the size of each mode and $I = \max_n\{I_n\}$, the memory requirement to store such a full tensor is $\prod_{n=1}^{N} I_n \leqslant I^N$, which grows exponentially with N. On the other hand, the separable representation in (1) is completely defined by its factors, $f^{(n)}(x_n)$, $(n = 1, 2, \ldots, N)$, and requires only $\sum_{n=1}^{N} I_n \ll I^N$ storage units. If $x_1, x_2, \ldots, x_N$ are statistically independent random variables, their joint probability density function is equal to the product of marginal probabilities, $f(\mathbf{x}) = f^{(1)}(x_1) f^{(2)}(x_2) \cdots f^{(N)}(x_N)$, in an exact analogy to outer products of elementary tensors. Unfortunately, this form of separability (1) is rather rare in practice.

The central idea behind tensor networks is therefore based on generalized (full or partial) separability of the variables of a high dimensional function. This can be achieved in different tensor network formats, including:

- The Canonical Polyadic (CP) format (see Section 4.2), where

$$f(x_1, x_2, \ldots, x_N) \cong \sum_{r=1}^{R} f_r^{(1)}(x_1) f_r^{(2)}(x_2) \cdots f_r^{(N)}(x_N), \tag{2}$$

  in an exact analogy to (1). In a discretized form, the above CP format can be written as an Nth-order tensor

$$\underline{\mathbf{F}} \cong \sum_{r=1}^{R} \mathbf{f}_r^{(1)} \circ \mathbf{f}_r^{(2)} \circ \cdots \circ \mathbf{f}_r^{(N)} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}, \tag{3}$$

  where $\mathbf{f}_r^{(n)} \in \mathbb{R}^{I_n}$ denotes a discretized version of the univariate function $f_r^{(n)}(x_n)$, symbol $\circ$ denotes the outer product, and R is the tensor rank.

- The Tucker format and its distributed tensor network variants, the Hierarchical Tucker (HT) decomposition and Tensor Tree Network States (see Section 4.3), given by

$$f(x_1, \ldots, x_N) \cong \sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} g_{r_1, \ldots, r_N} f_{r_1}^{(1)}(x_1) \cdots f_{r_N}^{(N)}(x_N) \tag{4}$$

- The Tensor Train (TT) format (see Section **??**), in the form

$$f(x_1, x_2, \ldots, x_N) \cong \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_{N-1}=1}^{R_{N-1}} f_{r_1}^{(1)}(x_1) f_{r_1 r_2}^{(2)}(x_2) \cdots f_{r_{N-1}}^{(N)}(x_N) \tag{5}$$

  with the equivalent compact matrix representation given by

$$f(x_1, x_2, \ldots, x_N) \cong \mathbf{F}^{(1)}(x_1)\mathbf{F}^{(2)}(x_2) \cdots \mathbf{F}^{(N)}(x_N),$$

  where $\mathbf{F}^{(n)}(x_n) \in \mathbb{R}^{R_{n-1} \times R_n}$, with $R_0 = R_N = 1$.

All the above approximations exploit the 'sum-of-products' of single-dimensional functions, a procedure which plays a key role in all tensor factorizations and decompositions.

Conceptually speaking, such a generalized separability framework is already employed in numerical methods for high-dimensional density function equations ([44, 173] and within a wide class of huge-scale optimization problems (see Section in Part 2 releted to Applications). Indeed, in many applications based on multivariate functions, very good approximations are obtained with a surprisingly small number of factors; this number corresponds to the tensor rank, R, or tensor network ranks, $R_1, R_2, \ldots, R_N$ (if the representations are exact and minimal). However, for some specific general functions this approach may fail to obtain sufficiently good low-rank TN approximations [114, 258].

To illustrate how tensor decompositions combat excessive volumes of data, if all computations are performed on a CP tensor format in (3) and not on the raw data tensor itself, then instead of the original, *exponentially growing*, data dimensionality of I^N, the number of parameters in a CP representation reduces to NIR, which *scales linearly* in N and I (see Table **??**). For example, the discretization of a 5-variate function over 100 sample points would yield the difficult to manage $100^5 = 10,000,000,000$ samples, while a rank-2 CP representation would require only $5 \times 2 \times 100 = 1000$ samples.

Although the CP format in (2) effectively bypasses the curse of dimensionality, the CP approximation may involve numerical problems for very high-order tensors, so that, in addition to the intrinsic uncloseness of the CP format, the corresponding algorithms for CP decompositions are often ill-posed [78]. As a remedy, greedy approaches may be considered which, for enhanced stability, perform consecutive rank-1 corrections [105, 175]. On the other hand, for the more flexible Tucker format in (4), many efficient and stable algorithms exist, however, it is not practical for $N > 5$ modes because the number of entries of both the original data tensor and the core tensor (expressed in (4) by elements $g_{r_1, r_2, \ldots, r_N}$) scales exponentially in the tensor order N (curse of dimensionality). In contrast to CP decomposition algorithms for TT/HT tensor network formats in (5) exhibit both very good numerical properties and the ability

to control the error of approximation, so that a desired accuracy of approximation is obtained relatively easily. [144, 204, 207]. The main advantage of the TT format over the CP decomposition is the ability to provide stable quasi-optimal rank reduction [204], achieved through, for example, truncated singular value decompositions (tSVD) or adaptive cross– approximation [21, 146, 210]. This makes the TT format one of the most stable and simple approaches to separate latent variables in a sophisticated way, while the associated TT decomposition algorithms provide full control over low-rank TN approximations[2]. In this tutorial, we therefore make extensive use of the TT format for low-rank TN approximations and employ the TT toolbox software for efficient implementations [207]. The TT format will serve as a basic prototype for high-order tensors, while considering the TTNS and HT formats (having more general tree-like structures) whenever advantageous for certain applications.

Furthermore, the concepts of tensorization of structured vectors and matrices will help to convert a wide class of huge-scale optimization problems into much smaller-scale optimization sub-problems which can be solved by existing optimization methods.

The tensor network optimization framework is therefore performed through the two main steps:

- Tensorization of data vectors and matrices, followed by a distributed approximate representation of a cost function in a specific low-rank tensor network format.
- Performing all computations and analysis in tensor network formats (i.e., using only core tensors), which offer both the reduced computational complexity and distributed memory requirements, that scale linearly, or even sub-linearly (quantized tensor networks), in the tensor order N.

1.4 Advantages of Multiway Analysis via Tensor Decompositions

Tensor decompositions (TDs) have been adopted in widely diverse disciplines, including signal and image processing, machine learning, Psychometrics, Chemometrics, Biometric, Quantum Physics/Information, Quantum Chemistry and Brain Science [51, 53, 97, 112, 153, 160, 239, 253]. This is largely due to their advantages for data that exhibit not only large volume but also very high variety (see Figure 1), as is the case in bio- and neuro-informatics and in computational neuroscience, where various forms of collecting data include sparse tabular structures and graphs or hyper-graphs.

Moreover, tensor networks have ability to efficiently parameterize, through structured compact representations, very general high-dimensional spaces which arise in modern applications [24, 50, 62, 146, 156, 170, 176, 287]. Tensor networks also naturally account for intrinsic multidimensional and distributed patterns

[2] Although similar approaches have been known in quantum physics for a long time, the rigorous mathematical analysis, however, is still work in progress (see [201, 204] and references therein).

present in data, and thus provide the opportunity to develop very sophisticated models for capturing multiple interactions and couplings in data, which are more insightful and interpretable than standard pair-wise interactions.

In this monograph, we focus on two main challenges in huge-scale data analysis which are addressed by tensor networks: (i) an approximate representation of a specific cost (objective) function by a tensor network while maintaining the desired accuracy of approximation, and (ii) the extraction of physically meaningful latent variables from data in a sufficiently accurate and computationally affordable way. To summarize, the benefits of multiway (tensor) analysis methods for large-scale datasets include:

- Ability to perform all mathematical operations in tractable tensor network formats;
- Simultaneous and flexible distributed representations of both structurally rich data and complex optimization tasks;
- Efficient compressed formats of large multidimensional data achieved via tensorization and low-rank tensor decompositions into low-order factor matrices and/or core tensors;
- Ability to operate with noisy and missing data by virtue of numerical stability and robustness to noise of low-rank tensor/matrix approximation algorithms;
- A flexible framework which naturally incorporates various diversities and constraints, thus seamlessly extending the standard Component Analysis (2-way CA) methods to multiway component analysis;
- Possibility to analyze linked (coupled) blocks of large-scale matrices and tensors in order to separate common/correlated from independent/ uncorrelated components in the observed raw data;
- Graphical representations of tensor networks allow us to simplify mathematical operations on tensors (e.g., tensor contractions and reshaping) in an intuitive way and without the explicit use of complex mathematical expressions.

This monograph therefore both reviews current research in this area and complements optimisation methods published in these Foundations, such as the Adaptive Direction Method of Multipliers (ADMM) [31].

1.5 Scope and Objectives

Review and tutorial papers [51,59,69,153,177,193,236] and books [53,112,160, 239] dealing with TDs and TNs already exist, however, they typically focus on standard models, with no explicit links to very large-scale data processing topics or connections to a wide class of optimization problems. The aim of this monograph is therefore to extend beyond the standard Tucker and CP tensor

decompositions [153], and to demonstrate the perspective of TNs in extremely large-scale data analytics, together with their role as a mathematical backbone in the discovery of hidden structures in prohibitively large-scale data. Indeed, we show that TN models provide a framework for the analysis of linked (coupled) blocks of tensors with millions and even billions of non-zero entries.

We also demonstrate that TNs provide natural extensions of 2-way (matrix) Component Analysis (2-way CA) methods, to multi-way component analysis (MWCA), for the extraction of desired components from multidimensional and multimodal data. This paradigm shift requires new models and associated algorithms capable of identifying core relations among the different tensor modes, while guaranteeing linear/sublinear scaling with the size of datasets[3].

Furthermore, we review tensor decompositions and the associated algorithms for very large-scale linear/multilinear dimensionality reduction problems (see [66, 99, 208, 269, 292, 303] and references therein). The related optimization problems often involve structured matrices and vectors with over a billion entries (see [84, 100, 108] and references therein). In particular, we focus on Symmetric Eigenvalue Decomposition (EVD/PCA) and Generalized Eigenvalue Decomposition (GEVD) [87,155,159,262], SVD [167], solutions of overdetermined and undetermined systems of linear algebraic equations [90, 206], the Moore-Penrose pseudo-inverse of structured matrices [170], and Lasso problems [168]. Tensor networks for extremely large-scale multi-block (multi-view) data are also discussed, especially TN models for orthogonal Canonical Correlation Analysis (CCA) and related Partial Least Squares (PLS) problems [289, 290]. For convenience, all these problems are reformulated as constrained optimization problems, which are then, by virtue of low-rank tensor networks reduced to manageable lower-scale optimization sub-problems. The enhanced tractability and scalability is achieved through tensor network contractions and other tensor network transformations.

The methods and approaches discussed in this work can be considered as both an alternative and as complimentary to other emerging methods for huge-scale optimization problems like random coordinate descent (RCD) scheme [195, 228], subgradient methods [196], alternating direction method of multipliers (ADMM) methods [31], and proximal gradient descent methods [213], (see also [40, 124] and references therein).

This monograph systematically introduces TN models and the associated algorithms for TNs/TDs and illustrates many potential applications of TDs/TNS. The dimensionality reduction and optimization frameworks are considered in detail, however, we also illustrate the use of TNs in other challenging problems for huge-scale datasets which can be solved using the tensor network approach, including anomaly detection, tensor completion, compressed sensing, clustering, and classification.

[3] Usually, we assume that huge-scale problems operate on at least 10^7 parameters.

2 Tensor Operations and Tensor Network Diagrams

Multilinear algebra is structurally much richer than linear algebra, and even some basic properties, such as the rank, have a more complex meaning. To this end, we next introduce the background on fundamental mathematical operations in multilinear algebra, a prerequisite for the understanding of higher-order tensor decompositions. A unified account of both the rigorous definitions and properties of tensor network operations are provided, including the outer, multi-linear, Kronecker, Khatri-Rao products. For clarity, numerous graphical illustrations are provided, together with an example rich guidance on the mathematical operations for tensors and their properties. To avoid any confusion that may arise given the numerous options for tensors reshaping, both mathematical operations and their properties are expressed directly in their native multilinear contexts.

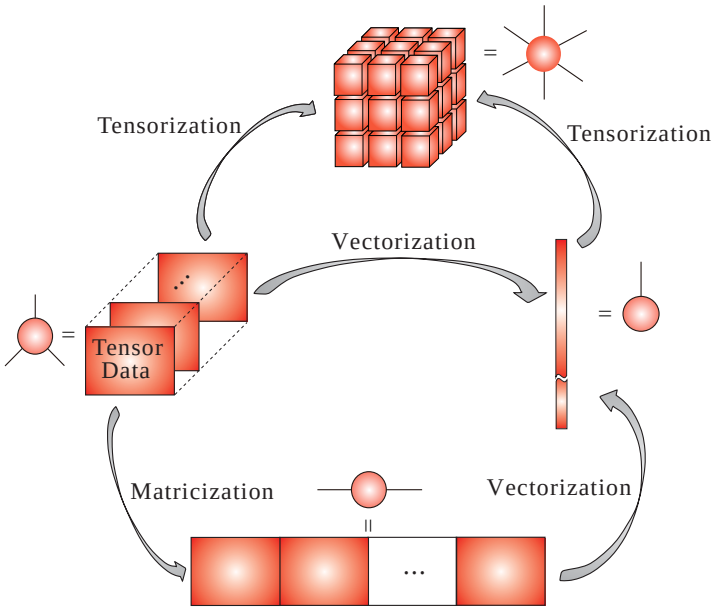


Figure 8: Tensor reshaping operations: Matricization, vectorization and tensorization. Matricization refers to converting a tensor into a matrix, vectorization to converting a tensor or a matrix into a vector, while tensorization refers to converting a vector, a matrix or a low-order tensor into a higher-order tensor. In a similar way, we can also perform reshaping of a tensor through random projections that change its entries, dimensionality or size of modes, and/or the tensor order. This is achieved by multiplying a tensor by random matrices or vectors, transformations which preserve its basic properties.

Table 3: Basic tensor/matrix operations.

$\underline{\mathbf{C}} = \underline{\mathbf{A}} \times_n \mathbf{B}$	Mode-n product (TTM) of a tensor $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and matrix $\mathbf{B} \in \mathbb{R}^{J \times I_n}$ yields a tensor $\underline{\mathbf{C}} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N}$, with entries $c_{i_1,\ldots,i_{n-1},j,i_{n+1},\ldots,i_N} = \sum_{i_n=1}^{I_n} a_{i_1,\ldots,i_n,\ldots,i_N} b_{j,i_n}$
$\underline{\mathbf{C}} = [\![\underline{\mathbf{G}}; \mathbf{B}^{(1)}, \ldots, \mathbf{B}^{(N)}]\!]$	Multilinear (Tucker) product of a core tensor, $\underline{\mathbf{G}}$, and factor matrices $\mathbf{B}^{(n)}$, which gives $\underline{\mathbf{C}} = \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \times_3 \cdots \times_N \mathbf{B}^{(N)}$
$\underline{\mathbf{C}} = \underline{\mathbf{A}} \,\bar{\times}_n\, \mathbf{b}$	Mode-n product (TVM) of a tensor $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and vector $\mathbf{b} \in \mathbb{R}^{I_n}$ yields a tensor $\underline{\mathbf{C}} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N}$, with entries $c_{i_1,\ldots,i_{n-1},i_{n+1},\ldots,i_N} = \sum_{i_n=1}^{I_n} a_{i_1,\ldots,i_{n-1},i_n,i_{n+1},\ldots,i_N} b_{i_n}$
$\underline{\mathbf{C}} = \underline{\mathbf{A}} \times_N^1 \underline{\mathbf{B}} = \underline{\mathbf{A}} \times^1 \underline{\mathbf{B}}$	Mode-$(N,1)$ contracted product of tensors $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_M}$, with $I_N = J_1$, yields a tensor $\underline{\mathbf{C}} \in \mathbb{R}^{I_1 \times \cdots \times I_{N-1} \times J_2 \times \cdots \times J_M}$ with entries $c_{i_1,\ldots,i_{N-1},j_2,\ldots,j_M} = \sum_{i_N=1}^{I_N} a_{i_1,\ldots,i_N} b_{i_N,j_2,\ldots,j_M}$
$\underline{\mathbf{C}} = \underline{\mathbf{A}} \circ \underline{\mathbf{B}}$	Outer product of tensors $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_M}$ yields an $(N+M)$th-order tensor $\underline{\mathbf{C}}$, with entries $c_{i_1,\ldots,i_N,j_1,\ldots,j_M} = a_{i_1,\ldots,i_N} b_{j_1,\ldots,j_M}$
$\underline{\mathbf{X}} = \mathbf{a} \circ \mathbf{b} \circ \mathbf{c} \in \mathbb{R}^{I \times J \times K}$	Outer product of vectors $\mathbf{a}, \mathbf{b}$ and $\mathbf{c}$ forms a rank-1 tensor, $\underline{\mathbf{X}}$, with entries $x_{ijk} = a_i\, b_j\, c_k$
$\underline{\mathbf{C}} = \underline{\mathbf{A}} \otimes_L \underline{\mathbf{B}}$	(Left) Kronecker product of tensors $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$ yields a tensor $\underline{\mathbf{C}} \in \mathbb{R}^{I_1 J_1 \times \cdots \times I_N J_N}$, with entries $c_{\overline{i_1 j_1},\ldots,\overline{i_N j_N}} = a_{i_1,\ldots,i_N} b_{j_1,\ldots,j_N}$
$\mathbf{C} = \mathbf{A} \odot_L \mathbf{B}$	(Left) Khatri-Rao product of matrices $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_J] \in \mathbb{R}^{I \times J}$ and $\mathbf{B} = [\mathbf{b}_1 \ldots, \mathbf{b}_J] \in \mathbb{R}^{K \times J}$ yields a matrix $\mathbf{C} \in \mathbb{R}^{IK \times J}$, with columns $\mathbf{c}_j = \mathbf{a}_j \otimes_L \mathbf{b}_j \in \mathbb{R}^{IK}$

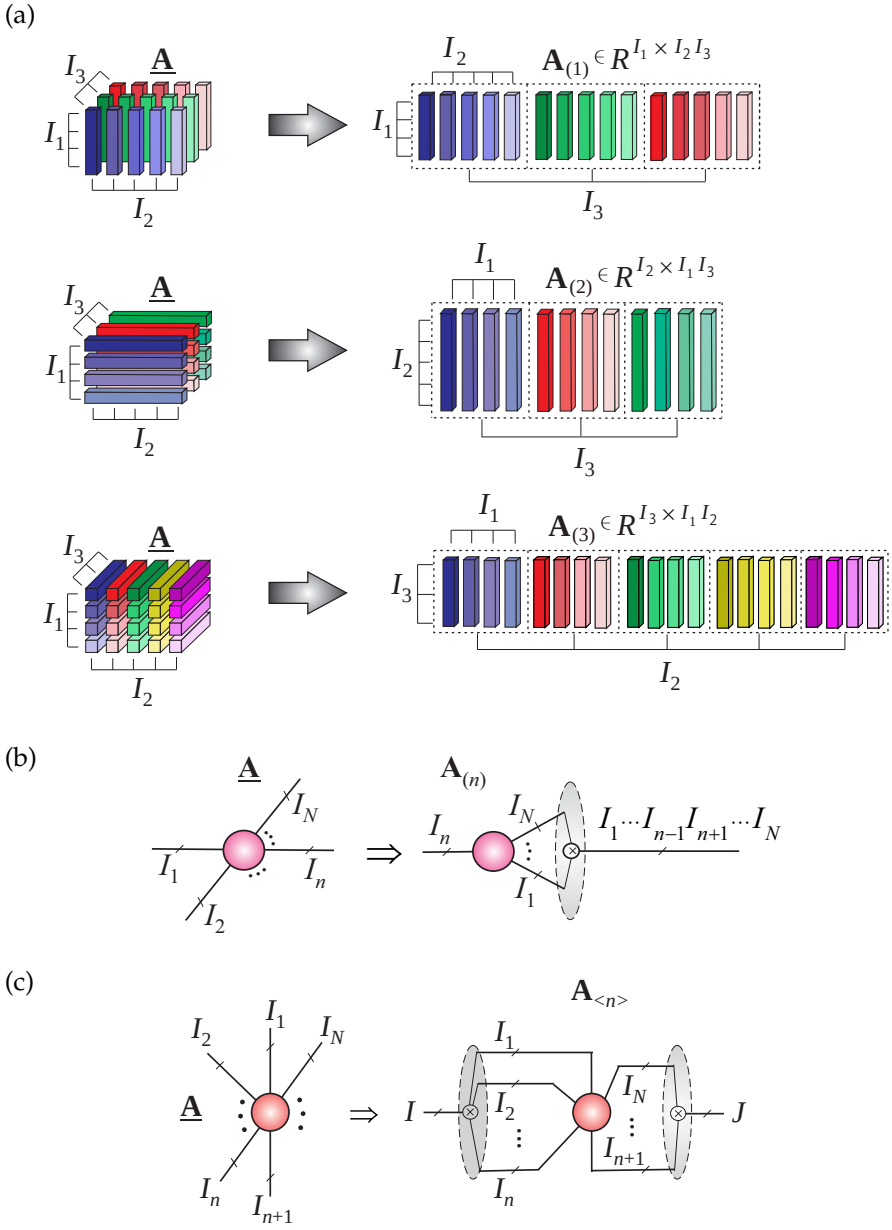


Figure 9: Matricization (flattening, unfolding) used in tensor reshaping. (a) Mode-1, mode-2, and mode-3 matricization of a 3rd-order tensor, from the top to the bottom panel. (b) Tensor diagram for the mode-n matricization, $\mathbf{A}_{(n)} \in \mathbb{R}^{I_n \times I_1 \cdots I_{n-1} I_{n+1} \cdots I_N}$, of an Nth-order tensor, $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. (c) nth canonical matricization of an Nth-order tensor, $\underline{\mathbf{A}}$, into a matrix $\mathbf{A}_{<n>} = \mathbf{A}_{(\overline{i_1 \cdots i_n}\,;\,\overline{i_{n+1} \cdots i_N})} \in \mathbb{R}^{I_1 I_2 \cdots I_n \times I_{n+1} \cdots I_N}$.

2.1 Basic Tensor Multilinear Operations

The following symbols are used for most common tensor multiplications: $\otimes$ for the Kronecker product, $\odot$ for the Khatri-Rao product, $\circledast$ for the Hadamard (componentwise) product, $\circ$ for the outer product and $\times_n$ for the mode-n product (see also Table 3). For convenience, general operations, such as $\text{vec}(\cdot)$ or $\text{diag}(\cdot)$, are defined similarly to the MATLAB syntax. Basic tensor operations are summarized in Table 3, and illustrated in Figures 8 – 19. We refer to [53,153] for more detail regarding the basic notations and tensor operations.

Multi–indices: By a multi-index $i = \overline{i_1 i_2 \cdots i_N}$ we refer to an index which takes all possible combinations of values of indices, $i_1, i_2, \ldots, i_N$, for $i_n = 1, 2, \ldots, I_n$, $n = 1, 2, \ldots, N$ and in a specific order. Multi–indices can be defined using two different conventions [90]:

1. Little-endian convention (reverse lexicographic ordering)

$$\begin{aligned} \overline{i_1 i_2 \cdots i_N} &= i_1 + (i_2 - 1) I_1 + (i_3 - 1) I_1 I_2 + \\ &\cdots \quad + (i_N - 1) I_1 \cdots I_{N-1}, \end{aligned} \tag{6}$$

2. Big-endian (colexicographic ordering)

$$\begin{aligned} \overline{i_1 i_2 \cdots i_N} &= i_N + (i_{N-1} - 1) I_N + \\ &+ \quad (i_{N-2} - 1) I_N I_{N-1} + \cdots + (i_1 - 1) I_2 \cdots I_N. \end{aligned} \tag{7}$$

The little-endian convention is used, for example, in Fortran and MATLAB, while the big-endian convention is used in C language. Given the complex and non-commutative nature of tensors, the basic definitions, such as the matricization, vectorization and the Kronecker product, should be consistent with the chosen convention[4]. In this monograph, unless otherwise stated, we will use little-endian notation.

Matricization. The matricization operator, also known as the unfolding or flattening, reorders the elements of a tensor into a matrix (see Figure 9). Such a matrix is re-indexed according to the choice of multi-index described above, and the following two fundamental matricizations are used extensively.

The mode-n matricization. For a fixed index $n \in \{1, 2, \ldots, N\}$, the mode-$n$ matricization of an Nth-order tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, is defined as the ("short" and "wide") matrix [153]

$$\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times I_1 I_2 \cdots I_{n-1} I_{n+1} \cdots I_N}, \tag{8}$$

[4] Note that using the colexicographic ordering, the vectorization of an outer product of two vectors $\mathbf{a}$ and $\mathbf{b}$, yields their Kronecker product, that is $\text{vec}(\mathbf{a} \circ \mathbf{b}) = \mathbf{a} \otimes \mathbf{b}$, while using the reverse lexicographic ordering, for the same operation, we need to use the Left Kronecker product, $\text{vec}(\mathbf{a} \circ \mathbf{b}) = \mathbf{b} \otimes \mathbf{a} = \mathbf{a} \otimes_L \mathbf{b}$.

with I_n rows and $I_1 I_2 \cdots I_{n-1} I_{n+1} \cdots I_N$ columns, and entries

$$(\mathbf{X}_{(n)})_{i_n, \overline{i_1 \cdots i_{n-1} i_{n+1} \cdots i_N}} = x_{i_1, i_2, \cdots, i_N}.$$

The nth canonical matricization. For a fixed index $n \in \{1, 2, \ldots, N\}$, the $(1, \ldots, n)$th matricization, or simply nth canonical matricization, of a tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is defined as the matrix [108]

$$\mathbf{X}_{<n>} \in \mathbb{R}^{I_1 I_2 \cdots I_n \times I_{n+1} \cdots I_N}, \tag{9}$$

with $I_1 I_2 \cdots I_n$ rows and $I_{n+1} \cdots I_N$ columns, and entries

$$(\mathbf{X}_{<n>})_{\overline{i_1 i_2 \cdots i_n}, \overline{i_{n+1} \cdots i_N}} = x_{i_1, i_2, \ldots, i_N}.$$

In the software coding practice, the matricization operator in the MATLAB notation (reverse lexicographic) is given by:

$$\mathbf{X}_{<n>} = \text{reshape}\left(\underline{\mathbf{X}}, \prod_{p=1}^{n} I_p, \prod_{p=n+1}^{N} I_p\right). \tag{10}$$

Note that the columns of a mode-n matricization, $\mathbf{X}_{(n)}$, of a tensor $\underline{\mathbf{X}}$, are the mode-n fibers of $\underline{\mathbf{X}}$. As special cases we immediately have (see Figure 9)

$$\mathbf{X}_{<1>} = \mathbf{X}_{(1)}, \quad \mathbf{X}_{<N-1>} = \mathbf{X}_{(N)}^T, \quad \mathbf{X}_{<N>} = \text{vec}(\mathbf{X}). \tag{11}$$

Consequently, the vectorization of a tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ can be expressed through its $(1, 2, \ldots, N)$th canonical matricization as

$$\text{vec}(\underline{\mathbf{X}}) \equiv \mathbf{X}_{<N>} \in \mathbb{R}^{I_1 I_2 \cdots I_N}, \tag{12}$$

for which the entries are $\text{vec}(\underline{\mathbf{X}})_{\overline{i_1 i_2 \cdots i_N}} = x_{i_1, i_2, \ldots, i_N}$.

Obviously, the tensorization of a vector or a matrix can be considered as a reverse process to the vectorization or matricization (see e.g., Figure 8).

Kronecker, Strong Kronecker Khatri-Rao products of matrices and tensors. For an $I \times J$ matrix $\mathbf{A}$ and a $K \times L$ matrix $\mathbf{B}$, the standard Right Kronecker product, $\mathbf{A} \otimes \mathbf{B}$ and the Left Kronecker product, $\mathbf{A} \otimes_L \mathbf{B}$, are the following $IK \times JL$ matrices

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{1,1}\mathbf{B} & \cdots & a_{1,J}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{I,1}\mathbf{B} & \cdots & a_{I,J}\mathbf{B} \end{bmatrix}, \qquad \mathbf{A} \otimes_L \mathbf{B} = \begin{bmatrix} \mathbf{A}b_{1,1} & \cdots & \mathbf{A}b_{1,L} \\ \vdots & \ddots & \vdots \\ \mathbf{A}b_{K,1} & \cdots & \mathbf{A}b_{K,L} \end{bmatrix}. \tag{13}$$

Observe that $\mathbf{A} \otimes_L \mathbf{B} = \mathbf{B} \otimes \mathbf{A}$, so that the Left Kronecker product will be used in most cases in this monograph as it is consistent with the little-endian notation.

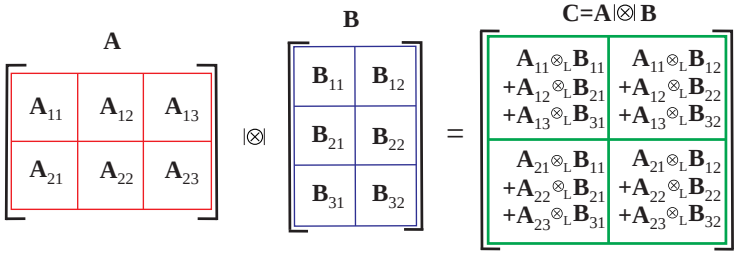


Figure 10: Illustration of the strong Kronecker product. The strong Kronecker product of two block matrices, $\mathbf{A} = [\mathbf{A}_{r_1,r_2}] \in \mathbb{R}^{R_1 I_1 \times R_2 J_1}$ and $\mathbf{B} = [\mathbf{B}_{r_2,r_3}] \in \mathbb{R}^{R_2 I_2 \times R_3 J_2}$, is defined as a block matrix $\mathbf{C} = \mathbf{A} \,|\otimes|\, \mathbf{B} \in \mathbb{R}^{R_1 I_1 I_2 \times R_3 J_1 J_2}$, with the blocks $\mathbf{C}_{r_1,r_3} = \sum_{r_2=1}^{R_2} \mathbf{A}_{r_1,r_2} \otimes_L \mathbf{B}_{r_2,r_3} \in \mathbb{R}^{I_1 I_2 \times J_1 J_2}$, for $r_1 = 1, \ldots, R_1$; $r_2 = 1, \ldots, R_2$ and $r_3 = 1, \ldots, R_3$.

Using Left Kronecker products, the strong Kronecker product of two block matrices, $\mathbf{A} \in \mathbb{R}^{R_1 I \times R_2 J}$ and $\mathbf{B} \in \mathbb{R}^{R_2 K \times R_3 L}$, given by

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \cdots & \mathbf{A}_{1,R_2} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{R_1,1} & \cdots & \mathbf{A}_{R_1,R_2} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} \mathbf{B}_{1,1} & \cdots & \mathbf{B}_{1,R_3} \\ \vdots & \ddots & \vdots \\ \mathbf{B}_{R_2,1} & \cdots & \mathbf{B}_{R_2,R_3} \end{bmatrix},$$

is defined as a block matrix (see Figure 10 for a graphical illustration)

$$\mathbf{C} = \mathbf{A} \;\; |\otimes| \;\; \mathbf{B} \in \mathbb{R}^{R_1 IK \times R_3 JL}, \tag{14}$$

with blocks $\mathbf{C}_{r_1,r_3} = \sum_{r_2=1}^{R_2} \mathbf{A}_{r_1,r_2} \otimes_L \mathbf{B}_{r_2,r_3} \in \mathbb{R}^{IK \times JL}$, where $\mathbf{A}_{r_1,r_2} \in \mathbb{R}^{I \times J}$ and $\mathbf{B}_{r_2,r_3} \in \mathbb{R}^{K \times L}$ are the blocks of matrices within $\mathbf{A}$ and $\mathbf{B}$, respectively [77, 140, 141]. Note that the strong Kronecker product is similar to the standard block-matrix multiplication, but performed using Kronecker products instead of the standard matrix-matrix products.

The above definitions of Kronecker products can be naturally extended to tensors [220] (see Table 3).

The Kronecker product of tensors. The (Left) Kronecker product of two tensors, $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$, yields a tensor $\underline{\mathbf{C}} = \underline{\mathbf{A}} \otimes_L \underline{\mathbf{B}} \in \mathbb{R}^{I_1 J_1 \times \cdots \times I_N J_N}$, with entries $c_{\overline{i_1 j_1},\ldots,\overline{i_N j_N}} = a_{i_1,\ldots,i_N}\, b_{j_1,\ldots,j_N}$ (see Figure 11).

The Mode-n of Khatri-Rao product of tensors. The Mode-n Khatri-Rao product of two tensors, $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots I_n \times \cdots \times I_N}$ and $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_n \times \cdots \times J_N}$, with $I_n = J_n$ yields a tensor $\underline{\mathbf{C}} = \underline{\mathbf{A}} \odot_n \underline{\mathbf{B}} \in \mathbb{R}^{I_1 J_1 \times \cdots \times I_{n-1} J_{n-1} \times I_n \times I_{n+1} J_{n+1} \times \cdots \times I_N J_N}$, with subtensors $\underline{\mathbf{C}}(:,\ldots:,i_n,:,\ldots,:) = \underline{\mathbf{A}}(:,\ldots:,i_n,:,\ldots,:) \otimes \underline{\mathbf{B}}(:,\ldots:,i_n,:,\ldots,:)$.

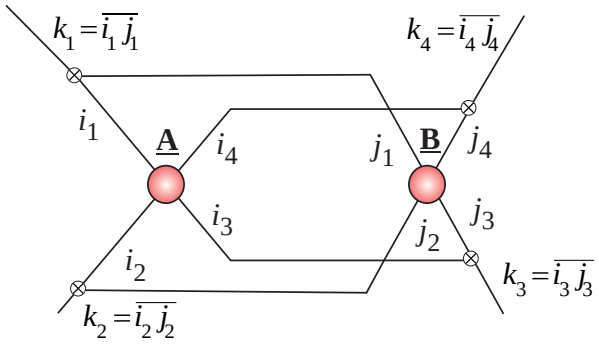


Figure 11: Kronecker product of two 4th-order tensors, $\underline{\mathbf{A}}$ and $\underline{\mathbf{B}}$, yields a tensor, $\underline{\mathbf{C}} = \underline{\mathbf{A}} \otimes_L \underline{\mathbf{B}} \in \mathbb{R}^{I_1 J_1 \times \cdots \times I_4 J_4}$, with entries $c_{k_1,k_2,\ldots,k_4} = a_{i_1,\ldots,i_4}\, b_{j_1,\ldots,j_4}$, where $k_n = \overline{i_n j_n}$, $(n = 1, 2, 3, 4)$.

The Mode-2 and mode-1 of Khatri-Rao product of matrices. The above definition simplifies to the standard Khatri-Rao (mode-2) product of two block matrices, $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_R] \in \mathbb{R}^{I \times R}$ and $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_R] \in \mathbb{R}^{J \times R}$, is defined as the "column-wise Kronecker product". Therefore, the standard Right and Left Khatri-Rao products are respectively given by[5]

$$\mathbf{A} \odot \mathbf{B} = [\mathbf{a}_1 \otimes \mathbf{b}_1, \mathbf{a}_2 \otimes \mathbf{b}_2, \ldots, \mathbf{a}_R \otimes \mathbf{b}_R] \in \mathbb{R}^{IJ \times R}, \tag{15}$$

$$\mathbf{A} \odot_L \mathbf{B} = [\mathbf{a}_1 \otimes_L \mathbf{b}_1, \mathbf{a}_2 \otimes_L \mathbf{b}_2, \ldots, \mathbf{a}_R \otimes_L \mathbf{b}_R] \in \mathbb{R}^{IJ \times R}. \tag{16}$$

Analogously, we define mode-1 Khatri-Rao product of two matrices $\mathbf{A} \in \mathbb{R}^{I \times R}$ and $\mathbf{B} \in \mathbb{R}^{I \times Q}$, as

$$\mathbf{A} \odot_1 \mathbf{B} = \begin{bmatrix} \mathbf{A}(1,:) \otimes \mathbf{B}(1,:) \\ \vdots \\ \mathbf{A}(I,:) \otimes \mathbf{B}(I,:) \end{bmatrix} \in \mathbb{R}^{I \times RQ}. \tag{17}$$

Direct sum of tensors. Direct sum of tensors $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times \cdots \times J_N}$ yields a tensor $\underline{\mathbf{C}} = \underline{\mathbf{A}} \oplus \underline{\mathbf{B}} \in \mathbb{R}^{(I_1+J_1) \times \cdots \times (I_N+J_N)}$, with entries $\underline{\mathbf{C}}(k_1, \ldots, k_N) = \underline{\mathbf{A}}(k_1, \ldots, k_N)$ if $1 \leqslant k_n \leqslant I_n \ \forall n$, $\underline{\mathbf{C}}(k_1, \ldots, k_N) = \underline{\mathbf{B}}(k_1 - I_1, \ldots, k_N - I_N)$ if $I_n < k_n \leqslant I_n + J_n \ \forall n$, and $\underline{\mathbf{C}}(k_1, \ldots, k_N) = 0$, otherwise (see Fig. 12).

Partial (mode-n) direct sum of tensors. Partial direct sum of tensors $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times \cdots \times J_N}$, with $I_n = J_n$, yields a tensor $\underline{\mathbf{C}} = \underline{\mathbf{A}} \oplus_{\bar{n}} \underline{\mathbf{B}} \in \mathbb{R}^{(I_1+J_1) \times \cdots \times (I_{n-1}+J_{n-1}) \times I_n \times (I_{n+1}+J_{n+1}) \times \cdots \times (I_N+J_N)}$, with $\underline{\mathbf{C}}(:, \ldots, :, i_n, :, \ldots, :) = \underline{\mathbf{A}}(:, \ldots, :, i_n, :, \ldots, :) \oplus \underline{\mathbf{B}}(:, \ldots, :, i_n, :, \ldots, :)$.

[5]For simplicity mode 2 subindex has been neglected,i.e. $\mathbf{A} \odot_2 \mathbf{B} = \mathbf{A} \odot \mathbf{B}$.

Concatenation of Nth-order tensors. Concatenation along mode-n of tensors $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times \cdots \times J_N}$, with $I_m = J_m, \ \forall m \neq n$ yields a tensor $\underline{\mathbf{C}} = \underline{\mathbf{A}} \boxplus_n \underline{\mathbf{B}} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times (I_n + J_n) \times I_{n+1} \times \cdots \times (I_N + J_N)}$, with subtensors $\underline{\mathbf{C}}(i_1, \ldots, i_{n-1}, :, i_{n+1}, \ldots, i_N) = \underline{\mathbf{A}}(i_1, \ldots, i_{n-1}, :, i_{n+1}, \ldots, i_N) \oplus \underline{\mathbf{B}}(i_1, \ldots, i_{n-1}, :, i_{n+1}, \ldots, i_N)$. For concatenation of two tensors of suitable dimensions along mode-n, we will use equivalent notations $\underline{\mathbf{C}} = \underline{\mathbf{A}} \boxplus_n \underline{\mathbf{B}} = \underline{\mathbf{A}} \frown_n \underline{\mathbf{B}}$.

3D Convolution. For simplicity lets consider two 3rd-order tensors: $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$. 3D Convolution of two 3rd-order tensors yields a tensor $\underline{\mathbf{C}} = \underline{\mathbf{A}} * \underline{\mathbf{B}} \in \mathbb{R}^{(I_1+J_1-1) \times (I_2+J_2-1) \times (I_3+J_3-1)}$, with entries $\underline{\mathbf{C}}(k_1, k_2, k_3) = \sum_{j_1} \sum_{j_2} \sum_{j_3} \underline{\mathbf{B}}(j_1, j_2, j_3) \, \underline{\mathbf{A}}(k_1 - j_1, k_2 - j_2, k_3 - j_3)$ (see Fig. 13 and Fig. 14).

Partial (Mode-n) Convolution. For simplicity lets consider two 3rd-order tensors: $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$. Mode-2 (partial) convolution yields a tensor $\underline{\mathbf{C}} = \underline{\mathbf{A}} \boxdot_2 \underline{\mathbf{B}} \in \mathbb{R}^{I_1 J_1 \times (I_2+J_2-1) \times I_3 J_3}$, with subtensors (vectors) $\underline{\mathbf{C}}(k_1, :, k_3) = \underline{\mathbf{A}}(i_1, :, \overline{i_3}) * \underline{\mathbf{B}}(j_1, :, j_3) \in \mathbb{R}^{I_2+J_2-1}$, where $k_1 = \overline{i_1 j_1}$, $k_2 = 1, 2, \ldots, (I_2 + J_2 - 1)$, $k_3 = \overline{i_3 j_3}$.

Outer product (tensor product). The central operator in tensor analysis is the outer or tensor product, which for the tensors $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times \cdots \times J_M}$, gives the tensor $\underline{\mathbf{C}} = \underline{\mathbf{A}} \circ \underline{\mathbf{B}} \in \mathbb{R}^{I_1 \times \cdots \times I_N \times J_1 \times \cdots \times J_M}$ with entries $c_{i_1, \ldots, i_N, j_1, \ldots, j_M} = a_{i_1, \ldots, i_N} \, b_{j_1, \ldots, j_M}$.

Note that for 1st-order tensors (vectors), the tensor product reduces to the standard outer product of two nonzero vectors, $\mathbf{a} \in \mathbb{R}^I$ and $\mathbf{b} \in \mathbb{R}^J$, which yields a rank-1 matrix, $\mathbf{X} = \mathbf{a} \circ \mathbf{b} = \mathbf{a}\mathbf{b}^T \in \mathbb{R}^{I \times J}$. The outer product of three nonzero vectors, $\mathbf{a} \in \mathbb{R}^I$, $\mathbf{b} \in \mathbb{R}^J$ and $\mathbf{c} \in \mathbb{R}^K$, gives a 3rd-order rank-1 tensor (called pure or elementary tensor), $\underline{\mathbf{X}} = \mathbf{a} \circ \mathbf{b} \circ \mathbf{c} \in \mathbb{R}^{I \times J \times K}$, with entries $x_{ijk} = a_i \, b_j \, c_k$.

Rank-1 tensor. A tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, is said to be of rank-1 if it can be expressed exactly as the outer product, $\underline{\mathbf{X}} = \mathbf{b}^{(1)} \circ \mathbf{b}^{(2)} \circ \cdots \circ \mathbf{b}^{(N)}$, where $\mathbf{b}^{(n)} \in \mathbb{R}^{I_n}$ are nonzero vectors and the tensor entries are given by $x_{i_1, i_2, \ldots, i_N} = b^{(1)}_{i_1} b^{(2)}_{i_2} \cdots b^{(N)}_{i_N}$.

Tensor rank, Kruskal tensor, CP decomposition. For further discussion, it is important to highlight that any tensor can be expressed as a sum of rank-1 tensors, in the form

$$\underline{\mathbf{X}} = \sum_{r=1}^{R} \mathbf{b}_r^{(1)} \circ \mathbf{b}_r^{(2)} \circ \cdots \circ \mathbf{b}_r^{(N)} = \sum_{r=1}^{R} \circ_{n=1}^{N} \mathbf{b}_r^{(n)}, \qquad \mathbf{b}_r^{(n)} \in \mathbb{R}^{I_n} \tag{18}$$

which is exactly the form of the Kruskal tensor, also known under the names of CANDECOMP/PARAFAC, Canonical Polyadic decomposition, or simply the CP decomposition in (2), for which we will use the acronyms CP and CPD (see Figure 15).

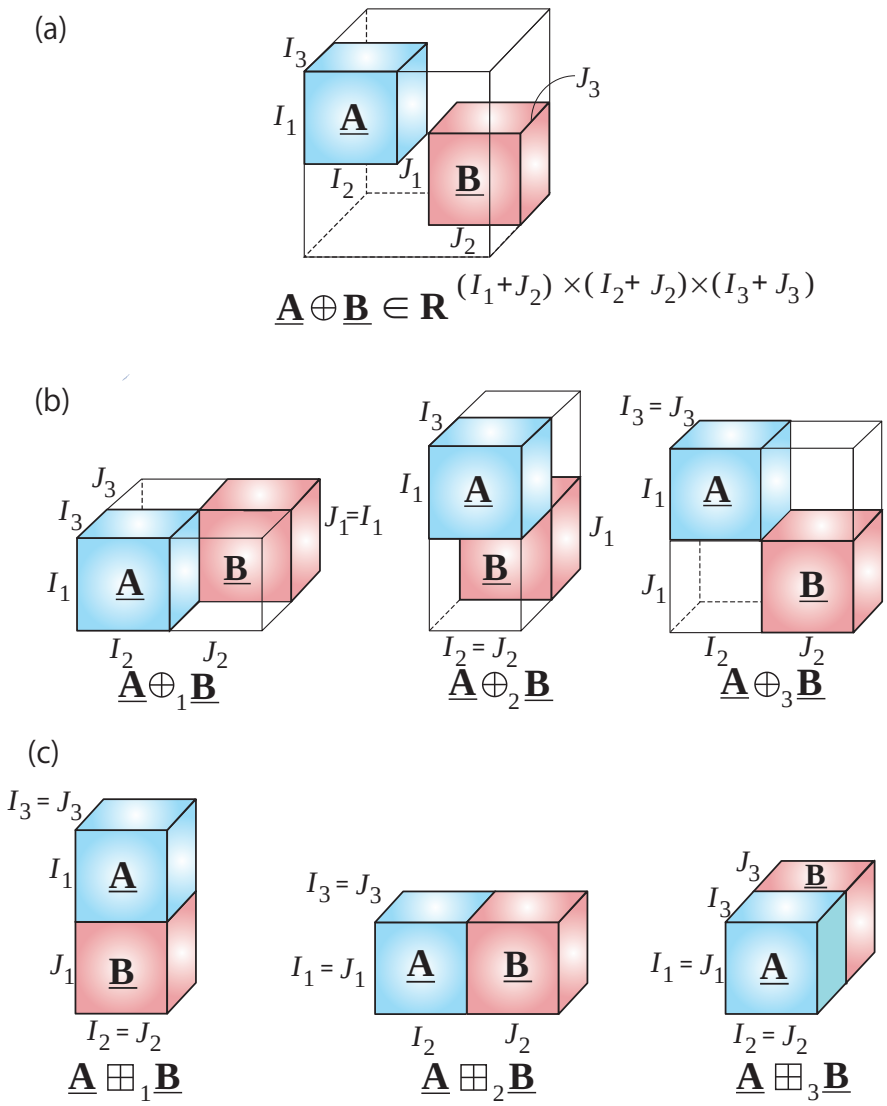


Figure 12: Illustration of direct sum, partial direct sum and concatenation of two 3rd-order tensors: (a) Direct sum, (b) partial (mode-1,2,3) direct sum, (c) concatenations along mode-1,2,3.

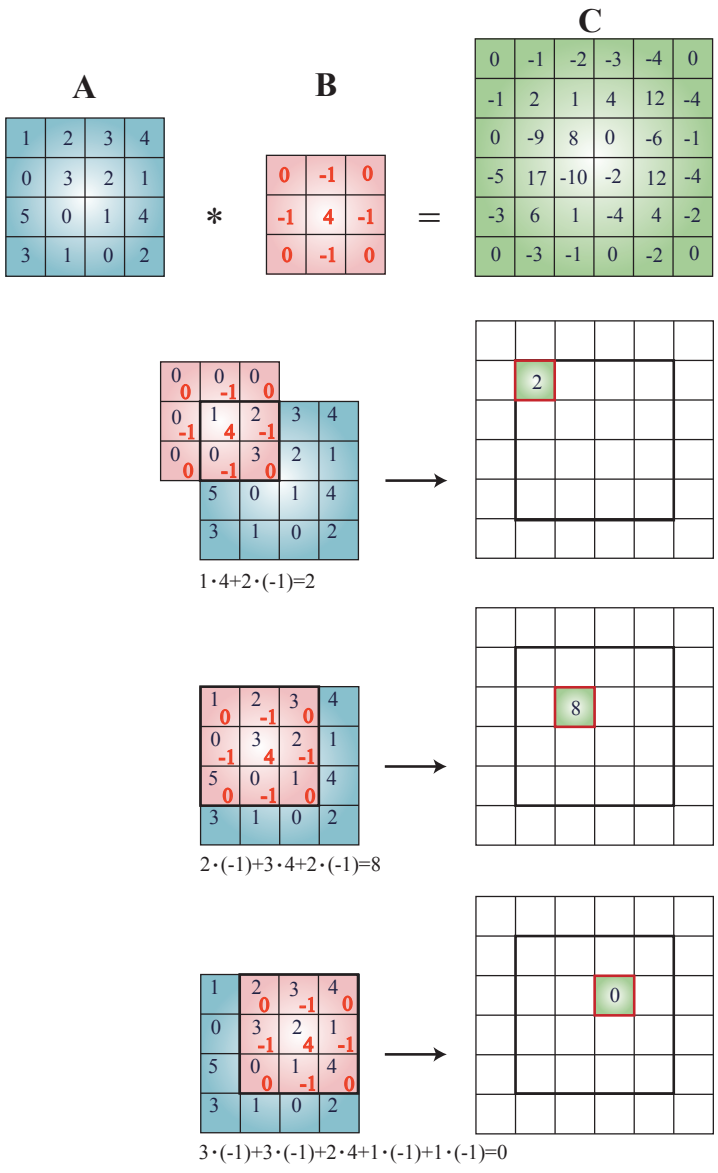


Figure 13: Illustration of 2D convolution.

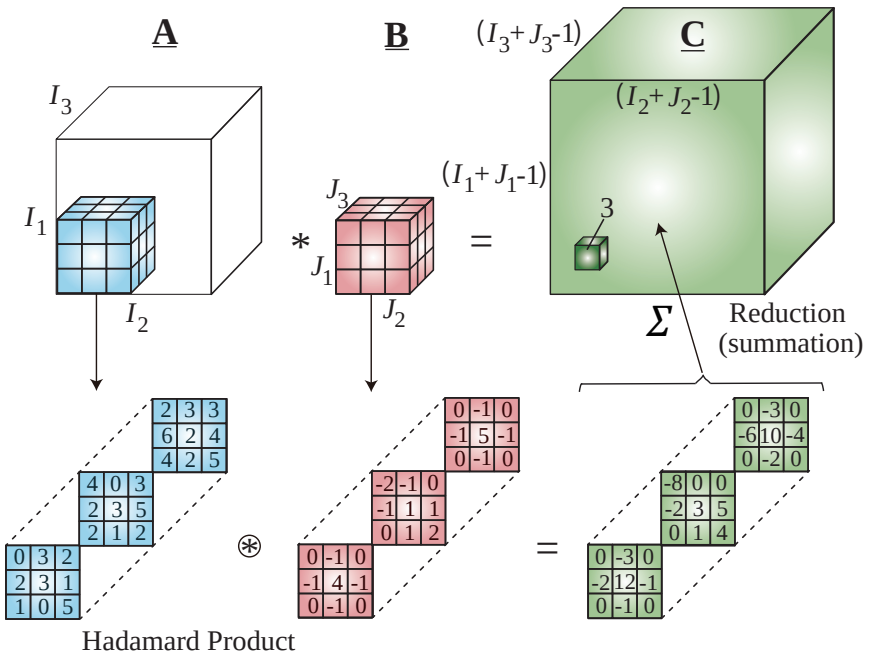


Figure 14: Illustration of 3D convolution.

The tensor rank, also called the CP rank, is a natural extension of the matrix rank and is defined as minimum number R of rank-1 terms in an exact CP decomposition of the form in (18). Although the CP decomposition has already found many practical applications, one limiting theoretical property of the CPD is that the best rank-R approximation of a given data tensor may not exist (see [78] for more detail). However, a rank-R tensor may be approximated arbitrarily close by a sequence of tensors for which the CP ranks are strictly less than R. For these reasons, the concept of border rank was proposed [27], and is defined as the minimum number of rank-1 tensors which provides the given tensor approximation with an arbitrary accuracy.

Symmetric tensor decomposition. A symmetric tensor (sometimes called a super-symmetric tensor) is invariant to the permutations of its indices. A symmetric tensor of Nth-order has equal sizes, $I_n = I, \forall n$, in all modes, and the same entry for every permutation of its indices. Notice that for a vector $\mathbf{b}^{(n)} = \mathbf{b} \in \mathbb{R}^I$, $\forall n$, the rank-1 tensor (constructed by N outer products), $\circ_{n=1}^{N} \mathbf{b}^{(n)} = \mathbf{b} \circ \mathbf{b} \circ \cdots \circ \mathbf{b} \in \mathbb{R}^{I \times I \times \cdots \times I}$, is symmetric. Moreover, every symmetric tensor may be expressed as a linear combination of such symmetric rank-1 tensors

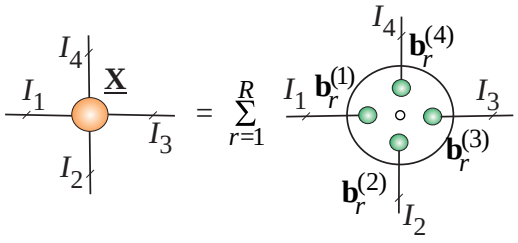


Figure 15: The CP decomposition for a 4th-order tensor $\underline{\mathbf{X}}$ of rank R. Observe that the rank-1 tensors $\underline{\mathbf{X}}$ are formed through the outer products of the vectors $\mathbf{b}_r^{(1)}, \cdots, \mathbf{b}_r^{(4)}$, $r = 1, \cdots, R$.

within the so-called symmetric CP decomposition, given by

$$\underline{\mathbf{X}} = \sum_{r=1}^{R} \lambda_r \mathbf{b}_r \circ \mathbf{b}_r \circ \cdots \circ \mathbf{b}_r, \qquad \mathbf{b}_r \in \mathbb{R}^I, \tag{19}$$

where $\lambda_r \in \mathbb{R}$ are the scaling parameters for the unit length vectors $\mathbf{b}_r$, while the symmetric tensor rank is the minimal number R of rank-1 tensors that is necessary to reconstruct it [57].

Multilinear products. The mode-n (multilinear) product, also called the tensor-times-matrix product (TTM), of a tensor, $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, and a matrix, $\mathbf{B} \in \mathbb{R}^{J \times I_n}$, gives the tensor [153]

$$\underline{\mathbf{C}} = \underline{\mathbf{A}} \times_n \mathbf{B} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N}, \tag{20}$$

with entries

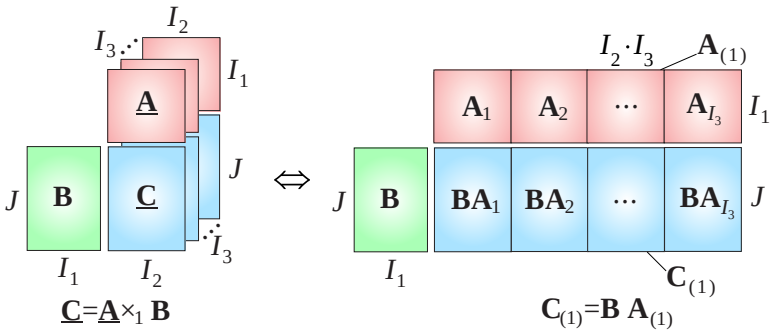
$$c_{i_1, i_2, \ldots, i_{n-1}, j, i_{n+1}, \ldots, i_N} = \sum_{i_n=1}^{I_n} a_{i_1, i_2, \ldots, i_N} \, b_{j, i_n}. \tag{21}$$

From (21) and Figure 16, the equivalent matrix form is $\mathbf{C}_{(n)} = \mathbf{B}\mathbf{A}_{(n)}$, which allows us to employ established fast matrix-by-vector and matrix-by-matrix multiplications when dealing with very large-scale tensors. Efficient and optimized algorithms for TTM are, however, still emerging [17, 18, 171].

Multilinear product of a tensor and a vector (TTV). In a similar way, the mode-n multiplication of a tensor, $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, and a vector, $\mathbf{b} \in \mathbb{R}^{I_n}$, (tensor-times-vector, TTV, multiply) yields a tensor

$$\underline{\mathbf{C}} = \underline{\mathbf{A}} \bar{\times}_n \mathbf{b} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N}, \tag{22}$$

(a)



(b)

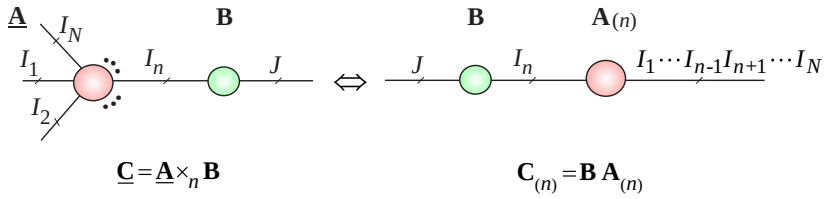


Figure 16: Illustration of the multilinear mode-n product, also known as the TTM (Tensor-Times-Matrix) product, performed in the tensor format (left) and the matrix format (right). (a) Mode-1 product of a 3rd-order tensor, $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, and a factor (component) matrix, $\mathbf{B} \in \mathbb{R}^{J \times I_1}$, yields a tensor $\underline{\mathbf{C}} = \underline{\mathbf{A}} \times_1 \mathbf{B} \in \mathbb{R}^{J \times I_2 \times I_3}$. This is equivalent to a simple matrix multiplication formula, $\mathbf{C}_{(1)} = \mathbf{B}\mathbf{A}_{(1)}$. (b) Mode-$n$ product of an Nth-order tensor, $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, and a factor matrix, $\mathbf{B} \in \mathbb{R}^{J \times I_n}$.

with entries

$$c_{i_1,\ldots,i_{n-1},i_{n+1},\ldots,i_N} = \sum_{i_n=1}^{I_n} a_{i_1,\ldots,i_{n-1},i_n,i_{n+1},\ldots,i_N}\, b_{i_n}. \tag{23}$$

Note that the mode-n multiplication of a tensor by a matrix does not change the tensor order, while the multiplication of a tensor by vectors reduces its order (see Figure 17).

Full multilinear (Tucker) product. A full multilinear product, also called the Tucker product, of an Nth-order tensor, $\underline{\mathbf{G}} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$, and a set of N factor matrices, $\underline{\mathbf{B}}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ for $n = 1, 2, \ldots, N$, performs the multiplications

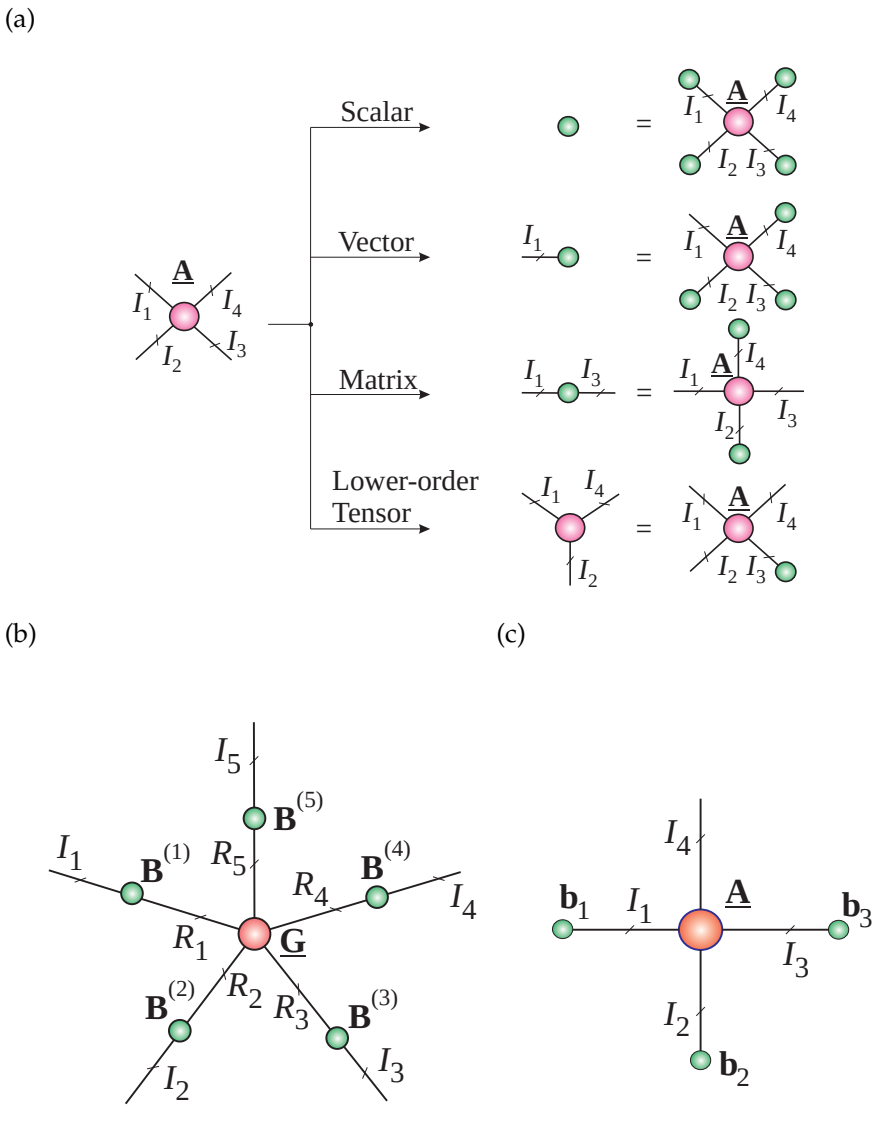


Figure 17: Multilinear tensor products in a compact tensor network notation. (a) Transforming and/or compressing a 4th-order tensor, $\underline{\mathbf{A}}$, into a scalar, vector, matrix and 3rd-order tensor, by multilinear products of the tensor and vectors. Note that a mode-n multiplication of a tensor by a matrix does not change the order of a tensor, while a multiplication of a tensor by a vector reduces its order. For example, a multilinear product of a tensor and four vectors (top diagram) yields a scalar. (b) Multilinear product of a tensor, $\underline{\mathbf{G}} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_5}$, and five factor (component) matrices, $\mathbf{B}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ $(n = 1, 2, \ldots, 5)$, yields the tensor $\underline{\mathbf{C}} = \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \times_3 \mathbf{B}^{(3)} \times_4 \mathbf{B}^{(4)} \times_5 \mathbf{B}^{(5)} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_5}$. This corresponds to the Tucker format. (c) Multilinear product of a tensor, $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4}$, and vectors, $\mathbf{b}_n \in \mathbb{R}^{I_n}$ $(n = 1, 2, 3)$, yields the vector $\mathbf{c} = \underline{\mathbf{A}} \bar{\times}_1 \mathbf{b}_1 \bar{\times}_2 \mathbf{b}_2 \bar{\times}_3 \mathbf{b}_3 \in \mathbb{R}^{I_4}$.

in all the modes which can be compactly written as [153] (see Figure 17 (b))):

$$\begin{aligned} \underline{\mathbf{C}} &= \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)} \\ &= [\![\underline{\mathbf{G}}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N)}]\!] \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}. \end{aligned} \tag{24}$$

Observe that this format corresponds to the Tucker decomposition [259, 260] (see Section 4.3).

Multilinear products of a tensor by matrices or vectors play a key role in deterministic methods for the reshaping of tensors and dimensionality reduction, as well as in statitistical/probabilistic methods for randomization/sketching procedures and in random projections of tensors into matrices or vectors [91, 161, 172, 177, 241, 248, 278].

Tensor contractions. Tensor contraction is a fundamental and the most important operation in tensor networks, and can be considered a higher-dimensional analogue of matrix multiplications, inner product, and outer product.

In a way similar to the mode-n multilinear product[6], the mode-$\binom{m}{n}$ product (tensor contraction) of two tensors, $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ and $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_M}$, with common modes, $I_n = J_m$, yields an $(N + M - 2)$-order tensor, $\underline{\mathbf{C}} \in \mathbb{R}^{I_1 \times \cdots I_{n-1} \times I_{n+1} \times \cdots \times I_N \times J_1 \times \cdots J_{m-1} \times J_{m+1} \times \cdots \times J_M}$, in the form (see Figure 18 (a)):

$$\underline{\mathbf{C}} = \underline{\mathbf{A}} \times_n^m \underline{\mathbf{B}}, \tag{25}$$

for which the entries are computed as

$$\begin{aligned} & c_{i_1, \ldots, i_{n-1}, i_{n+1}, \ldots, i_N, j_1, \ldots, j_{m-1}, j_{m+1}, \ldots, j_M} = \\ &= \sum_{i_n=1}^{I_n} a_{i_1, \ldots, i_{n-1}, i_n, i_{n+1}, \ldots, i_N} \; b_{j_1, \ldots, j_{m-1}, i_n, j_{m+1}, \ldots, j_M}. \end{aligned} \tag{26}$$

This operation is referred to as a *contraction of two tensors in single common mode*.

Tensors can be contracted in several modes or even in all modes, as illustrated in Figure 18. For convenience of presentation, the super- or sub-index, e.g. m, n, will be omitted. In this way, for example, the multilinear product of the tensors, $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_M}$, with common modes, $I_N = J_1$, can be written as

$$\underline{\mathbf{C}} = \underline{\mathbf{A}} \; \times_N^1 \; \underline{\mathbf{B}} = \underline{\mathbf{A}} \times^1 \underline{\mathbf{B}} = \underline{\mathbf{A}} \bullet \underline{\mathbf{B}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N-1} \times J_2 \times \cdots \times J_M}, \tag{27}$$

for which the entries $c_{i_1, i_2, \ldots, i_{N-1}, j_2, j_3, \ldots, j_M} = \sum_{i_N=1}^{I_N} a_{i_1, i_2, \ldots, i_N} \; b_{i_N, j_2, \ldots, j_M}$.

In this notation, the multiplications of matrices and vectors, can be written as, $\mathbf{A} \times_2^1 \mathbf{B} = \mathbf{A} \times^1 \mathbf{B} = \mathbf{A}\mathbf{B}$, $\mathbf{A} \times_2^2 \mathbf{B} = \mathbf{A}\mathbf{B}^T$, $\mathbf{A} \times_{1,2}^{1,2} \mathbf{B} = \mathbf{A} \bar{\times} \mathbf{B} = \langle \mathbf{A}, \mathbf{B} \rangle$, and $\mathbf{A} \times_2^1 \mathbf{x} = \mathbf{A} \times^1 \mathbf{x} = \mathbf{A}\mathbf{x}$.

When contracting more than two tensors, the order has to be precisely specified (defined), for example, $\underline{\mathbf{A}} \times_a^b \underline{\mathbf{B}} \times_c^d \underline{\mathbf{C}} = \underline{\mathbf{A}} \times_a^b (\underline{\mathbf{B}} \times_c^d \underline{\mathbf{C}})$ for $b < c$.

[6] In the literature, sometimes the symbol $\times_n$ is replaced by $\bullet_n$.

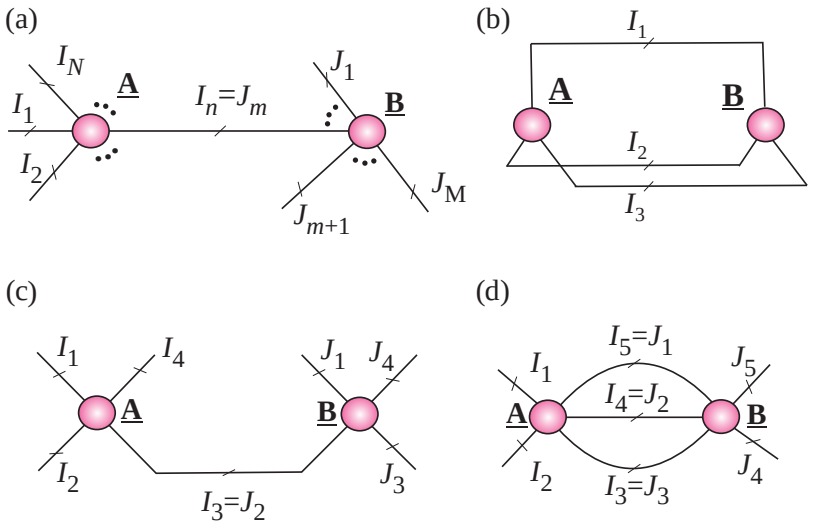


Figure 18: Examples of contractions of two tensors. (a) Multilinear product of two tensors is denoted by $\underline{\mathbf{A}} \times_n^m \underline{\mathbf{B}}$. (b) Inner product of two 3rd-order tensors yields a scalar $c = \langle \underline{\mathbf{A}}, \underline{\mathbf{B}} \rangle = \underline{\mathbf{A}} \times_{1,2,3}^{1,2,3} \underline{\mathbf{B}} = \underline{\mathbf{A}} \,\bar{\times}\, \underline{\mathbf{B}} = \sum_{i_1,i_2,i_3} a_{i_1,i_2,i_3}\, b_{i_1,i_2,i_3}$. (c) Tensor contraction of two 4th-order tensors yields a 6th-order tensor, $\underline{\mathbf{C}} = \underline{\mathbf{A}} \times_3^2 \underline{\mathbf{B}} \in \mathbb{R}^{I_1 \times I_2 \times I_4 \times J_1 \times J_3 \times J_4}$, with entries $c_{i_1,i_2,i_4,j_1,j_3,j_4} = \sum_{i_3} a_{i_1,i_2,i_3,i_4}\, b_{j_1,i_3,j_3,j_4}$. (d) Tensor contraction of two 5th-order tensors yields a 4th-order tensor, $\underline{\mathbf{C}} = \underline{\mathbf{A}} \times_{5,4,3}^{1,2,3} \underline{\mathbf{B}} \in \mathbb{R}^{I_1 \times I_2 \times J_4 \times J_5}$, with entries $c_{i_1,i_2,j_4,j_5} = \sum_{i_3,i_4,i_5} a_{i_1,i_2,i_3,i_4,i_5}\, b_{i_5,i_4,i_3,j_4,j_5}$.

It is important to note that a matrix-by-vector product, $\mathbf{y} = \mathbf{A}\mathbf{x} \in \mathbb{R}^{I_1 \cdots I_N}$, with $\mathbf{A} \in \mathbb{R}^{I_1 \cdots I_N \times J_1 \cdots J_N}$ and $\mathbf{x} \in \mathbb{R}^{J_1 \cdots J_N}$, can be expressed in a tensorized form via the contraction operator as $\underline{\mathbf{Y}} = \underline{\mathbf{A}} \,\bar{\times}\, \underline{\mathbf{X}}$, where the symbol $\bar{\times}$ denotes the contraction of all modes of the tensor $\underline{\mathbf{X}}$. Similarly, a scalar, $c = \mathbf{y}^T \mathbf{A}\mathbf{x}$, where all modes are contracted, can be expressed in a tensorized form as $c = (\underline{\mathbf{A}} \,\bar{\times}\, \underline{\mathbf{X}} | \,\bar{\times}\, \underline{\mathbf{Y}})$ (see Section in Part 2).

Unlike the matrix-by-matrix multiplications for which several efficient parallel schemes have been developed, the number of efficient algorithms for tensor contractions is rather limited [82, 135, 216]. In practice, due to the high computational complexity of tensors contractions, especially for tensor networks with loops, this operation is often performed not exactly but approximately [178].

Tensor trace. The trace of a tensor, $\underline{\mathbf{A}} \in \mathbb{R}^{R \times I_1 \times I_2 \times \cdots \times I_N \times R}$, is defined as a reduced-order tensor, $\widetilde{\underline{\mathbf{A}}} = \mathrm{Tr}(\underline{\mathbf{A}}) \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, with entries [169]

$$\widetilde{\underline{\mathbf{A}}}(i_1, i_2, \ldots, i_N) = \sum_{r=1}^{R} \underline{\mathbf{A}}(r, i_1, i_2, \ldots, i_N, r), \tag{28}$$

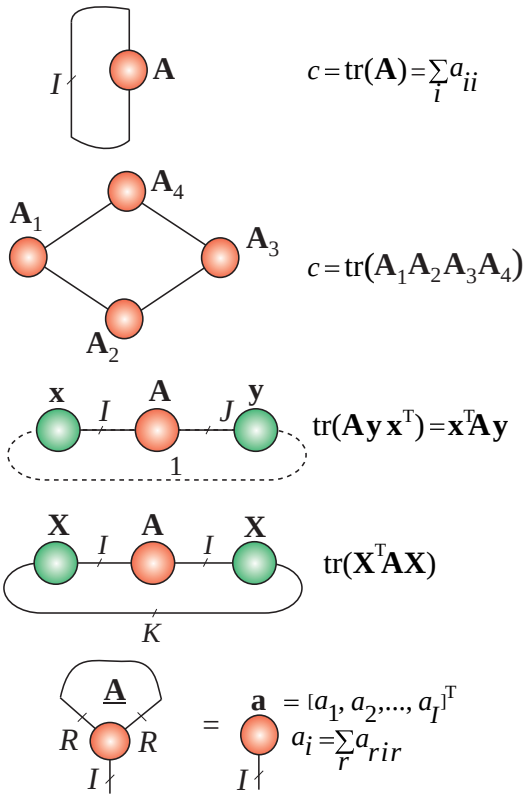


Figure 19: Tensor network notation for the traces of matrices (panels 1-4 from the top), and a (partial) trace of a 3rd-order tensor (bottom panel).

which can be considered as a generalization of the standard matrix trace or as *a self-contraction of a tensor*. For example, the trace of a 3rd-order tensor, $\underline{\mathbf{A}} \in \mathbb{R}^{R \times I \times R}$, is a vector, $\mathbf{a} = \mathrm{Tr}(\underline{\mathbf{A}}) \in \mathbb{R}^{I}$, the elements of which are the traces of its slice matrices, that is,

$$\mathbf{a} = \mathrm{Tr}(\underline{\mathbf{A}}) = [\mathrm{tr}(\mathbf{A}_1), \ldots, \mathrm{tr}(\mathbf{A}_i), \ldots, \mathrm{tr}(\mathbf{A}_I)]^T, \tag{29}$$

where $\mathbf{A}_i \in \mathbb{R}^{R \times R}$ $(i = 1, 2, \ldots, I)$ are the lateral slices of a tensor $\underline{\mathbf{A}}$ (see bottom of Figure 19).

Conversions of tensors to traces, scalars, vectors, matrices or tensors with reshaped modes and/or reduced orders are illustrated in Figures 17– 19.

3 Graphical Representation of Fundamental Tensor Networks (TNs)

The role of tensor networks is to represent a higher-order tensor as a set of sparsely interconnected lower-order tensors (see Figure 20), and in this way provide computational and storage benefits. The lines (branches, edges) connecting core tensors correspond to the contracted modes and their number also represents the rank of a tensor network[7], whereas the lines which do not connect core tensors correspond to the physical variables (modes, indices) in the data tensor. In other words, the number of free (dangling) edges determines the order of a data tensor under consideration.

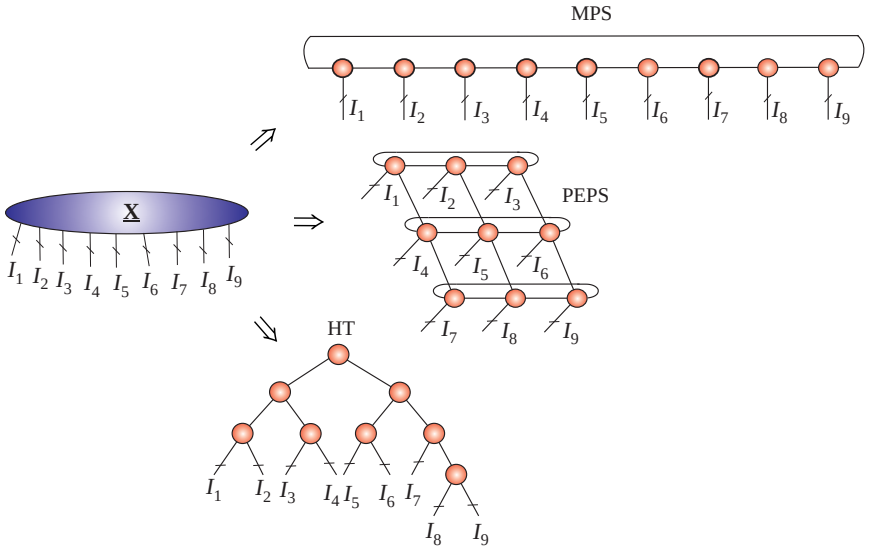


Figure 20: Illustration of the decomposition of a 9th-order tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_9}$, into different forms of tensor networks (TNs). In general, the objective is to decompose a very high-order tensor into sparsely (weakly) connected low-order and small size tensors, typically 3rd-order and 4th-order tensors called cores. Top: The Tensor Chain (TC) model, equivalent to the Matrix Product State (MPS) with periodic boundary conditions (PBC). Middle: The Projected Entangled-Pair States (PEPS), also with PBC. Bottom: The Hierarchical Tucker (HT) decomposition, which is a special case of the Tree Tensor Network State (TTNS).

[7] Strictly speaking, the minimum set of internal indices $\{R_1, R_2, R_3, \ldots\}$ is called the rank (bond dimensions) of a specific tensor network.

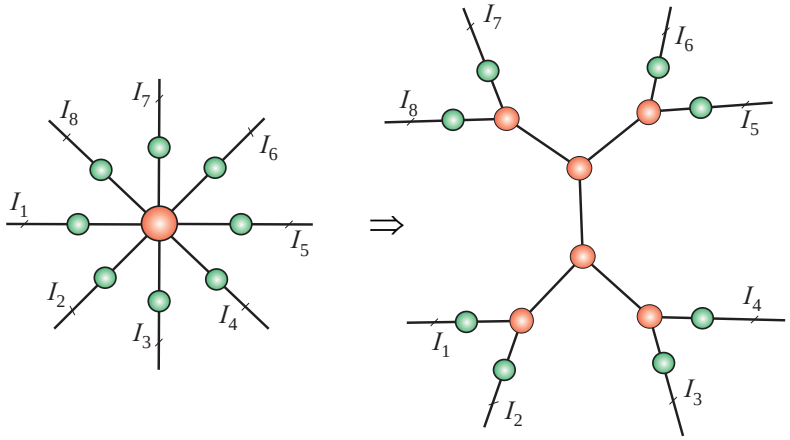


Figure 21: The standard Tucker decomposition for an 8th-order tensor and its transformation into an equivalent Hierarchical Tucker (HT) model using interconnected 3rd-order core tensors.

3.1 Hierarchical Tucker (HT) and Tensor Tree Network State (TTNS) Models

Hierarchical Tucker (HT) decompositions (also called hierarchical tensor representation) have been introduced in [113] and also independently in [106], see also [108, 158, 181, 263] and references therein[8]. The general form of HT decompositions requires splitting of the modes of a tensor in a hierarchical way, which results in a binary tree containing a subset of modes at each branch; an example of a binary tree is given in the bottom of Figure 20. In tensor networks based on binary trees, all the cores are of order of three or less. Observe that the HT model does not contain any loops, i.e., no edges connecting a node with itself; also, the splitting of the modes of the original data tensor by binary tree edges is performed through a suitable matricization [107, 108]. The *dimension tree* within the HT format is chosen *a priori* and defines the topology of the HT decomposition. Intuitively, the dimension tree specifies which groups of modes are "separated" from other groups of modes, so that a sequential HT decomposition can be performed via a (truncated) SVD applied to suitably matricized tensor. [107, 108, 112, 263].

One of the simplest and most straightforward choices of a dimension tree is the linear and unbalanced tree, which gives rise to the tensor-train (TT) decomposition, discussed in detail in Part 2).

The HT decompositions lead naturally to a distributed Tucker decomposi-

[8] The HT model was developed independently, from a different perspective, in the chemistry community under the name MultiLayer Multi-Configurational Time-Dependent Hartree method (ML-MCTDH) [275]. Furthermore, the PARATREE model, developed independently for signal processing applications [229], is quite similar to the HT model [106].

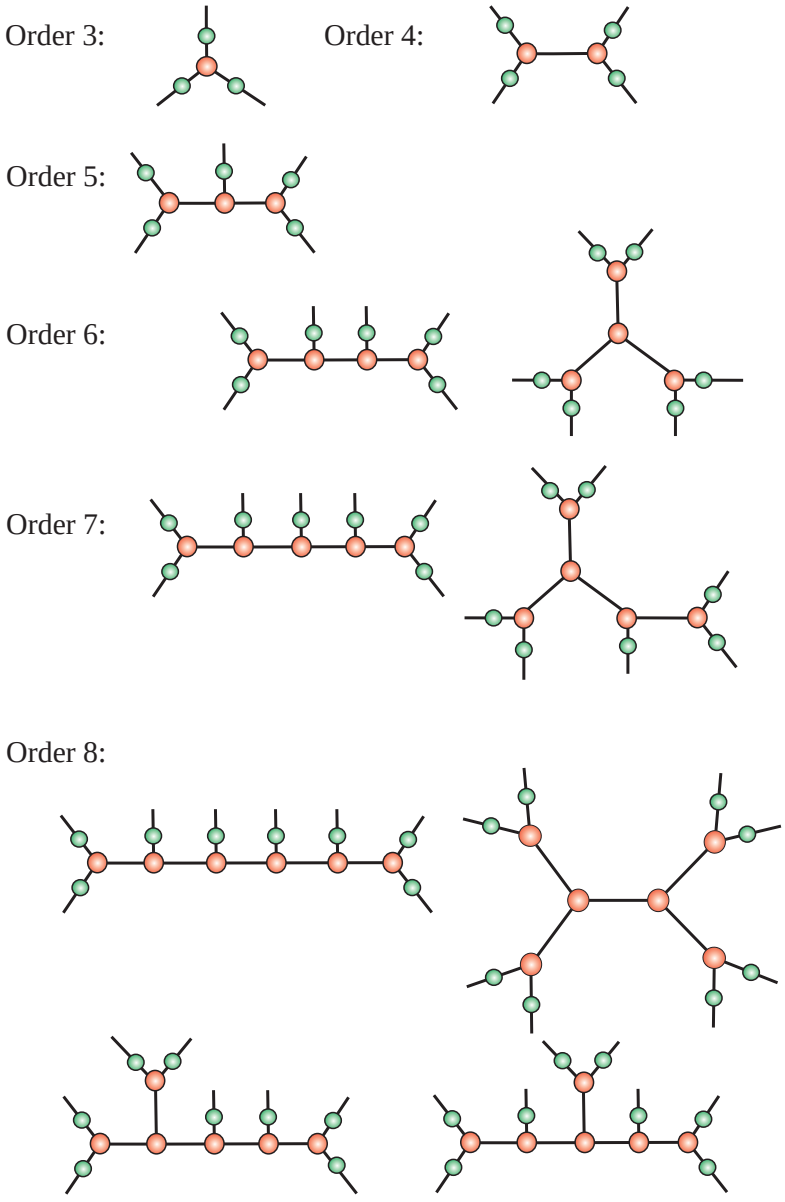


Figure 22: Examples of HT/TT models (formats) for distributed Tucker decompositions with 3rd-order cores, for different orders of data tensors. Green circles denote factor matrices (which can be absorbed by core tensors), while red circles indicate cores.

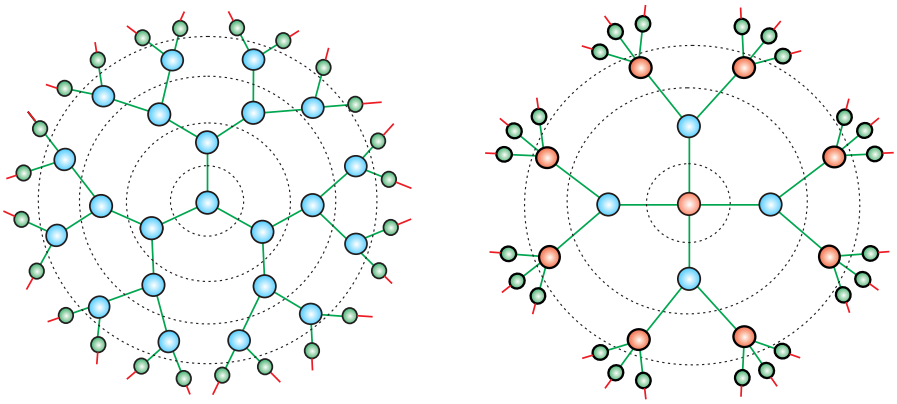


Figure 23: The Tree Tensor Network State (TTNS) with 3rd-order and 4th-order cores for the representation of 24th-order data tensors. The TTNS can be considered both as a generalization of HT/TT formats and as a distributed model for the Tucker-N decomposition (see Section 4.3).

tion, where a single core tensor is replaced by interconnected cores of lower-order, resulting in a distributed network in which only some cores are connected directly with factor matrices, as illustrated in Figure 21.

A simple approach to reduce the size of a core tensor in the standard Tucker decomposition (typically, for $N > 5$) would be to apply the concept of distributed tensor networks (DTNs). The DTNs assume two kinds of cores (blocks), the internal cores (nodes) which have no free edges and external cores which do have free edges representing physical indices of a data tensor (see also Section 3.4). Such distributed representations of tensors are not unique. Figure 22 illustrates that for a 6th-order tensor, there are only two tensor networks for its representation, while for a 5th-order we have 5, and for a 10th-order tensor there are 11 possible HT architectures [158, 257].

The tree tensor network state (TTNS) model, whereby all nodes are of 3rd-order or higher, can be considered as a generalization of the Hierarchical Tucker (HT) decomposition, as illustrated by two examples in Figure 23 (see [194, 233] and references therein). A more detailed mathematical description of the TTNS, which can be considered as a natural generalization of the Tucker decomposition, is given in Section 4.3.

3.2 Tensor Train (TT) network

The Tensor Train (TT) format was first proposed in numerical analysis and scientific computing in [204, 209]. It can be interpreted as a special case of the HT, where all nodes of the underlying tensor network are aligned while the leaf matrices in a tensor tree are assumed to be identities and thus need not be

stored. Figure 24 presents the concept of TT decomposition of an Nth-order tensor where the entries are computed as a cascaded (multilayer) multiplication of appropriate matrices. It is important to highlight that TT networks can be applied not only for the approximation of tensorized vectors but also for scalar multivariate functions, matrices, and even large-scale low-order tensors, as illustrated in Figure 25(for detail see Part 2).

In the quantum physics community, the TT format is known as the Matrix Product State (MPS) representation with the Open Boundary Conditions (OBC) and was introduced in 1987 as the ground state of the 1D AKLT model [6]. It was subsequently extended by many researchers[9].

An important advantage of the TT format over the HT format is its simpler practical implementation, as no binary tree needs to be determined (see Part 2). Other attractive properties of the TT-decomposition are its ability to efficiently compress the data and to perform basic mathematical operations on tensors directly in the TT-format (that is, employing only core tensors). These include matrix-by-matrix and matrix-by-vector multiplications, tensor addition, and the entry-wise (Hadamard) product of tensors. These operations produce tensors, also in the TT-format, which generally exhibit increased TT ranks. A detailed description of basic operations supported by TT formats is given in Part 2.

The TT rank is defined as the $(N-1)$-tuple of minimum rank in the form

$$\text{rank}_{\text{TT}}(\underline{\mathbf{X}}) = \mathbf{r}_{TT} = \{R_1, \ldots, R_{N-1}\}, \quad R_n = \text{rank}(\mathbf{X}_{<n>}), \tag{30}$$

where $\mathbf{X}_{<n>} \in \mathbb{R}^{I_1 \cdots I_n \times I_{n-1} \cdots I_N}$ is an nth canonical matricization of the tensor $\underline{\mathbf{X}}$. For more detail regarding the TT rank, its uniqueness, and relation to the CP tensor rank see [204].

The number of data samples scales linearly in the tensor order, N, and the size, I, and quadratically in the rank, R, that is

$$\sum_{n=1}^{N} R_{n-1} R_n I_n \sim \mathcal{O}(NR^2 I), \quad R := \max\{R_n\}, \quad I := \max\{I_n\}. \tag{31}$$

Since the TT-rank determines memory requirements of a tensor train, it has a strong impact on the complexity, i.e. the suitability of tensor train representation of a given raw data tensor.

Some remarks. A full tensor with $C = \prod_{n=1}^{N} I_n$ entries can be reconstructed by a TT from only $\mathcal{O}(\log C)$ samples [109], since only TT cores need to be stored and processed; this makes the number of data samples linear in the order, N, of a data tensor. The resulting reduction in dimensionality both helps with

[9] In fact, the TT was rediscovered several times under different names: MPS, valence bond states, and density matrix renormalization group (DMRG) [280]. The DMRG usually refers not only to a tensor network format but also the efficient computational algorithms (see also [231] and references therein). Also, in quantum physics the ALS algorithm is called one-site DMRG, while the Modified ALS is known as the two-site DMRG (for more detail, see Part 2 and [128, 201, 215, 232, 267, 271, 280] and references therein

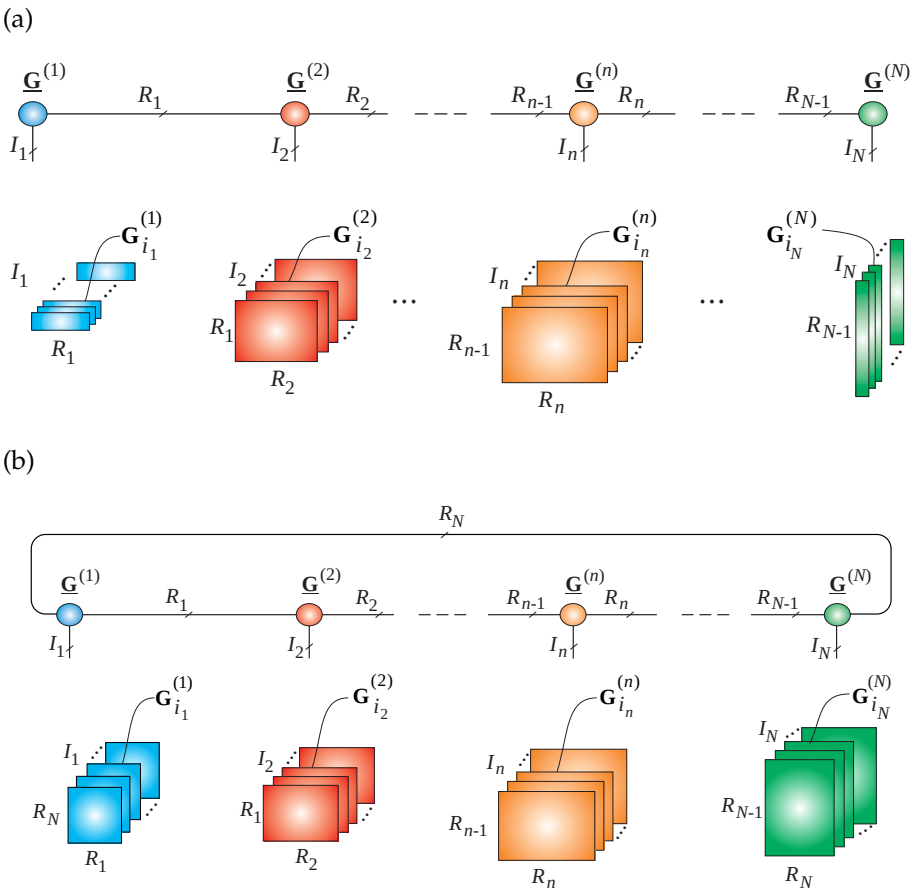


Figure 24: Comparison of the concept of the tensor train and tensor chain decompositions (MPS with OPC and PBC, respectively) for an Nth-order data tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. (a) Tensor Train (TT) can be mathematically described as $x_{i_1,i_2,\ldots,i_N} = \mathbf{G}^{(1)}_{i_1}\, \mathbf{G}^{(2)}_{i_2} \cdots \mathbf{G}^{(N)}_{i_N}$, where the slice matrices of TT cores $\underline{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$ are defined as $\mathbf{G}^{(n)}_{i_n} = \underline{\mathbf{G}}^{(n)}(:, i_n, :) \in \mathbb{R}^{R_{n-1} \times R_n}$ with $R_0 = R_N = 1$; while (b) for Tensor Chain (TC) entries of a tensor are expressed as $x_{i_1,i_2,\ldots,i_N} = \operatorname{tr}\left(\mathbf{G}^{(1)}_{i_1}\, \mathbf{G}^{(2)}_{i_2} \cdots \mathbf{G}^{(N)}_{i_N}\right) = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_N=1}^{R_N} g^{(1)}_{r_N,i_1,r_1}\, g^{(2)}_{r_1,i_2,r_2} \cdots g^{(N)}_{r_{N-1},i_N,r_N}$, where the lateral slices of TC cores $\mathbf{G}^{(n)}_{i_n} = \underline{\mathbf{G}}^{(n)}(:, i_n, :) \in \mathbb{R}^{R_{n-1} \times R_n}$ and $g^{(n)}_{r_{n-1},i_n,r_n} = \underline{\mathbf{G}}^{(n)}(r_{n-1}, i_n, r_n)$ for $n = 1, 2, \ldots, N$, with $R_0 = R_N > 1$.

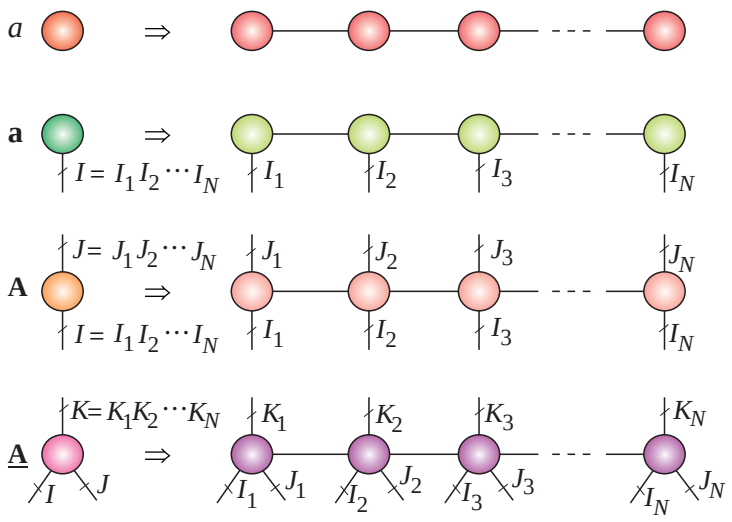


Figure 25: Forms of tensor train decompositions for a scalar, a, vector, $\mathbf{a} \in \mathbb{R}^I$, matrix, $\mathbf{A} \in \mathbb{R}^{I \times J}$, and 3rd-order tensor, $\underline{\mathbf{A}} \in \mathbb{R}^{I \times J \times K}$ (by applying a suitable tensorization).

the amount of required storage and also considerably reduces computational complexity, as mathematical operations are performed only on low-order and small size core tensors.

A drawback of the TT format is that the ranks of a tensor train decomposition depend on the ordering (permutation) of the modes, which gives different size of cores for different ordering. The optimal ordering for specific datasets remains a challenging problem, which can be partially alleviated by applying TT with periodic boundary conditions (PBC), that is, the Tensor Chain (TC) with a loop.

3.3 Tensor Networks with loops: PEPS, MERA and Honey-Comb Lattice (HCL)

An important issue in tensor networks is the rank-complexity trade-off in the design. Namely, the main idea behind TNs is to dramatically reduce computational cost and provide distributed storage through low-rank TN approximation, however, the TT/HT ranks, R_n, of 3rd-order core tensors sometimes increase rapidly with the order of a data tensor and/or increase of a desired approximation accuracy, for any choice of a tree of tensor network [158]. These ranks can be often kept under control through hierarchical two-dimensional

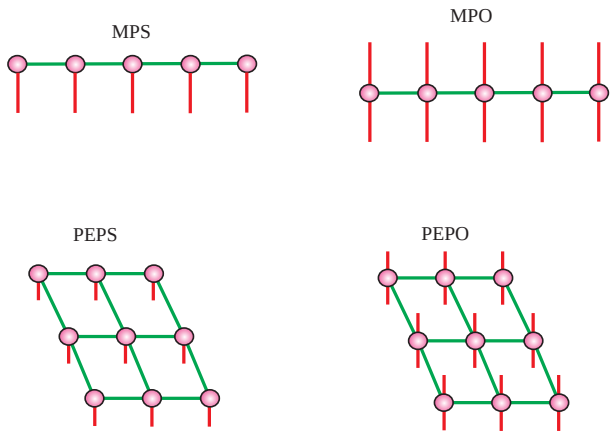


Figure 26: Class of 1D and 2D tensor train networks with open boundary conditions (OBC): the Matrix Product State (MPS) or (vector) Tensor Train (TT), the Matrix Product Operator (MPO) or Matrix TT, the Projected Entangled-Pair States (PEPS) or Tensor Product State (TPS), and the Projected Entangled-Pair Operators (PEPO).

TT models[10] called the PEPS (Projected Entangled Pair States) and PEPO (Projected Entangled Pair Operators) tensor networks, which contain loops, as shown in Figure 26. In the PEPS and PEPO, the ranks are kept considerably smaller at a cost of employing 5th- or even 6th-order core tensors and the associated higher computational complexity with respect to the ranks [95, 101].

Even with the PEPS/PEPO architectures, for very high-order tensors, the ranks (internal size of cores) may increase rapidly with an increase in the desired accuracy of approximation. For further control of the ranks, alternative tensor networks can be employed, such as: (1) the Honey-Comb Lattice (HCL) which uses 3rd-order cores, and (2) the Multi-scale Entanglement Renormalization Ansatz (MERA) which consist of both 3rd- and 4th-order core tensors (see Figure 27) [158, 201]. The ranks are often kept considerably small through special architectures of such TNs, at the expense of higher computational complexity with respect to tensor contractions due to many loops.

Compared with the PEPS and PEPO formats, the main advantage of the HCL and MERA formats is that the order and size of each core tensor in the internal tensor network structure is often much smaller, which dramatically reduces the number of free parameters and provides more efficient distributed storage of huge-scale data tensors. Indeed, even a simple TT with a single loop (tensor chain) can reduce TT ranks considerably. To this end, the highly

[10] An 'entangled state' is a tensor that cannot be represented as an elementary rank-1 tensor. The state is called 'projected' because it is not a real physical state but a projection onto some subspace. The term 'pair' refers to the entanglement being considered only for maximally entangled state pairs [117, 201].

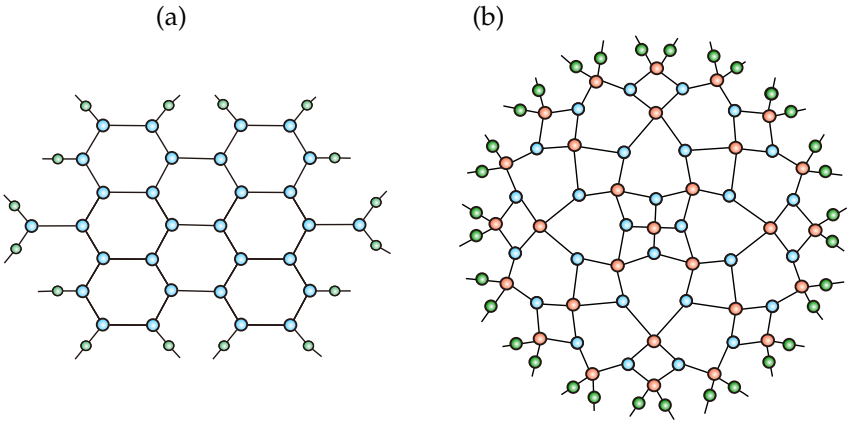


Figure 27: Examples of TN architectures with loops. (a) Honey-Comb Lattice (HCL) for a 16th-order tensor. (b) MERA for a 32th-order tensor.

complex contractions in tensor networks with loops (e.g., PEPS/PEPO, HCL, MERA) are in practice performed approximately [82, 135].

3.4 Concatenated (distributed) representation of TT networks

Algorithms for computation on tensor networks typically scale polynomially with the rank, R_n, or size, I_n, of the core tensors, so that the computations quickly become intractable with the increase in R_n, thus suggesting networks containing core tensors of small dimension. A step towards reducing storage and computational requirements would be to reduce the size of core tensors by increasing their number through distributed tensor networks (DTNs), as illustrated in Figure 27. A DTN consists of two kinds of relatively small-size cores (nodes), internal nodes which have no free edges and external nodes which have free edges representing natural (physical) indices of a data tensor. The underpinning idea is that each core tensor in an original TN is replaced by another TN (see Figure 28 for TT networks), resulting in a distributed TN in which only some core tensors are associated with physical (natural) modes of the original data tensor [127].

The obvious advantage of DTNs is that the size of each core tensor in the internal tensor network structure is usually much smaller than the size of the initial core tensor; this allows for a better management of distributed storage, and often in the total number of network parameters to be reduced through distributed computing. However, compared to initial tree structures, the contraction of the resulting distributed tensor network becomes much more difficult because of the loops in the architecture.

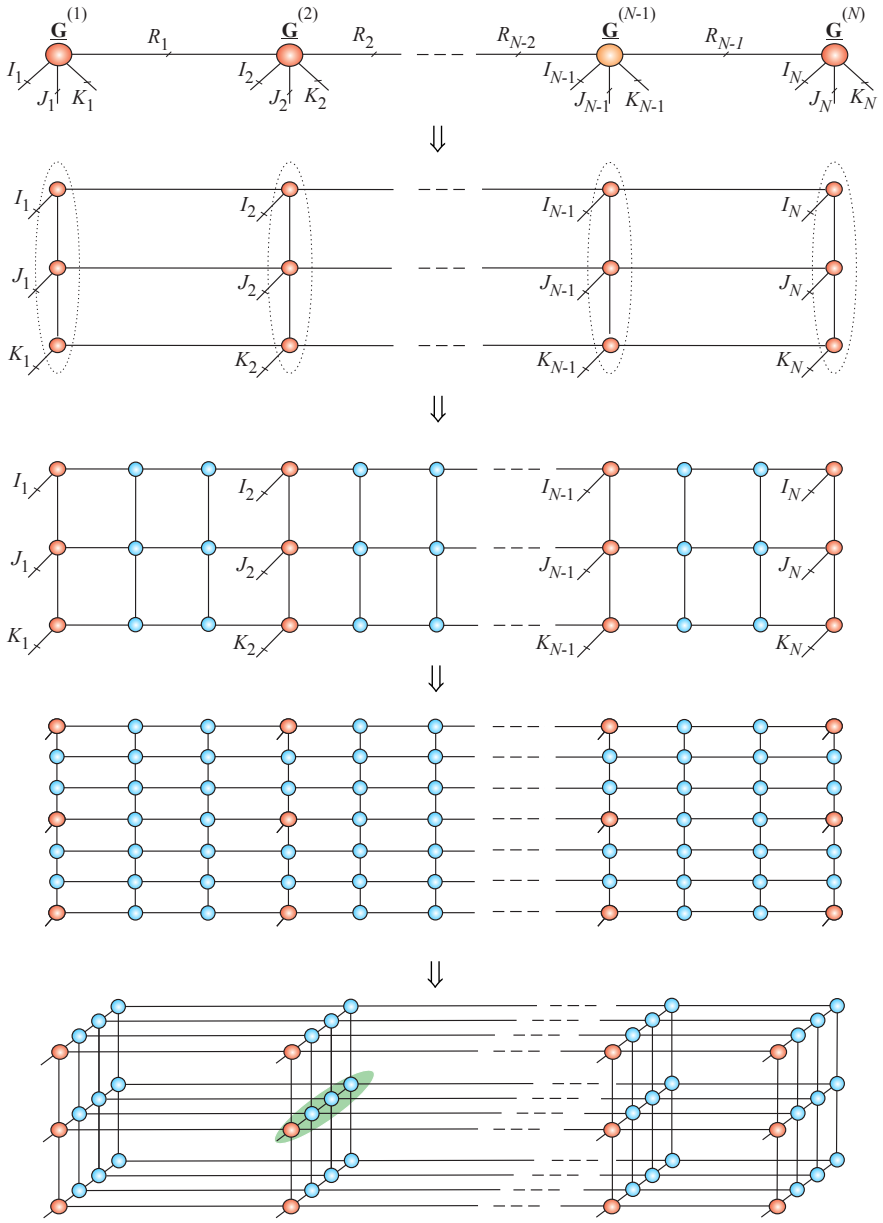


Figure 28: Graphical representation of a large-scale data tensor via its TT model (top panel), the PEPS model of the TT , and its transformation to a distributed 2D (second from bottom) and 3D (bottom) tensor train network.

Table 4: Links between tensor networks (TNs) and graphical models used in Machine Learning (ML) and Statistics. The corresponding categories are not exactly the same, but have general correspondence.

Tensor Networks	Neural Networks and Graphical Models in ML/Statistics
TT/MPS	Hidden Markov Models (HMM)
HT/TTNS	Deep Learning NN, Gaussian Mixture Model (GMM)
TNS/PEPS	Markov Random Field (MRF), Conditional Random Field (CRF)
MERA	Deep Belief Networks (DBN)
ALS, DMRG/MALS Algorithms	Forward-Backward Algorithms, Block Nonlinear Gauss-Seidel Methods

3.5 Links between TNs and Machine Learning models

Table 4 summaries the connections of tensor networks with graphical and neural network models in machine learning and statistics [54,55,64,65,138,192,198, 285]. We outline these links at a conceptual level, as more research is needed to establish deeper and more precise relationships.

3.6 Changing the structure of tensor networks

An advantage of a graphical representation of tensor networks is that the graphs allow us to perform complex mathematical operations on core tensors in an easy to understand and intuitive way. Another important advantage is the ability to modify (optimize) the topology of a TN, while keeping the original physical modes intact. The so optimized topologies yield simplified or more convenient graphical representations of a higher-order data tensor and facilitate practical applications [116,117,127,288]. In particular:

- A change in topology to a HT/TT tree structure provides reduced computational complexity, through iterative contraction of core tensors and enhanced stability of algorithms;
- TNs with cycles can be modified so as to completely eliminate the cycles or to reduce their number;
- The opportunity for joint group (linked) analysis of tensor networks with different topologies. Topology modifications may produce identical or

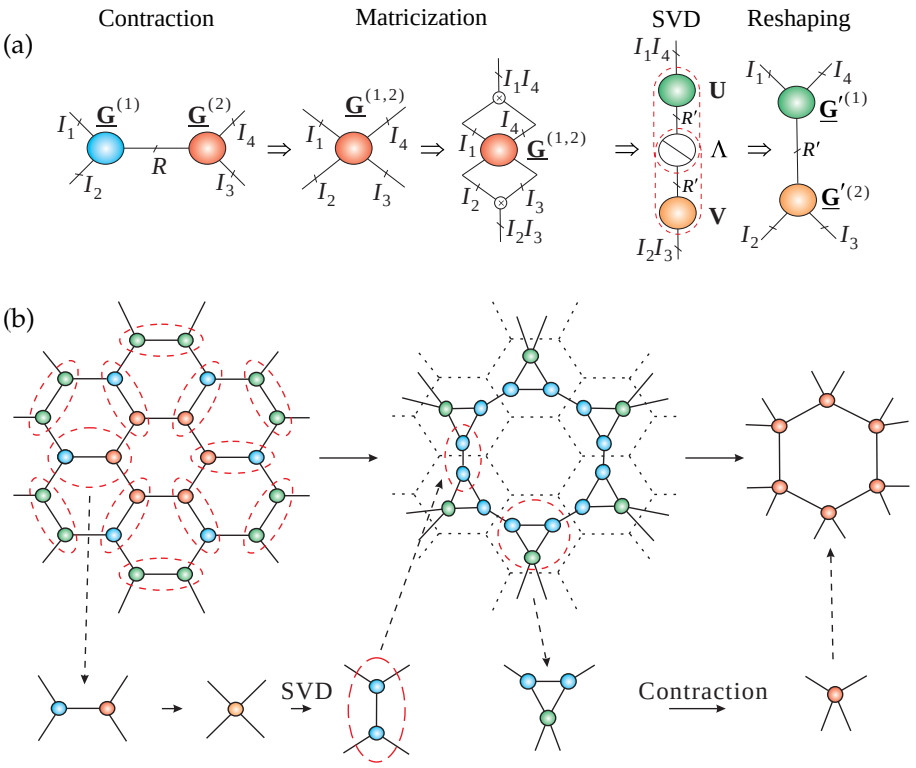


Figure 29: Illustration of basic transformations of a tensor network. (a) Contraction, matricization, matrix factorization (SVD) and reshaping of matrices back into tensors. (b) Transformation of a Honey-Comb lattice into a Tensor Chain (TC) via tensor contractions and the SVD.

very similar structures of diverse data tensors which could be easier to compare and jointly analyze.

It is important to note that, due to the iterative way in which tensor contractions are performed, the computational requirements associated with tensor contractions are usually much smaller for tree-structured networks than for tensor networks containing many cycles. Therefore, for stable computations, it is advantageous to transform a tensor network with cycles into a tree structure.

Applications for TN transformations. In order to modify tensor structures, we may perform sequential core contractions, followed by the unfolding of these contracted tensors into matrices, matrix factorizations (typically truncated SVD) and finally reshaping of such matrices back to new core tensors, as illustrated in Figures 29 (a), (b), (c).

The example in Figure 29 (a) shows that, in the first step a contraction of two core tensors, $\underline{\mathbf{G}}^{(1)} \in \mathbb{R}^{I_1 \times I_2 \times R}$ and $\underline{\mathbf{G}}^{(2)} \in \mathbb{R}^{R \times I_3 \times I_4}$, is performed to give

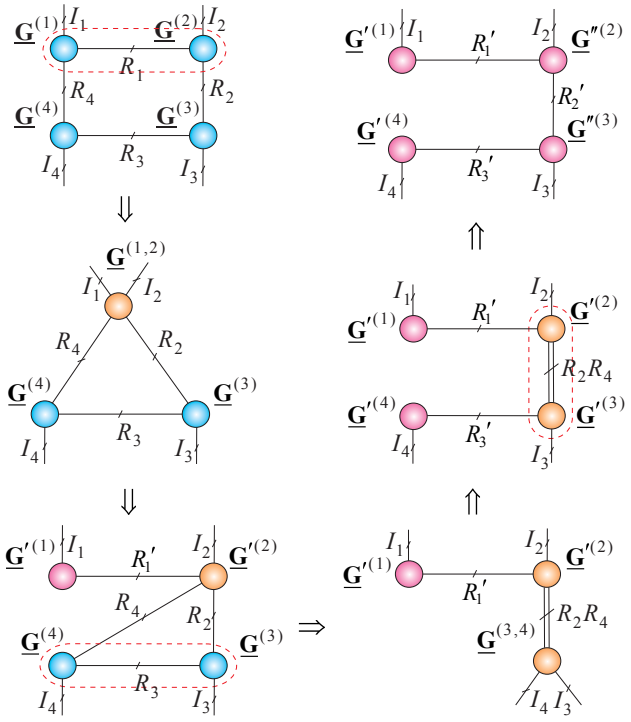


Figure 30: Transformation of the Tensor Chain (TC) into the standard Tensor Train (TT).

the tensor:

$$\underline{\mathbf{G}}^{(1,2)} = \underline{\mathbf{G}}^{(1)} \times_3^1 \underline{\mathbf{G}}^{(2)} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4}, \tag{32}$$

with entries $g^{(1,2)}_{i_1,i_2,i_3,i_4} = \sum_{r=1}^{R} g^{(1)}_{i_1,i_2,r}\, g^{(2)}_{r,i_3,i_4}$. In the next step, the tensor $\underline{\mathbf{G}}^{(1,2)}$ is transformed into a matrix via matricization, followed by a low-rank matrix factorization using the SVD, to give

$$\mathbf{G}^{(1,2)}_{i_1 i_4; i_2 i_3} \cong \mathbf{U}\mathbf{S}\mathbf{V}^T \in \mathbb{R}^{I_1 I_4 \times I_2 I_3}. \tag{33}$$

In the final step, the factor matrices, $\mathbf{U}\mathbf{S}^{1/2} \in \mathbb{R}^{I_1 I_4 \times R'}$ and $\mathbf{V}\mathbf{S}^{1/2} \in \mathbb{R}^{R' \times I_2 I_3}$, are reshaped into new core tensors, $\underline{\mathbf{G}}'^{(1)} \in \mathbb{R}^{I_1 \times R' \times I_4}$ and $\underline{\mathbf{G}}'^{(2)} \in \mathbb{R}^{R' \times I_2 \times I_3}$.

The above tensor transformation procedure is quite general, and is applied in Figure 29 (b) to transform a Honey-Comb lattice into a tensor chain (TC) [288], while Figure 30 illustrates the conversion of a tensor chain (TC) into TT/MPS with OBC.

To convert a TC into TT/MPS, in the first step, we perform a contraction of two tensors, $\underline{\mathbf{G}}^{(1)} \in \mathbb{R}^{I_1 \times R_4 \times R_1}$ and $\underline{\mathbf{G}}^{(2)} \in \mathbb{R}^{R_1 \times R_2 \times I_2}$, as:

$$\underline{\mathbf{G}}^{(1,2)} = \underline{\mathbf{G}}^{(1)} \times_3^1 \underline{\mathbf{G}}^{(2)} \in \mathbb{R}^{I_1 \times R_4 \times R_2 \times I_2}, \tag{34}$$

for which the entries $g^{(1,2)}_{i_1,r_4,r_2,i_2} = \sum_{r_1=1}^{R_1} g^{(1)}_{i_1,r_4,r_1}\ g^{(2)}_{r_1,r_2,i_2}$. In the next step, the tensor $\underline{\mathbf{G}}^{(1,2)}$ is transformed into a matrix, followed by a truncated SVD:

$$\mathbf{G}^{(1,2)}_{(1)} \cong \mathbf{U}\mathbf{S}\mathbf{V}^T \in \mathbb{R}^{I_1 \times R_4 R_2 I_2}. \tag{35}$$

Finally, the matrices, $\mathbf{U} \in \mathbb{R}^{I_1 \times R_1'}$ and $\mathbf{V}\mathbf{S} \in \mathbb{R}^{R_1' \times R_4 R_2 I_2}$, are reshaped back into the core tensors, $\underline{\mathbf{G}}'^{(1)} = \mathbf{U} \in \mathbb{R}^{1 \times I_1 \times R_1'}$ and $\underline{\mathbf{G}}'^{(2)} \in \mathbb{R}^{R_1' \times R_4 \times R_2 \times I_2}$. The procedure is repeated all over again for different pairs of cores, as illustrated in Figure 30.

3.7 Generalized Tensor Networks

The fundamental TNs considered so far assume that links between the cores are expressed by tensor contractions. In general, links between the core tensors (or tensor subnetworks) can also be expressed via other mathematical linear/multilinear or nonlinear operators, such as the outer (tensor) product, Kronecker product, Hadamard product and convolution operator. In simple scenarios for example for the outer product this leads to Block Term Decomposition (BTD) [72,75,162,242] and in the case Kronceker product to the Kronecker Tensor Decomposition (KTD) [220, 221].

Figure 31 illustrates such a BTD model for a 6-th-order tensor, where the links between the components are expressed via outer products, while Figure 32 shows flexible models suitable for very high-order tensors. The advantage of BTD over other TN models is their flexibility[11] and the ability to model more complex data structures by approximating very high-order tensors through a relatively small number of low-order cores.

The BTD and KTD models can be expressed mathematically, for example, in simple nested (hierarchical) forms, given by

$$\text{BTD} : \underline{\mathbf{X}} \quad \cong \quad \sum_{r=1}^{R} (\underline{\mathbf{A}}_r \circ \underline{\mathbf{B}}_r), \tag{36}$$

$$\text{KTD} : \underline{\tilde{\mathbf{X}}} \quad \cong \quad \sum_{r=1}^{R} (\underline{\mathbf{A}}_r \otimes \underline{\mathbf{B}}_r), \tag{37}$$

where e.g. for BTD, each factor tensor can be represented recursively as $\underline{\mathbf{A}}_r \cong \sum_{r_1=1}^{R_1} (\underline{\mathbf{A}}^{(1)}_{r_1} \circ \underline{\mathbf{B}}^{(1)}_{r_1})$ or $\underline{\mathbf{B}}_r \cong \sum_{r_2=1}^{R_2} \underline{\mathbf{A}}^{(2)}_{r_2} \circ \underline{\mathbf{B}}^{(2)}_{r_2}$.

[11] Block term decompositions are closely related to constrained Tucker formats (with a sparse block core), Hierarchical Outer Product Tensor Approximation (HOPTA), and Kronecker Tensor Decomposition (KTD), applied to very high-order tensors [50, 220, 221, 226].

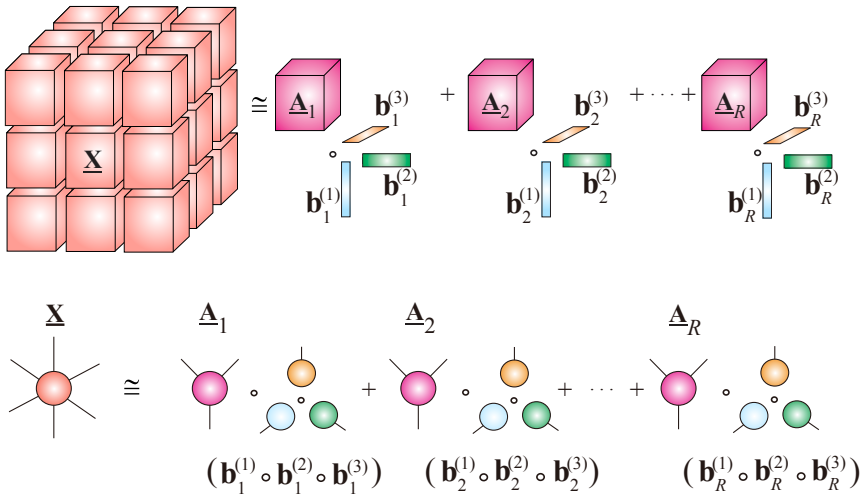


Figure 31: Block term decomposition (BTD) of a 6th-order block tensor, to yield $\underline{\mathbf{X}} = \sum_{r=1}^{R} \underline{\mathbf{A}}_r \circ \left(\mathbf{b}_r^{(1)} \circ \mathbf{b}_r^{(2)} \circ \mathbf{b}_r^{(3)}\right)$, for more detail see [72, 242].

Note that the $2N$th-order sub-tensors, $\underline{\mathbf{A}}_r \circ \underline{\mathbf{B}}_r$ and $\underline{\mathbf{A}}_r \otimes \underline{\mathbf{B}}_r$, have the same elements, just arranged differently. For example, if $\underline{\mathbf{X}} = \underline{\mathbf{A}} \circ \underline{\mathbf{B}}$ and $\underline{\mathbf{X}}' = \underline{\mathbf{A}} \otimes \underline{\mathbf{B}}$, where $\underline{\mathbf{A}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$ and $\underline{\mathbf{B}} \in \mathbb{R}^{K_1 \times K_2 \times \cdots \times K_N}$, then $x_{j_1, j_2, \ldots, j_N, k_1, k_2, \ldots, k_N} = x'_{k_1 + K_1(j_1 - 1), \ldots, k_N + K_N(j_N - 1)}$.

The definition of the tensor Kronecker product in the KTD model assumes that both core tensors, $\underline{\mathbf{A}}_r$ and $\underline{\mathbf{B}}_r$, have the same order [220, 221]. This is not a limitation, given that vectors and matrices can also be treated as tensors, e.g, a matrix of dimension $I \times J$ as is also a 3rd-order tensor of dimension $I \times J \times 1$. In fact, from the BTD/KTD models, many existing and new TDs/TNs can be derived by changing the structure and orders of factor tensors, $\underline{\mathbf{A}}_r$ and $\underline{\mathbf{B}}_r$. For example:

- If $\underline{\mathbf{A}}_r$ are rank-1 tensors of size $I_1 \times I_2 \times \cdots \times I_N$, and $\underline{\mathbf{B}}_r$ are scalars, $\forall r$, then (37) expresses the rank-R CP decomposition;
- If $\underline{\mathbf{A}}_r$ are rank-L_r tensors of size $I_1 \times I_2 \times \cdots \times I_R \times 1 \times \cdots \times 1$, in the Kruskal (CP) format, and $\underline{\mathbf{B}}_r$ are rank-1 tensors of size $1 \times \cdots \times 1 \times I_{R+1} \times \cdots \times I_N$, $\forall r$, then (37) expresses the rank-$(L_r \circ 1)$ BTD [72];
- If $\underline{\mathbf{A}}_r$ and $\underline{\mathbf{B}}_r$ are expressed by KTDs, we arrive at the Nested Kronecker Tensor Decomposition (NKTD), a special case of which is the Tensor Train (TT) decomposition [144, 204, 211]. Therefore, the BTD model in (37) can also be used for recursive TT-decompositions [204].

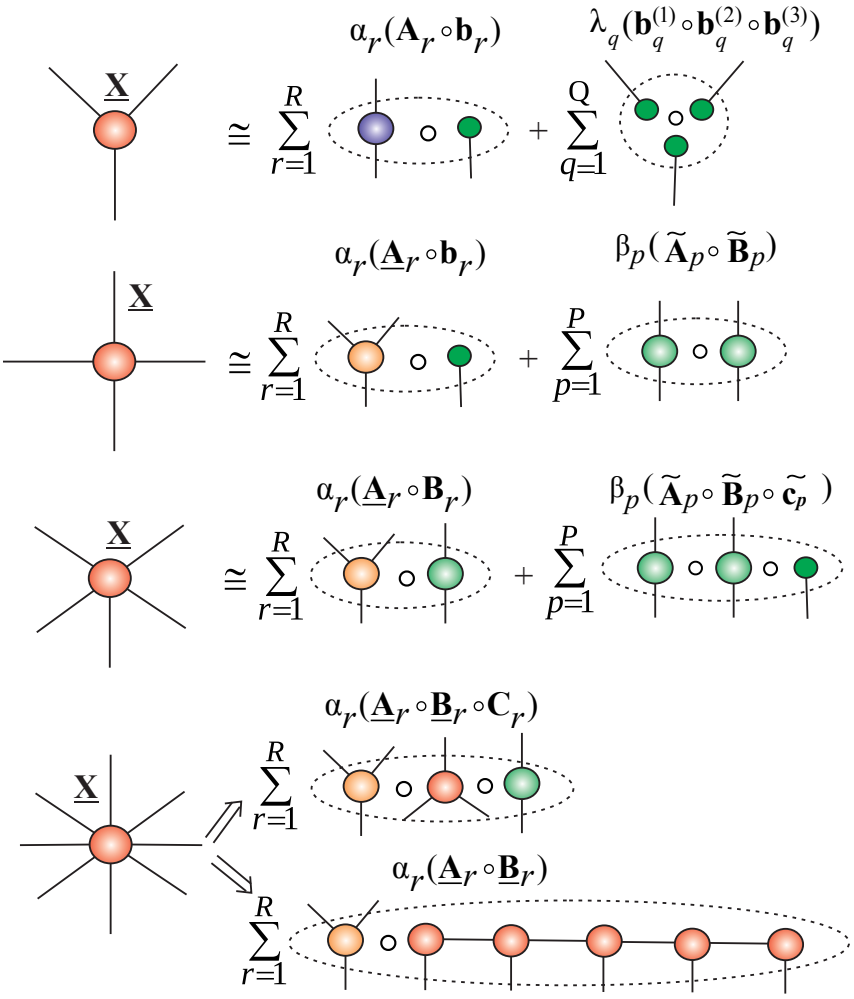


Figure 32: Conceptual model of generalized tensor networks for higher-order data tensors of different orders. For simplicity, we used outer (tensor) products, but conceptually other mathematical matrix/tensor operators can be employed. Each component (block tensor), $\underline{\mathbf{A}}_r$, $\underline{\mathbf{B}}_r$ and/or $\underline{\mathbf{C}}_r$, can be further hierarchically decomposed using suitable matrix/tensor decompositions. These models can be applied to very high-order tensors.

The generalized tensor network approach caters for a large variety of tensor decomposition models, some of which may find applications in scientific computing, signal processing or deep learning (see, eg., [49, 50, 55, 162, 224].

In this monograph, we will mostly focus on more established the Tucker and TT decompositions (and for some of their extensions), due to their conceptual simplicity, availability of stable and efficient algorithms and the possibility to naturally extend these models to more complex tensor networks. In other words, the Tucker and TT models are considered here as simplest prototypes, which can then serve as building blocks for more sophisticated tensor networks.

4 Constrained Tensor Decompositions: From Two-way to Multiway Component Analysis

The component analysis (CA) framework usually refers to the application of constrained matrix factorization techniques in order to extract components with specific properties from observed mixed signals and/or estimate the mixing matrix from observed mixed signals [53, 58, 70]. In the machine learning practice, to aid the well-posedness and uniqueness of the problem, component analysis methods exploit prior knowledge about the statistics and diversities of latent variables (hidden sources) within the data. Here, the diversities refer to different characteristics, features or morphology of latent variables which allow us to extract the desired components or features, for example, sparse or statistical independent components.

4.1 Constrained Low-Rank Matrix Factorizations

Two-way Component Analysis (2-way CA), in its simplest form, can be formulated as a constrained matrix factorization, typically, low-rank, of the form

$$\mathbf{X} = \mathbf{A}\mathbf{\Lambda}\mathbf{B}^T + \mathbf{E} = \sum_{r=1}^{R} \lambda_r \, \mathbf{a}_r \circ \mathbf{b}_r + \mathbf{E} = \sum_{r=1}^{R} \lambda_r \, \mathbf{a}_r \, \mathbf{b}_r^T + \mathbf{E}, \tag{38}$$

where $\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_R)$ is an optional diagonal scaling matrix. The potential constraints imposed on the factor matrices, $\mathbf{A}$ and/or $\mathbf{B}$, include orthogonality, sparsity, statistical independence, nonnegativity or smoothness. In the bilinear 2-way CA in (38), $\mathbf{X} \in \mathbb{R}^{I \times J}$ is a known matrix of observed data, $\mathbf{E} \in \mathbb{R}^{I \times J}$ represents residuals or noise, $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_R] \in \mathbb{R}^{I \times R}$ is the unknown mixing matrix with R basis vectors $\mathbf{a}_r \in \mathbb{R}^I$, and $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_R] \in \mathbb{R}^{J \times R}$, depending on applications, is the matrix of unknown components, factors, latent variables, or hidden sources, $\mathbf{b}_r \in \mathbb{R}^J$ (see Figure 34).

It should be noted that 2-way CA has an inherent symmetry. Indeed, Eq. (38) could also be written as $\mathbf{X}^T \approx \mathbf{B}\mathbf{A}^T$, thus interchanging the roles of sources and mixing process.

Algorithmic approaches to 2-way (matrix) component analysis are well established, and include Principal Component Analysis (PCA), Robust PCA (RPCA), Independent Component Analysis (ICA), Nonnegative Matrix Factorization (NMF), Sparse Component Analysis (SCA) and Smooth component Analysis (SmCA) [32, 53, 58, 186]. These techniques have become standard tools, in blind source separation (BSS), feature extraction, and classification paradigms. The columns of the matrix $\mathbf{B}$, which represent different latent components, are then determined by specific chosen constraints and should be, for example, (i) as statistically mutually independent as possible for ICA; (ii) as sparse as possible for SCA; (iii) as smooth as possible for SmCA; (iv) take only nonnegative values for NMF [11, 53, 58, 137].

Singular value decomposition (SVD) of the data matrix $\mathbf{X} \in \mathbb{R}^{I \times J}$ is a special, highly significant, case of the factorization in Eq. (38), and is given by

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T = \sum_{r=1}^{R} \sigma_r \, \mathbf{u}_r \circ \mathbf{v}_r = \sum_{r=1}^{R} \sigma_r \, \mathbf{u}_r \mathbf{v}_r^T, \tag{39}$$

where $\mathbf{U} \in \mathbb{R}^{I \times R}$ and $\mathbf{V} \in \mathbb{R}^{J \times R}$ are column-wise orthogonal matrices and $\mathbf{S} \in \mathbb{R}^{R \times R}$ is a diagonal matrix containing only nonnegative singular values σ_r in a monotonically non-increasing order.

According to the well known Eckart-Young theorem, the truncated SVD provides the optimal, in the least-squares (LS) sense, low-rank matrix approximation[12]. The SVD therefore, forms the backbone of low-rank matrix approximations (and consequently low-rank tensor approximations).

Another virtue of component analysis comes from the ability to perform simultaneous matrix factorizations:

$$\mathbf{X}_k \approx \mathbf{A}_k \mathbf{B}_k^T, \qquad (k = 1, 2, \ldots, K), \tag{40}$$

on several data matrices, $\mathbf{X}_k$, which represent linked datasets, subject to various constraints imposed on linked (interrelated) component (factor) matrices. In the case of orthogonality or statistical independence constraints, the problem in (40) can be related to models of group PCA/ICA through suitable pre-processing, dimensionality reduction and post-processing procedures [94, 110, 240]. The terms "group component analysis" and "joint multi-block data analysis" are used interchangeably to refer to methods which aim to identify links (correlations, similarities, statistical independence) between hidden components in data. In other words, the objective of group component analysis is to analyze the correlation, variability, and consistency of the latent components across multi-block datasets. The field of 2-way CA is maturing and has generated efficient algorithms for 2-way component analysis, especially for sparse/functional PCA/SVD, ICA, NMF and SCA [11, 58, 129, 299].

The rapidly emerging field of tensor decompositions is the next important step which naturally generalizes 2-way CA/BSS models and algorithms. Tensors, by virtue of multilinear algebra, offer enhanced flexibility in CA, in the sense that not all components need to be statistically independent, and can be instead smooth, sparse, and/or non-negative (e.g., spectral components). Furthermore, additional constraints can be used to reflect physical properties and/or diversities of spatial distributions, spectral and temporal patterns [48, 302]. We proceed to show how constrained matrix factorizations or 2-way CA models can be extended to multilinear models using tensor decompositions, such as the Canonical Polyadic (CP) and the Tucker decomposition, illustrated in Figures 33, 34 and 35.

The CP and Tucker decompositions have a long history, for recent surveys and more detailed historical information, we refer to [51, 108, 153].

[12] Mirsky [190] has generalized this optimality to arbitrary unitary invariant norms.

4.2 The CP Format

The CP decomposition (also called the CANDECOMP, PARAFAC, or Canonical Polyadic decomposition) decomposes an Nth-order tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, into a linear combination of terms, $\mathbf{b}_r^{(1)} \circ \mathbf{b}_r^{(2)} \circ \cdots \circ \mathbf{b}_r^{(N)}$, which are rank-1 tensors, and is given by [39, 119, 120]

$$\begin{aligned} \underline{\mathbf{X}} &\cong \sum_{r=1}^{R} \lambda_r \, \mathbf{b}_r^{(1)} \circ \mathbf{b}_r^{(2)} \circ \cdots \circ \mathbf{b}_r^{(N)} \\ &= \underline{\mathbf{\Lambda}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)} \\ &= [\![\underline{\mathbf{\Lambda}}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N)}]\!], \end{aligned} \tag{41}$$

where λ_r are non-zero entries of the diagonal core tensor $\underline{\mathbf{\Lambda}} \in \mathbb{R}^{R \times R \times \cdots \times R}$ and $\mathbf{B}^{(n)} = [\mathbf{b}_1^{(n)}, \mathbf{b}_2^{(n)}, \ldots, \mathbf{b}_R^{(n)}] \in \mathbb{R}^{I_n \times R}$ are factor (component) matrices (see Figure 33 and Figure 34).

Via the Khatri-Rao products (see Table 3), the CP decomposition can be equivalently expressed in a matrix/vector form as:

$$\begin{aligned} \mathbf{X}_{(n)} &\cong \mathbf{B}^{(n)} \mathbf{\Lambda} (\mathbf{B}^{(N)} \odot \cdots \odot \mathbf{B}^{(n+1)} \odot \mathbf{B}^{(n-1)} \odot \cdots \odot \mathbf{B}^{(1)})^T \\ &= \mathbf{B}^{(n)} \mathbf{\Lambda} (\mathbf{B}^{(1)} \odot_L \cdots \odot_L \mathbf{B}^{(n-1)} \odot_L \mathbf{B}^{(n+1)} \odot_L \cdots \odot_L \mathbf{B}^{(N)})^T \end{aligned} \tag{42}$$

and

$$\begin{aligned} \text{vec}(\underline{\mathbf{X}}) &\cong [\mathbf{B}^{(N)} \odot \mathbf{B}^{(N-1)} \odot \cdots \odot \mathbf{B}^{(1)}] \, \boldsymbol{\lambda} \\ &\cong [\mathbf{B}^{(1)} \odot_L \mathbf{B}^{(2)} \odot_L \cdots \odot_L \mathbf{B}^{(N)}] \, \boldsymbol{\lambda}, \end{aligned} \tag{43}$$

where $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_R]^T$ and $\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_R)$ is a diagonal matrix.

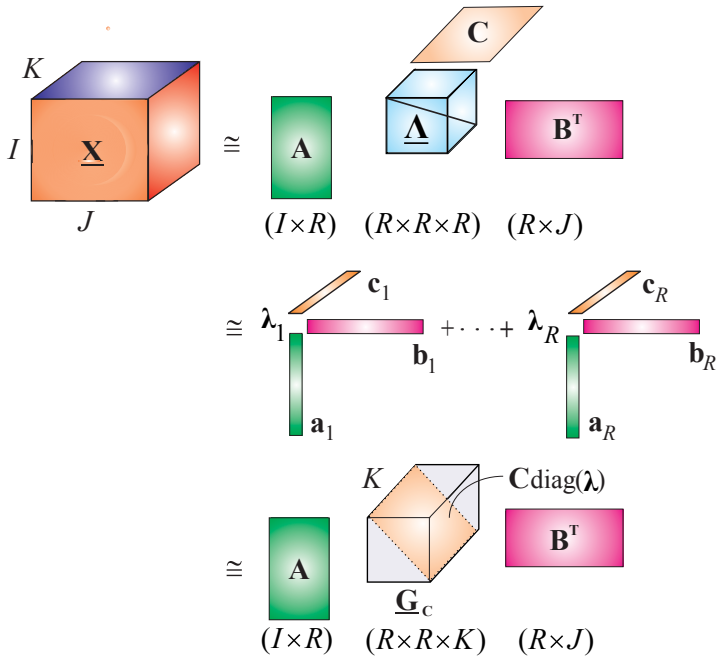
The rank of a tensor $\underline{\mathbf{X}}$ is defined as the smallest R for which the CP decomposition in (41) holds exactly.

Algorithms to compute CP decomposition. In real world applications, the signals of interest are corrupted by noise, so that the CP decomposition is rarely exact and has to be estimated by minimizing a suitable cost function. Such cost functions are typically of the Least-Squares (LS) type, in the form of the Frobenius norm $||\underline{\mathbf{X}} - [\![\underline{\mathbf{\Lambda}}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N)}]\!]||_F$, or Least Absolute Error (LAE) criteria [272]. The Alternating Least Squares (ALS) based algorithms minimize the cost function iteratively by individually optimizing each component (factor matrix, $\mathbf{B}^{(n)}$, while keeping the other component matrices fixed [52,59,105,119].

To illustrate the ALS principle, assume that the diagonal matrix $\mathbf{\Lambda}$ has been absorbed into one of the component matrices; then, by taking advantage of the Khatri-Rao structure, the component matrices, $\mathbf{B}^{(n)}$, can be updated sequentially as [153]

$$\mathbf{B}^{(n)} \leftarrow \mathbf{X}_{(n)} \left(\bigodot_{k \neq n} \mathbf{B}^{(k)} \right) \left(\circledast_{k \neq n} (\mathbf{B}^{(k)\,T} \mathbf{B}^{(k)}) \right)^{\dagger}. \tag{44}$$

(a) Standard block diagram for a 3rd-order tensor



(b) CP decomposition for a 4th-order tensor in the tensor network notation

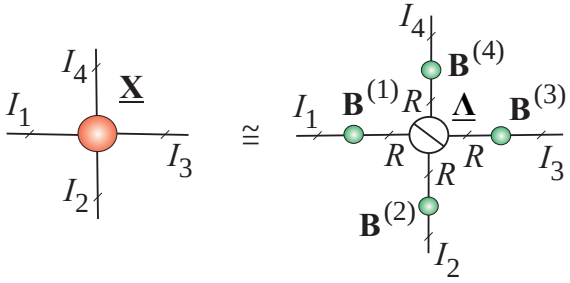


Figure 33: Representations of the CP decomposition. The objective of the CP decomposition is to estimate the factor matrices $\mathbf{B}^{(n)} \in \mathbb{R}^{I_n \times R}$. (a) The CP of a 3rd-order tensor in the form, $\underline{\mathbf{X}} \cong \underline{\mathbf{\Lambda}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} = \sum_{r=1}^{R} \lambda_r \, \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r = \underline{\mathbf{G}}_c \times_1 \mathbf{A} \times_2 \mathbf{B}$, with $\underline{\mathbf{G}} = \underline{\mathbf{\Lambda}}$ and $\underline{\mathbf{G}}_c = \underline{\mathbf{\Lambda}} \times_3 \mathbf{C}$. (b) The CP for a 4th-order tensor in the form $\underline{\mathbf{X}} \cong \underline{\mathbf{\Lambda}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \times_3 \mathbf{B}^{(3)} \times_4 \mathbf{B}^{(4)} = \sum_{r=1}^{R} \lambda_r \, \mathbf{b}_r^{(1)} \circ \mathbf{b}_r^{(2)} \circ \mathbf{b}_r^{(3)} \circ \mathbf{b}_r^{(4)}$.

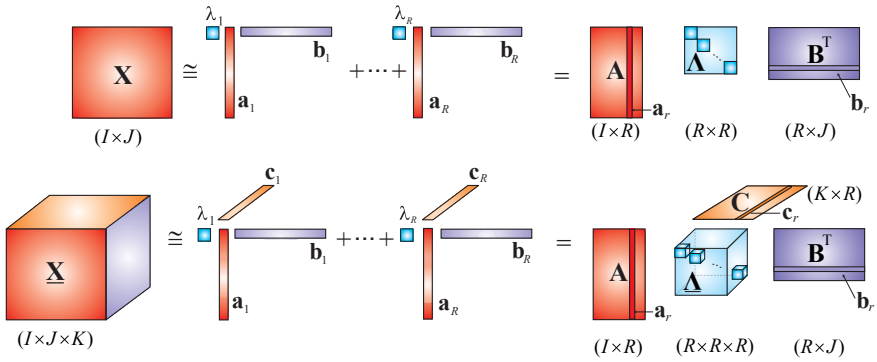


Figure 34: Analogy between a low-rank matrix factorization, $\mathbf{X} \cong \mathbf{A}\mathbf{\Lambda}\mathbf{B}^T = \sum_{r=1}^{R} \lambda_r \ \mathbf{a}_r \circ \mathbf{b}_r$, (top), and a simple low-rank tensor factorization (CP decomposition), $\underline{\mathbf{X}} \cong \underline{\mathbf{\Lambda}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} = \sum_{r=1}^{R} \lambda_r \ \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$, (bottom).

The main challenge (or bottleneck) in implementing ALS and Gradient Decent (GD) techniques for CP decomposition lies therefore in multiplying a matricized tensor and Khatri-Rao product (of factor matrices) [45] and in the computation of the pseudo-inverse of $(R \times R)$ matrices (see the basic ALS Algorithm 1.

The ALS approach is attractive for its simplicity, and often provides satisfactory performance for well defined problems with well separated and non-collinear components for high SNRs. For ill-conditioned problems, advanced algorithms typically exploit the rank-1 structure of the terms within CP decomposition to perform efficient computation and storage of the Jacobian and Hessian of the cost function [223, 242]. A more sophisticated parallel ALS algorithm for very large-scale tensors was proposed in [45].

Multiple random projections, tensor sketching and Giga-Tensor. Most of the existing algorithms for the computation of CP decomposition are based on the ALS or GD approaches, however, these can be too computationally expensive for huge tensors. Indeed, in general, algorithms for tensor decompositions have not yet reached the level of maturity and efficiency of low-rank matrix factorization (LRMF) methods. In order to employ efficient LRMF algorithms to tensors, we can either: (i) reshape the tensor at hand into a set of matrices using traditional matricizations, (ii) employ reduced randomized unfolding matrices, or (iii) perform random multiple projections of a data tensor. The principles of the approaches (i) and (ii) are self-evident, while the approach (iii) employs a multilinear product of an Nth-order tensor and $(N-2)$ random vectors, which are either chosen uniformly from a unit sphere or assumed to be i.i.d. Gaussian vectors [161]. For example, for a 3rd-order tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, we can use the set of random projections, $\mathbf{X}_{\bar{3}} = \underline{\mathbf{X}} \ \bar{\times}_3 \ \boldsymbol{\omega}_3 \in \mathbb{R}^{I_1 \times I_2}$,

Algorithm 1: **Basic ALS for the CP decomposition of a 3rd-order tensor**

Input: Data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$ and rank R
Output: Factor matrices $\mathbf{A} \in \mathbb{R}^{I \times R}$, $\mathbf{B} \in \mathbb{R}^{J \times R}$, $\mathbf{C} \in \mathbb{R}^{K \times R}$, and scaling vector $\boldsymbol{\lambda} \in \mathbb{R}^{R}$

1: Initialize $\mathbf{A}, \mathbf{B}, \mathbf{C}$
2: **while** not converged or iteration limit is not reached **do**
3: $\quad \mathbf{A} \leftarrow \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})(\mathbf{C}^T\mathbf{C} \circledast \mathbf{B}^T\mathbf{B})^{\dagger}$
4: $\quad$ Normalize column vectors of $\mathbf{A}$ to unit length (by computing a norm of each column vector and dividing each element of a vector by its norm)
5: $\quad \mathbf{B} \leftarrow \mathbf{X}_{(2)}(\mathbf{C} \odot \mathbf{A})(\mathbf{C}^T\mathbf{C} \circledast \mathbf{A}^T\mathbf{A})^{\dagger}$
6: $\quad$ Normalize column vectors of $\mathbf{B}$ to unit length
7: $\quad \mathbf{C} \leftarrow \mathbf{X}_{(3)}(\mathbf{B} \odot \mathbf{A})(\mathbf{B}^T\mathbf{B} \circledast \mathbf{C}^T\mathbf{C})^{\dagger}$
8: $\quad$ Normalize column vectors of $\mathbf{C}$ to unit length, store the norms in vector $\boldsymbol{\lambda}$
9: **end while**
10: **return** $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\boldsymbol{\lambda}$.

$\mathbf{X}_{\bar{2}} = \underline{\mathbf{X}} \;\bar{\times}_2\; \boldsymbol{\omega}_2 \in \mathbb{R}^{I_1 \times I_3}$ and $\mathbf{X}_{\bar{1}} = \underline{\mathbf{X}} \;\bar{\times}_1\; \boldsymbol{\omega}_1 \in \mathbb{R}^{I_2 \times I_3}$, where the vectors $\boldsymbol{\omega}_n \in \mathbb{R}^{I_n}$, $n = 1, 2, 3$, are suitably chosen random vectors. Note that random projections in such a case are non-typical – instead of using projections for dimensionality reduction, they are used to reduce any order tensor to matrices and consequently transform the CP decomposition problem to constrained matrix factorizations problem, which can be solved via simultaneous (joint) matrix diagonalization [41, 71]. Usually, even a small number of random projections, such as $\mathcal{O}(\log R)$ is sufficient to preserve the spectral information in a tensor. This mitigates the problem of the dependence on the eigen-gap that plagued earlier tensor-to-matrix reductions. Although a uniform random sampling may experiences problem for tensors with spiky elements, this approach often outperforms the standard CP-ALS decomposition algorithms [161].

Alternative algorithms for the CP decomposition of huge-scale tensors, include tensor sketching – a random mapping technique, which exploits kernel methods and regression [278], and the class of distributed algorithms such as the CPOPT [4], the DFacTo [45] and the GigaTensor which is based on Hadoop / Mapreduce paradigm [134].

Constraints. Under rather mild conditions, the CP decomposition is generally unique by itself [237], and does not require additional constraints on the factor matrices to achieve uniqueness, which makes it a powerful and useful tool. Of course, if the components in one or more modes are known to possess some properties e.g., they are known to be nonnegative, orthogonal, statistically independent or sparse, such prior knowledge may be incorporated to relax uniqueness conditions. More importantly, such constraints may enhance the accuracy and stability of CP decomposition algorithms and also facilitate better

physical interpretability of the extracted components [81,147,174,236,244,296].

Applications. The CP decomposition has already been established as an advanced tool for blind signal separation in vastly diverse branches of signal processing and machine learning, such as in audio and speech processing, biomedical engineering, chemometrics, and machine learning [5,153,193,239]. It is also routinely used in exploratory data analysis, where the rank-1 terms capture essential properties of dynamically complex datasets [56, 58, 160], while in wireless communication systems, signals transmitted by different users correspond to rank-1 terms in the case of line-of-sight propagation [234] and therefore admit the analysis in the CP format. Another potential application is in harmonic retrieval and direction of arrival problems, where real or complex exponentials have rank-1 structures, for which the use of CP decomposition is quite natural [235, 243].

4.3 The Tucker Tensor Format

Compared to the CP decomposition, the Tucker decomposition provides a more general factorization of an Nth-order tensor into a relatively small size core tensor and factor matrices, and can be expressed as follows [259]:

$$\begin{aligned} \underline{\mathbf{X}} &\cong \sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} g_{r_1 r_2 \cdots r_N} \left(\mathbf{b}_{r_1}^{(1)} \circ \mathbf{b}_{r_2}^{(2)} \circ \cdots \circ \mathbf{b}_{r_N}^{(N)} \right) \\ &= \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)} \\ &= [\![\underline{\mathbf{G}}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N)}]\!]. \end{aligned} \tag{45}$$

where $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is the given data tensor, $\underline{\mathbf{G}} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$ is the core tensor, and $\mathbf{B}^{(n)} = [\mathbf{b}_1^{(n)}, \mathbf{b}_2^{(n)}, \ldots, \mathbf{b}_{R_n}^{(n)}] \in \mathbb{R}^{I_n \times R_n}$ are the mode-n factor (component) matrices, $n = 1, 2, \ldots, N$ (see Figure 35). The core tensor (typically, $R_n << I_n$) models a potentially complex pattern of mutual interaction between the vectors (components) in different modes. The model in (45) is often referred to as the Tucker-N model.

Using the properties of the Kronecker tensor product, the Tucker-N decomposition in (45) can be expressed in an equivalent matrix and vector form as:

$$\begin{aligned} \mathbf{X}_{(n)} &\cong \mathbf{B}^{(n)} \mathbf{G}_{(n)} (\mathbf{B}^{(1)} \otimes_L \cdots \otimes_L \mathbf{B}^{(n-1)} \otimes_L \mathbf{B}^{(n+1)} \otimes_L \cdots \otimes_L \mathbf{B}^{(N)})^T \\ &= \mathbf{B}^{(n)} \mathbf{G}_{(n)} (\mathbf{B}^{(N)} \otimes \cdots \otimes \mathbf{B}^{(n+1)} \otimes \mathbf{B}^{(n-1)} \otimes \cdots \otimes \mathbf{B}^{(1)})^T, \end{aligned} \tag{46}$$

$$\begin{aligned} \mathbf{X}_{<n>} &\cong (\mathbf{B}^{(1)} \otimes_L \cdots \otimes_L \mathbf{B}^{(n)}) \, \mathbf{G}_{<n>} (\mathbf{B}^{(n+1)} \otimes_L \cdots \otimes_L \mathbf{B}^{(N)})^T \\ &= (\mathbf{B}^{(n)} \otimes \cdots \otimes \mathbf{B}^{(1)}) \, \mathbf{G}_{<n>} (\mathbf{B}^{(N)} \otimes \mathbf{B}^{(N-1)} \otimes \cdots \otimes \mathbf{B}^{(n+1)})^T, \end{aligned} \tag{47}$$

$$\begin{aligned} \text{vec}(\underline{\mathbf{X}}) &\cong [\mathbf{B}^{(1)} \otimes_L \mathbf{B}^{(2)} \otimes_L \cdots \otimes_L \mathbf{B}^{(N)}] \, \text{vec}(\underline{\mathbf{G}}) \\ &= [\mathbf{B}^{(N)} \otimes \mathbf{B}^{(N-1)} \otimes \cdots \otimes \mathbf{B}^{(1)}] \, \text{vec}(\underline{\mathbf{G}}), \end{aligned} \tag{48}$$

where the multi-indices are ordered in a reverse lexicographic order (little-endian).

The Tucker decomposition is said to be in an *independent Tucker format* if the all factor matrices, $\mathbf{B}^{(n)}$, are full column rank, while a Tucker format is termed an *orthonormal format*, or Higher Order SVD (HOSVD), if in addition, all the factor matrices, $\mathbf{B}^{(n)} = \mathbf{U}^{(n)}$, are orthogonal [73]. The standard Tucker model usually has orthogonal factor matrices (see Section 4.4).

Multilinear rank. The N-tuple $(R_1, R_2, \ldots, R_N)$ is called the multilinear-rank of an Nth-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, if the Tucker decomposition holds exactly and is defined as:

$$\text{rank}_{ML}(\underline{\mathbf{X}}) = \{\text{rank}(\mathbf{X}_{(1)}), \text{rank}(\mathbf{X}_{(2)}), \ldots, \text{rank}(\mathbf{X}_{(N)})\}, \tag{49}$$

with $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times I_1 \cdots I_{n-1} I_{n+1} \cdots I_N}$ for $n = 1, 2, \ldots, N$.

The independent Tucker format has the following important properties if the equality in (45) holds exactly (see e.g., [132] and references therein):

1. Tensor (CP) rank of any tensor, $\underline{\mathbf{X}} = [\![\underline{\mathbf{G}}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N)}]\!] \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, and the rank of its core tensor, $\underline{\mathbf{G}} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$, are the same, i.e.,

$$\text{rank}_{CP}(\mathbf{X}) = \text{rank}_{CP}(\mathbf{G}). \tag{50}$$

2. Border ranks[13] of the tensor $\underline{\mathbf{X}}$ and its core tensor $\underline{\mathbf{G}}$ are the same, i.e,

$$\text{rank}_{B}(\underline{\mathbf{X}}) = \text{rank}_{B}(\underline{\mathbf{G}}). \tag{51}$$

3. If a tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} = [\![\underline{\mathbf{G}}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N)}]\!]$, admits an independent Tucker format with multilinear rank $\{R_1, R_2, \ldots, R_N\}$, then[14]

$$R_n \leqslant \prod_{p \neq n}^{N} R_p \quad \forall n. \tag{52}$$

   Moreover, without loss of generality, under the assumption $R_1 \leqslant R_2 \leqslant \cdots \leqslant R_N$, we have

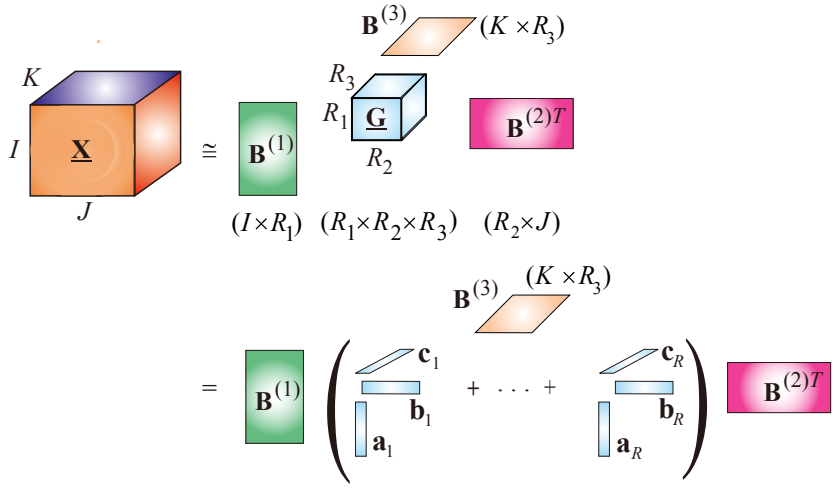
$$R_1 \leqslant \text{rank}_{CP}(\underline{\mathbf{X}}) \leqslant R_2 R_3 \cdots R_N. \tag{53}$$

4. If a data tensor is symmetric and has an independent Tucker format, $\underline{\mathbf{X}} = [\![\underline{\mathbf{G}}; \mathbf{B}, \mathbf{B}, \ldots, \mathbf{B}]\!] \in \mathbb{R}^{I \times I \times \cdots \times I}$, then its core tensor, $\underline{\mathbf{G}} \in \mathbb{R}^{R \times R \times \cdots \times R}$, is also symmetric, with $\text{rank}_{CP}(\underline{\mathbf{X}}) = \text{rank}_{CP}(\underline{\mathbf{G}})$.

[13] Border rank of a given tensor is the minimum number of rank-1 tensors which provides approximations with an arbitrary accuracy.

[14] This follows from the fact that mode-n matricizations, $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times I_1 \cdots I_{n-1} I_{n+1} \cdots \times I_N}$, are not independent of one another, which makes it impossible to chose the multilinear rank arbitrarily.

(a) Standard block diagrams of Tucker (top) and Tucker-CP (bottom) decompositions for a 3rd-order tensor



(b) The TN diagram for the Tucker and Tucker/CP decompositions of a 4th-order tensor

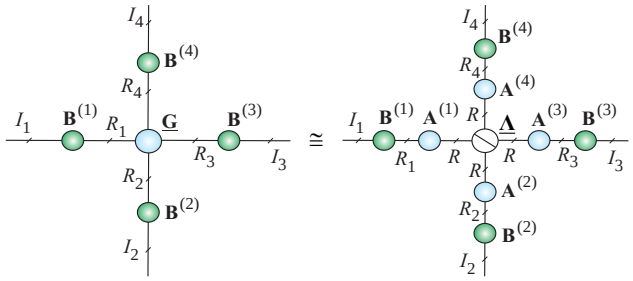


Figure 35: Illustration of the Tucker and Tucker-CP decompositions. The objective is to compute factor matrices, $\mathbf{B}^{(n)}$, and the core tensor, $\underline{\mathbf{G}}$. (a) Tucker decomposition of a 3rd-order tensor $\underline{\mathbf{X}} \cong \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \times_3 \mathbf{B}^{(3)}$. In some applications, the core tensor is approximately factorized using the CP decomposition as $\underline{\mathbf{G}} \cong \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$, or alternatively using TT/HT decompositions. (b) Graphical representation of the Tucker and CP decompositions in a two-stage procedure, for a 4th-order tensor $\underline{\mathbf{X}} \cong \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \times_3 \mathbf{B}^{(3)} \times_4 \mathbf{B}^{(4)} = [\![\underline{\mathbf{G}}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \mathbf{B}^{(3)}, \mathbf{B}^{(4)}]\!] \cong (\underline{\mathbf{\Lambda}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \mathbf{A}^{(3)} \times_4 \mathbf{A}^{(4)}) \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \times_3 \mathbf{B}^{(3)} \times_4 \mathbf{B}^{(4)} = [\![\underline{\mathbf{\Lambda}}; \mathbf{B}^{(1)}\mathbf{A}^{(1)}, \mathbf{B}^{(2)}\mathbf{A}^{(2)}, \mathbf{B}^{(3)}\mathbf{A}^{(3)}, \mathbf{B}^{(4)}\mathbf{A}^{(4)}]\!]$.

5. For the orthonormal Tucker format, that is, the HOSVD, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} = [\![\underline{\mathbf{G}}; \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(N)}]\!]$, with $\mathbf{U}^{(n)\,T}\mathbf{U}^{(n)} = \mathbf{I}, \quad \forall n$, the Frobenius norms and the Schatten p-norms[15] of the data tensor, $\underline{\mathbf{X}}$, and its core tensor, $\underline{\mathbf{G}}$, are equal, i.e.,

$$||\underline{\mathbf{X}}||_F = ||\underline{\mathbf{G}}||_F, \tag{54}$$

$$||\underline{\mathbf{X}}||_{\mathcal{S}_p} = ||\underline{\mathbf{G}}||_{\mathcal{S}_p} \quad 1 \leqslant p < \infty, \tag{55}$$

Thus the computation of the Frobenius norms can be performed with an $\mathcal{O}(R^N)$ complexity ($R = \max\{R_1, \ldots, R_N\}$), instead of the usual order $\mathcal{O}(I^N)$ complexity (typically $R \ll I$).

Note that the CP decomposition can be considered as a special case of the Tucker decomposition, whereby the cube core tensor has nonzero elements only on the main diagonal (see Figure 33). In contrast to the CP decomposition, the unconstrained Tucker decomposition is not unique. However, constraints imposed on all factor matrices and/or core tensor can reduce the indeterminacies inherent in CA to only column-wise permutation and scaling, thus yielding a unique core tensor and factor matrices [297].

The Tucker-N model, in which $(N-K)$ factor matrices are identity matrices is called the Tucker-(K, N) model. In the simplest scenario, for a 3rd-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$, the Tucker-(2,3) or simply Tucker-2 model, can be described as[16]

$$\underline{\mathbf{X}} \cong \underline{\mathbf{G}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{I} = \underline{\mathbf{G}} \times_1 \mathbf{A} \times_2 \mathbf{B}, \tag{56}$$

or in an equivalent matrix form

$$\mathbf{X}_k = \mathbf{A}\mathbf{G}_k\mathbf{B}^T, \qquad (k = 1, 2, \ldots, K) \tag{57}$$

where $\mathbf{X}_k = \underline{\mathbf{X}}(:, :, k) \in \mathbb{R}^{I \times J}$ and $\mathbf{G}_k = \underline{\mathbf{G}}(:, :, k) \in \mathbb{R}^{R_1 \times R_2}$ are respectively the frontal slices of the data tensor $\underline{\mathbf{X}}$ and the core tensor $\underline{\mathbf{G}} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$, and $\mathbf{A} \in \mathbb{R}^{I \times R_1}$, $\mathbf{B} \in \mathbb{R}^{J \times R_2}$.

Similarly, the PARALIND/CONFAC-(K, N) models[17] can be defined as

$$\begin{aligned} \underline{\mathbf{X}} &\cong \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \times_3 \cdots \times_N \mathbf{B}^{(N)} \\ &= [\![\underline{\mathbf{I}}; \mathbf{B}^{(1)}\mathbf{\Phi}^{(1)}, \ldots, \mathbf{B}^{(K)}\mathbf{\Phi}^{(K)}, \mathbf{B}^{(K+1)}, \ldots, \mathbf{B}^{(N)}]\!], \end{aligned} \tag{58}$$

[15] The Schatten p-norm of an Nth-order tensor $\underline{\mathbf{X}}$ is defined as the average of the Schatten norms of mode-n unfolding, i.e., $||\underline{\mathbf{X}}||_{\mathcal{S}_p} = (1/N)\sum_{n=1}^{N} ||\mathbf{X}_{(n)}||_{\mathcal{S}_p}$ and $||\mathbf{X}||_{\mathcal{S}_p} = (\sum_r \sigma_r^p)^{1/p}$, where σ_r is r-th singular value of the matrix $\mathbf{X}$. For $p = 1$, the Schatten norm of a matrix $\mathbf{X}$ is called the nuclear norm or the trace norm, while for $p = 0$ the Schatten norm is the rank of $\mathbf{X}$, which can be replaced by the surrogate function $\log\det(\mathbf{X}\mathbf{X}^T + \varepsilon\mathbf{I})$, $\varepsilon > 0$.

[16] The Tucker-2 model is equivalent to the TT model for a 3rd-order tensor. The case where the factor matrices and the core tensor are non-negative is referred to as the NTD-2 (Nonnegative Tucker-2 decomposition).

[17] PARALIND is an abbreviation of PARAllel with LINear Dependencies, while CONFAC refers to CONstrained FACtor model; for more details see [69, 97] and references therein.

Table 5: Different forms of CP and Tucker representations of a third-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$.

CP Decomposition	Tucker Decomposition
Scalar representation	
$x_{ijk} = \sum_{r=1}^{R} \lambda_r \, a_{ir} \, b_{jr} \, c_{kr}$	$x_{ijk} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} g_{r_1 r_2 r_3} \, a_{i r_1} \, b_{j r_2} \, c_{k r_3}$
Tensor representation, outer products	
$\underline{\mathbf{X}} = \sum_{r=1}^{R} \lambda_r \, \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$	$\underline{\mathbf{X}} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} g_{r_1 r_2 r_3} \, \mathbf{a}_{r_1} \circ \mathbf{b}_{r_2} \circ \mathbf{c}_{r_3}$
Tensor representation, multilinear products	
$\underline{\mathbf{X}} = \underline{\mathbf{\Lambda}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}$ $= [\![\underline{\mathbf{\Lambda}}; \, \mathbf{A}, \, \mathbf{B}, \, \mathbf{C}]\!]$	$\underline{\mathbf{X}} = \underline{\mathbf{G}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}$ $= [\![\underline{\mathbf{G}}; \, \mathbf{A}, \, \mathbf{B}, \, \mathbf{C}]\!]$
Matrix representations	
$\mathbf{X}_{(1)} = \mathbf{A} \, \mathbf{\Lambda} \, (\mathbf{B} \odot_L \mathbf{C})^T$ $\mathbf{X}_{(2)} = \mathbf{B} \, \mathbf{\Lambda} \, (\mathbf{A} \odot_L \mathbf{C})^T$ $\mathbf{X}_{(3)} = \mathbf{C} \, \mathbf{\Lambda} \, (\mathbf{A} \odot_L \mathbf{B})^T$	$\mathbf{X}_{(1)} = \mathbf{A} \, \mathbf{G}_{(1)} \, (\mathbf{B} \otimes_L \mathbf{C})^T$ $\mathbf{X}_{(2)} = \mathbf{B} \, \mathbf{G}_{(2)} \, (\mathbf{A} \otimes_L \mathbf{C})^T$ $\mathbf{X}_{(3)} = \mathbf{C} \, \mathbf{G}_{(3)} \, (\mathbf{A} \otimes_L \mathbf{B})^T$
Vector representation	
$\text{vec}(\underline{\mathbf{X}}) = (\mathbf{A} \odot_L \mathbf{B} \odot_L \mathbf{C}) \boldsymbol{\lambda}$	$\text{vec}(\underline{\mathbf{X}}) = (\mathbf{A} \otimes_L \mathbf{B} \otimes_L \mathbf{C}) \, \text{vec}(\underline{\mathbf{G}})$
Matrix slices $\mathbf{X}_k = \mathbf{X}(:, :, k)$	
$\mathbf{X}_k = \mathbf{A} \, \text{diag}(\lambda_1 \, c_{k,1}, \ldots, \lambda_R \, c_{k,R}) \, \mathbf{B}^T$	$\mathbf{X}_k = \mathbf{A} \left(\sum_{r_3=1}^{R_3} c_{k r_3} \mathbf{G}(:, :, r_3) \right) \mathbf{B}^T$

Table 6: Different forms of CP and Tucker representations of of an N-th order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_N]^T$, $\boldsymbol{\Lambda} = \text{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_N\}$

CP	Tucker
Scalar product	
$x_{i_1,\ldots,i_N} = \sum_{r=1}^{R} \lambda_r \, b_{i_1,r}^{(1)} \cdots b_{i_N,r}^{(N)}$	$x_{i_1,\ldots,i_N} = \sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} g_{r_1,\ldots,r_N} \, b_{i_1,r_1}^{(1)} \cdots b_{i_N,r_N}^{(N)}$
Outer product	
$\underline{\mathbf{X}} = \sum_{r=1}^{R} \lambda_r \, \mathbf{b}_r^{(1)} \circ \cdots \circ \mathbf{b}_r^{(N)}$	$\underline{\mathbf{X}} = \sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} g_{r_1,\ldots,r_N} \, \mathbf{b}_{r_1}^{(1)} \circ \cdots \circ \mathbf{b}_{r_N}^{(N)}$
Multilinear product	
$\underline{\mathbf{X}} = \underline{\boldsymbol{\Lambda}} \times_1 \mathbf{B}^{(1)} \cdots \times_N \mathbf{B}^{(N)}$	$\underline{\mathbf{X}} = \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \cdots \times_N \mathbf{B}^{(N)}$
$\underline{\mathbf{X}} = \left[\!\left[\underline{\boldsymbol{\Lambda}}; \mathbf{B}^{(1)}, \ldots, \mathbf{B}^{(N)} \right]\!\right]$	$\underline{\mathbf{X}} = \left[\!\left[\underline{\mathbf{G}}; \mathbf{B}^{(1)}, \ldots, \mathbf{B}^{(N)} \right]\!\right]$
Vectorization	
$\text{vec}(\underline{\mathbf{X}}) = \left(\bigodot_{n=N}^{1} \mathbf{B}^{(n)} \right) \boldsymbol{\lambda}$	$\text{vec}(\underline{\mathbf{X}}) = \left(\bigotimes_{n=N}^{1} \mathbf{B}^{(n)} \right) \text{vec}(\underline{\mathbf{G}})$
Matricization	
$\mathbf{X}_{(n)} = \mathbf{B}^{(n)} \boldsymbol{\Lambda} \left(\bigodot_{m=N,\, m\neq n}^{1} \mathbf{B}^{(m)} \right)^{\mathrm{T}}$	$\mathbf{X}_{(n)} = \mathbf{B}^{(n)} \mathbf{G}_{(n)} \left(\bigotimes_{m=N,\, m\neq n}^{1} \mathbf{B}^{(m)} \right)^{\mathrm{T}}$
$\mathbf{X}_{<n>} = (\bigodot_{m=n}^{1} \mathbf{B}^{(m)}) \boldsymbol{\Lambda} (\bigodot_{m=N}^{n+1} \mathbf{B}^{(m)})^{\mathrm{T}}$,	$\mathbf{X}_{<n>} = (\bigotimes_{m=n}^{1} \mathbf{B}^{(m)}) \mathbf{G}_{<n>} (\bigotimes_{m=N}^{n+1} \mathbf{B}^{(m)})^{\mathrm{T}}$
Slice representation	
$\underline{\mathbf{X}}(:,:,k_3) = \mathbf{B}^{(1)} \widetilde{\mathbf{D}}_{k_3} \mathbf{B}^{(2)\,T}$	$\underline{\mathbf{X}}(:,:,k_3) = \mathbf{B}^{(1)} \widetilde{\mathbf{G}}_{k_3} \mathbf{B}^{(2)\,T}$, $k_3 = \overline{i_3 i_4 \cdots i_N}$
$\widetilde{\mathbf{D}}_{k_3} = \text{diag}(\tilde{d}_{11}, \ldots, \tilde{d}_{RR}) \in \mathbb{R}^{R \times R}$ with entries $\tilde{d}_{rr} = \lambda_r b^{(3)}(i_3, r) \cdots b^{(N)}(i_N, r)$	
$\widetilde{\mathbf{G}}_{k_3} = \sum_{r_3} \cdots \sum_{r_N} b_{i_3,r_3}^{(3)} \cdots b_{i_N,r_N}^{(N)} \mathbf{G}_{:,:,r_3,\ldots,r_N}$ is the sum of frontal slices.	

where the core tensor, also called the constrained tensor or interaction tensor, is expressed as

$$\mathbf{G} = \underline{\mathbf{I}} \times_1 \mathbf{\Phi}^{(1)} \times_2 \mathbf{\Phi}^{(2)} \times_3 \cdots \times_K \mathbf{\Phi}^{(K)}, \tag{59}$$

with $K \leqslant N$. The factor matrices, $\mathbf{\Phi}^{(n)} \in \mathbb{R}^{R_n \times R}$ with $R \geqslant \max(R_n)$, are constrained and are often called the interaction matrices.

Another useful constrained tensor decomposition model, which can be represented graphically as a nested Tucker-(K, N) model, is the PARATUCK-(K, N) model (see, e.g., the review paper [97] and references therein). The PARATUCK-(K, N), like the PARALIND/CONFAC-(K, N) models, performs a decomposition in which the core tensor is defined through the interaction matrices, however, the PARATUCK-(K, N) models provide constraints between the factor matrices whereas the PARALIND/CONFAC-(K, N) describes the constraints of each factor matrix independently.

Generalized Tucker format and its links to TTNS model. For high-order tensors, $\underline{\mathbf{X}} \in \mathbb{R}^{I_{1,1} \times \cdots \times I_{1,K_1} \times I_{2,1} \times \cdots \times I_{N,K_N}}$, the Tucker-$N$ format can be naturally generalized by replacing the factor matrices, $\mathbf{B}^{(n)} \in \mathbb{R}^{I_n \times R_n}$, by higher-order tensors $\underline{\mathbf{B}}^{(n)} \in \mathbb{R}^{I_{n,1} \times I_{n,2} \times \cdots \times I_{n,K_n} \times R_n}$, to give

$$\underline{\mathbf{X}} \cong [\![\underline{\mathbf{G}}; \underline{\mathbf{B}}^{(1)}, \underline{\mathbf{B}}^{(2)}, \ldots, \underline{\mathbf{B}}^{(N)}]\!], \tag{60}$$

where the entries of the data tensor are computed as

$$\underline{\mathbf{X}}(\mathbf{i}_1, \ldots, \mathbf{i}_N) = \sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} \underline{\mathbf{G}}(r_1, \ldots, r_N) \underline{\mathbf{B}}^{(1)}(\mathbf{i}_1, r_1) \cdots \mathbf{B}^{(N)}(\mathbf{i}_N, r_N),$$

and $\mathbf{i}_n = (i_{n,1} i_{n,2} \ldots i_{n,K_n})$ [169].

Furthermore, the nested (hierarchical) form of such a generalized Tucker decomposition leads to the Tensor Tree Networks States (TTNS) model (see Figure 21 and Figure 23), possibly with a variable order of cores, which can be formulated as

$$\begin{aligned}
\underline{\mathbf{X}} &= [\![\underline{\mathbf{G}}_1;\ \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N_1)}]\!] \\
\underline{\mathbf{G}}_1 &= [\![\underline{\mathbf{G}}_2;\ \underline{\mathbf{A}}^{(1,2)}, \underline{\mathbf{A}}^{(2,2)}, \ldots, \underline{\mathbf{A}}^{(N_2,2)}]\!]. \\
&\ldots \\
\underline{\mathbf{G}}_P &= [\![\underline{\mathbf{G}}_{P+1};\ \underline{\mathbf{A}}^{(1,P+1)}, \underline{\mathbf{A}}^{(2,P+1)}, \ldots, \underline{\mathbf{A}}^{(N_{P+1},P+1)}]\!],
\end{aligned} \tag{61}$$

where $\underline{\mathbf{G}}_p \in \mathbb{R}^{R_1^{(p)} \times R_2^{(p)} \times \cdots \times R_{N_p}^{(p)}}$ and $\underline{\mathbf{A}}^{(n_p,p)} \in \mathbb{R}^{R_1^{(p-1)} \times \cdots \times R_{N_p}^{(p-1)} \times R_{n_p}^{(p)}}$ with $p = 2, \ldots, P+1$.

Note that some factor tensors, $\underline{\mathbf{B}}^{(n)}$ and/or $\underline{\mathbf{A}}^{(n_p,p)}$, can be identity tensors which yields an irregular structure, potentially with a variable order of tensors.

This follows from the simple observation that a mode-n product may have, e.g., the following form:

$$\underline{\mathbf{X}} \times_n \mathbf{B}^{(n)} = [\![\underline{\mathbf{X}}; \mathbf{I}_1, \ldots, \mathbf{I}_{I_{n-1}}, \mathbf{B}^{(n)}, \mathbf{I}_{I_{n+1}}, \ldots, \mathbf{I}_{I_N}]\!].$$

The efficiency of this representation strongly relies on an appropriate choice of the tree structure. We usually assume that the tree structure of TTNS is given or assumed *a priory*, however, recent attempts aim to find an optimal tree structure from a subset of tensor entries and without any *a priori* knowledge on the tree structure. This is achieved using so called rank-adaptive cross–approximation techniques which approximate a tensor by hierarchical tree formats [15, 16].

Operations in the Tucker format. Consider the Nth-order tensors $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$, represented in the Tucker format as

$$\underline{\mathbf{X}} = [\![\underline{\mathbf{G}}_X; \mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(N)}]\!] \quad \text{and} \quad \underline{\mathbf{Y}} = [\![\underline{\mathbf{G}}_Y; \mathbf{Y}^{(1)}, \ldots, \mathbf{Y}^{(N)}]\!], \tag{62}$$

with with multilinear rank $\{R_1, R_2, \ldots, R_N\}$ and $\{Q_1, Q_2, \ldots, Q_N\}$, respectively.

Many mathematical operations can then be performed directly in the Tucker format, which may reduce computational costs significantly as follows [169, 224]:

- **The addition** of two Tucker tensors of the same order and sizes [169]

$$\underline{\mathbf{X}} + \underline{\mathbf{Y}} = [\![\underline{\mathbf{G}}_X \oplus \underline{\mathbf{G}}_Y;\ [\mathbf{X}^{(1)}, \mathbf{Y}^{(1)}], \ldots, [\mathbf{X}^{(N)}, \mathbf{Y}^{(N)}]]\!], \tag{63}$$

  where $\oplus$ denotes a direct sum of two tensors, and $[\mathbf{X}^{(n)}, \mathbf{Y}^{(n)}] \in \mathbb{R}^{I_n \times (R_n + Q_n)}$, $\mathbf{X}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ and $\mathbf{Y}^{(n)} \in \mathbb{R}^{I_n \times Q_n}$, $\forall n$.

- **The Kronecker product** of two Tucker tensors of arbitrary orders and sizes [224]

$$\underline{\mathbf{X}} \otimes \underline{\mathbf{Y}} = [\![\underline{\mathbf{G}}_X \otimes \underline{\mathbf{G}}_Y;\ \mathbf{X}^{(1)} \otimes \mathbf{Y}^{(1)}, \ldots, \mathbf{X}^{(N)} \otimes \mathbf{Y}^{(N)}]\!]. \tag{64}$$

- **The Hadamard** or element-wise product of two Tucker tensors (of the same order and the same sizes)

$$\underline{\mathbf{X}} \circledast \underline{\mathbf{Y}} = [\![\underline{\mathbf{G}}_X \otimes \underline{\mathbf{G}}_Y;\ \mathbf{X}^{(1)} \odot_1 \mathbf{Y}^{(1)}, \ldots, \mathbf{X}^{(N)} \odot_1 \mathbf{Y}^{(N)}]\!], \tag{65}$$

  where $\odot_1$ denotes mode-1 Khatri-Rao product called sometimes the transposed Khatri-Rao product or row-wise Kronecker product.

- **The inner product** of two Tucker tensors of the same order and the same size can be reduced to the inner product of two smaller tensors, by exploiting the Kronecker product structure in the vectorized form, as fol-

lows

$$\begin{aligned}\langle \underline{\mathbf{X}}, \underline{\mathbf{Y}} \rangle &= \text{vec}(\underline{\mathbf{X}})^T \, \text{vec}(\underline{\mathbf{Y}}) \\ &= \text{vec}(\underline{\mathbf{G}}_X)^T \left(\bigotimes_{n=1}^{N} \mathbf{X}^{(n)\,T} \right) \left(\bigotimes_{n=1}^{N} \mathbf{Y}^{(n)} \right) \text{vec}(\underline{\mathbf{G}}_Y) \\ &= \text{vec}(\underline{\mathbf{G}}_X)^T \left(\bigotimes_{n=1}^{N} \mathbf{X}^{(n)\,T} \mathbf{Y}^{(n)} \right) \text{vec}(\underline{\mathbf{G}}_Y) \\ &= \langle [\![\underline{\mathbf{G}}_X; (\mathbf{X}^{(1)\,T} \mathbf{Y}^{(1)}), \ldots, (\mathbf{X}^{(N)\,T} \mathbf{Y}^{(N)})]\!], \underline{\mathbf{G}}_Y \rangle. \end{aligned} \tag{66}$$

- **The Frobenius norm** can be computed in a particularly simple way if the factor matrices are orthogonal, since then all products $\mathbf{X}^{(n)\,T}\mathbf{X}^{(n)}$, $\forall n$ become the identity matrices:

$$\begin{aligned} ||\mathbf{X}||_F &= \langle \underline{\mathbf{X}}, \underline{\mathbf{X}} \rangle \\ &= \text{vec}\left([\![\underline{\mathbf{G}}_X; (\mathbf{X}^{(1)\,T} \mathbf{X}^{(1)}), \ldots, (\mathbf{X}^{(N)\,T} \mathbf{X}^{(N)})]\!] \right)^T \text{vec}(\underline{\mathbf{G}}_X) \\ &= \text{vec}(\underline{\mathbf{G}}_X)^T \, \text{vec}(\underline{\mathbf{G}}_X) = ||\underline{\mathbf{G}}_X||_F. \end{aligned}$$

- **N-D discrete convolution** of two tensors: $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ with multilinear rank $\{R_1, R_2, \ldots, R_N\}$ and $\underline{\mathbf{Y}} \in \mathbb{R}^{J_1 \times \cdots \times J_N}$ with multilinear rank $\{Q_1, Q_2, \ldots, Q_N\}$ in the Tucker formats can be expressed as

$$\underline{\mathbf{Z}} = \underline{\mathbf{X}} * \underline{\mathbf{Y}} = [\![\underline{\mathbf{G}}_Z; \mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(N)}]\!] \tag{67}$$

$$\in \mathbb{R}^{(I_1+J_1-1) \times \cdots \times (I_N+J_N-1)}, \tag{68}$$

with the core tensor $\underline{\mathbf{G}}_Z = \underline{\mathbf{G}}_X \otimes \underline{\mathbf{G}}_Y \in \mathbb{R}^{R_1 Q_1 \times \cdots \times R_N Q_N}$ and factor matrices

$$\mathbf{Z}^{(n)} = \mathbf{X}^{(n)} \boxdot_1 \mathbf{Y}^{(n)} \in \mathbb{R}^{(I_n+J_n-1) \times R_n Q_n}, \tag{69}$$

where $\mathbf{Z}^{(n)}(:, s_n) = \mathbf{X}^{(n)}(:, r_n) * \mathbf{Y}^{(n)}(:, q_n) \in \mathbb{R}^{(I_n+J_n-1)}$ for $s_n = 1, 2, \ldots, R_n Q_n$.

- **Discrete Fourier Transform** (MATLAB functions fftn($\underline{\mathbf{X}}$) and fft($\mathbf{X}^{(n)}$, [], 1)) of tensor in the Tucker format

$$\mathcal{F}(\underline{\mathbf{X}}) = [\![\underline{\mathbf{G}}_X; \mathcal{F}(\mathbf{X}^{(1)}), \ldots, \mathcal{F}(\mathbf{X}^{(N)})]\!]. \tag{70}$$

4.4 Higher Order SVD (HOSVD) for Large-Scale Problems

The MultiLinear Singular Value Decomposition (MLSVD), also called the higher-order SVD (HOSVD), can be considered as a special form of the constrained Tucker decomposition [73, 74], in which all factor matrices, $\mathbf{B}^{(n)} = \mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times I_n}$, are orthogonal and the core tensor, $\underline{\mathbf{G}} = \underline{\mathbf{S}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, is all-orthogonal (see Figure 36).

The orthogonality properties of the core tensor are defined through the following conditions:

1. *All orthogonality.* The slices in each mode are mutually orthogonal, e.g., for a 3rd-order tensor

$$\langle \mathbf{S}_{:,k,:} \mathbf{S}_{:,l,:} \rangle = 0, \quad \text{for} \quad k \neq l, \tag{71}$$

2. *Pseudo-diagonality.* The Frobenius norms of slices in each mode are decreasing with the increase in the running index, e.g., for a 3rd-order tensor

$$||\mathbf{S}_{:,k,:}||_F \geqslant ||\mathbf{S}_{:,l,:}||_F, \quad k \geqslant l. \tag{72}$$

   These norms play a similar role as the singular values in standard matrix SVD [73, 74].

In practice, the orthogonal matrices $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times R_n}$, with $R_n \leqslant I_n$, can be computed by applying both the randomized and standard truncated SVD to the unfolded mode-n matrices, $\mathbf{X}_{(n)} = \mathbf{U}^{(n)} \mathbf{S}_n \mathbf{V}^{(n)T} \in \mathbb{R}^{I_n \times I_1 \cdots I_{n-1} I_{n+1} \cdots I_N}$. After obtaining the orthogonal matrices $\mathbf{U}^{(n)}$ of left singular vectors of $\mathbf{X}_{(n)}$, for each n, the core tensor $\underline{\mathbf{G}} = \underline{\mathbf{S}}$ can be computed as

$$\underline{\mathbf{S}} = \underline{\mathbf{X}} \times_1 \mathbf{U}^{(1)\,T} \times_2 \mathbf{U}^{(2)\,T} \cdots \times_N \mathbf{U}^{(N)\,T}, \tag{73}$$

so that

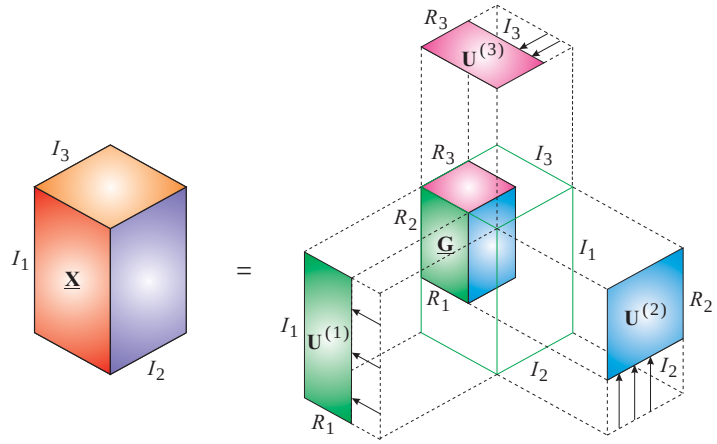
$$\underline{\mathbf{X}} = \underline{\mathbf{S}} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \cdots \times_N \mathbf{U}^{(N)}. \tag{74}$$

Due to the orthogonality of the core tensor $\underline{\mathbf{S}}$, its slices are mutually orthogonal.

Analogous to the standard truncated SVD, a large-scale data tensor, $\underline{\mathbf{X}}$, can be approximated by discarding the multilinear singular vectors and slices of the core tensor corresponding to small multilinear singular values. Figure 36 and Algorithm 3 outline the truncated HOSVD, for which any optimized matrix SVD procedure can be applied.

For large-scale tensors, the unfolding matrices, $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times I_{\bar{n}}}$, may become prohibitively large (with $I_{\bar{n}} \gg I_n$), easily exceeding the memory of standard computers. Using a direct and naive approach, the truncated SVD of an unfolding matrix, $\mathbf{X}_{(n)} = \mathbf{U}^{(n)} \mathbf{\Sigma}_n \mathbf{V}^{(n)T}$, can be easily partitioned into Q slices, as $\mathbf{X}_{(n)} = [\mathbf{X}_{1,n}, \mathbf{X}_{2,n}, \ldots, \mathbf{X}_{Q,n}] = \mathbf{U}^{(n)} \mathbf{\Sigma}_n [\mathbf{V}_{1,n}^T, \mathbf{V}_{2,n}^T, \ldots, \mathbf{V}_{Q,n}^T]$. Next, the orthogonal matrices $\mathbf{U}^{(n)}$ and the diagonal matrices $\mathbf{\Sigma}_n$ can be obtained from the eigenvalue decompositions $\mathbf{X}_{(n)} \mathbf{X}_{(n)}^T = \mathbf{U}^{(n)} \mathbf{\Sigma}_n^2 \mathbf{U}^{(n)T} = \sum_q \mathbf{X}_{q,n} \mathbf{X}_{q,n}^T \in \mathbb{R}^{I_n \times I_n}$, allowing for the terms $\mathbf{V}_{q,n} = \mathbf{X}_{q,n}^T \mathbf{U}^{(n)} \mathbf{\Sigma}_n^{-1}$ to be computed separately. This enables us to optimize the size of the q-th slice $\mathbf{X}_{q,n} \in \mathbb{R}^{I_n \times (I_{\bar{n}}/Q)}$ so as to match the available computer memory. Such a simple approach to compute matrices $\mathbf{U}^{(n)}$ and/or $\mathbf{V}^{(n)}$ does not require loading the entire unfolding matrices at once into computer memory; instead the access to the datasets is sequential. For current standard sizes of computer memory, the dimension I_n is typically less than 10,000, while there is no limit on the dimension $I_{\bar{n}} = \prod_{k \neq n} I_k$.

(a)



(b)

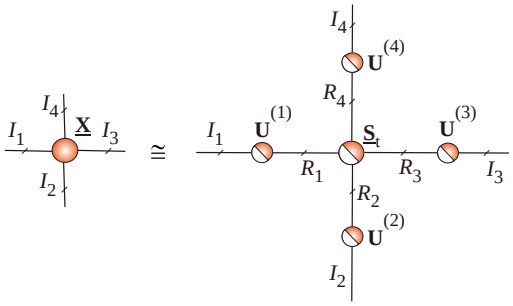


Figure 36: Graphical illustration of the HOSVD. (a) The exact HOSVD and truncated (approximative) HOSVD for a 3rd-order tensor calculated using SVD as: $\underline{\mathbf{X}} \cong \underline{\mathbf{G}} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)}$. (b) Tensor network notation for the HOSVD for a 4th-order tensor $\underline{\mathbf{X}} \cong \underline{\mathbf{S}} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)} \times_4 \mathbf{U}^{(4)}$. All the factor matrices, $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times R_n}$, and the core tensor, $\underline{\mathbf{S}} = \underline{\mathbf{G}} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$, are orthogonal.

For very large-scale and low-rank matrices, instead of the standard truncated SVD approach, we can alternatively apply the randomized SVD algorithm which reduces the original data matrix to a relatively small matrix by multiplying it with a random sampling matrix $\boldsymbol{\Omega}$ (see Algorithm 2) [115]. Using special random sampling matrices, for instance, a sub-sampled random Fourier transform, substantial gain in the execution time can be achieved, together with the asymptotic complexity of $\mathcal{O}(IJ\log(R)$. Unfortunately, this approach is typically not accurate enough for matrices for which the singular values decay slowly [115].

Algorithm 2: **Random SVD (rSVD) for large-scale and low-rank matrices [115]**

Input: A matrix $\mathbf{X} \in \mathbb{R}^{I \times J}$, estimated rank R, and oversampling parameter P or overestimated rank $\widetilde{R} = R + P$

Output: An approximate rank-$\widetilde{R}$ SVD, $\mathbf{X} \cong \mathbf{U}\mathbf{S}\mathbf{V}^T$, i.e., orthogonal matrices $\mathbf{U} \in \mathbb{R}^{I \times \widetilde{R}}$, $\mathbf{V} \in \mathbb{R}^{J \times \widetilde{R}}$, and diagonal matrix $\mathbf{S} \in \mathbb{R}^{\widetilde{R} \times \widetilde{R}}$ with singular values

1: Draw a random Gaussian matrix $\mathbf{\Omega} \in \mathbb{R}^{J \times \widetilde{R}}$,
2: Form the sample matrix $\mathbf{Y} = \mathbf{A}\mathbf{\Omega} \in \mathbb{R}^{I \times \widetilde{R}}$
3: Compute a QR decomposition $\mathbf{Y} = \mathbf{Q}\mathbf{R}$
4: Form the matrix $\mathbf{A} = \mathbf{Q}^T\mathbf{X} \in \mathbb{R}^{\widetilde{R} \times J}$
5: Compute the SVD of the small matrix $\mathbf{A}$ as $\mathbf{A} = \widehat{\mathbf{U}}\mathbf{S}\mathbf{V}^T$
6: Form the matrix $\mathbf{U} = \mathbf{Q}\widehat{\mathbf{U}}$.

The truncated HOSVD can be optimized and implemented in several alternative ways. For example, if $R_n \ll I_n$, the truncated tensor $\underline{\mathbf{Z}} \leftarrow \underline{\mathbf{X}} \times_1 \mathbf{U}^{(1)T}$ yields a smaller unfolding matrix $\mathbf{Z}_{(2)} \in \mathbb{R}^{I_2 \times R_1 I_3 \cdots I_N}$, so the multiplication $\mathbf{Z}_{(2)}\mathbf{Z}_{(2)}^T$ can be faster in the next iterations, as illustrated in Algorithm 4 [10, 265].

Furthermore, since the unfolding matrices $\mathbf{Y}_{(n)}^T$ are typically very "tall and skinny", a huge-scale truncated SVD and other constrained low-rank matrix factorizations can be computed efficiently based on the Hadoop / MapReduce paradigm [26, 60, 61].

Low multilinear rank approximation is always well-posed, however, in contrast to the standard truncated SVD for matrices, the truncated HOSVD does not yield the best multilinear rank approximation but satisfies the quasi-best approximation property [108]:

$$||\underline{\mathbf{X}} - [\![\underline{\mathbf{S}}; \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(N)}]\!]|| \leqslant \sqrt{N}||\underline{\mathbf{X}} - \underline{\mathbf{X}}_{\text{Best}}||, \tag{75}$$

where $\underline{\mathbf{X}}_{\text{Best}}$ is the best multilinear rank approximation of $\underline{\mathbf{X}}$, for a specific tensor norm $||\cdot||$.

When it comes to finding the best approximation, the ALS type algorithm called the Higher Order Orthogonal Iteration (HOOI) exhibits both the advantages and drawbacks of ALS algorithms for CP decomposition [74, 160]. For the HOOI algorithms see Algorithm 5 and Algorithm 6 [10, 301]. For more sophisticated algorithms for Tucker decompositions with orthogonality or nonnegativity constraints, suitable for large-scale data tensors, see [61, 131, 217, 299].

When a data tensor $\underline{\mathbf{X}}$ is very large and cannot be stored in computer memory, another challenge is to compute a core tensor $\underline{\mathbf{G}} = \underline{\mathbf{S}}$ directly, using the formula (73). The computation is therefore performed sequentially by fast matrix-by-matrix multiplications, as illustrated in Figure 37 (a) and (b) [250, 276][18].

[18] Efficient and parallel (state of the arts) algorithms for multiplications of such very large-scale

Table 7: Basic multiway component analysis (MWCA)/Low-Rank Tensor Approximations (LRTA) and related multiway dimensionality reduction models. The symbol $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots I_N}$ denotes the noisy data tensor, $\mathbf{Y} = \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)}$ is the general constrained Tucker model with the latent factor matrices $\mathbf{B}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ and the core tensor $\underline{\mathbf{G}} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$. In the special case of a CP decomposition, the core tensor is diagonal, $\underline{\mathbf{G}} = \underline{\mathbf{\Lambda}} \in \mathbb{R}^{R \times \cdots \times R}$, so that $\underline{\mathbf{Y}} = \sum_{r=1}^{R} \lambda_r (\mathbf{b}_r^{(1)} \odot_L \cdots \odot_L \mathbf{b}_r^{(N)})$.

Cost Function	Constraints
Multilinear (sparse) PCA (MPCA) $\max_{\mathbf{u}_r^{(n)}} \ \underline{\mathbf{X}} \bar{\times}_1 \mathbf{u}_r^{(1)} \bar{\times}_2 \mathbf{u}_r^{(2)} \cdots \bar{\times}_N \mathbf{u}_r^{(N)} + \gamma \sum_{n=1}^{N} \|\mathbf{u}_r^{(n)}\|_1$	$\mathbf{u}_r^{(n)\,T} \mathbf{u}_r^{(n)} = 1, \ \forall (n,r)$ $\mathbf{u}_r^{(n)\,T} \mathbf{u}_q^{(n)} = 0 \ \text{ for } \ r \neq q$
HOSVD/HOOI $\min_{\mathbf{U}^{(n)}} \ \|\underline{\mathbf{X}} - \underline{\mathbf{G}} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \cdots \times_N \mathbf{U}^{(N)}\|_F^2$	$\mathbf{U}^{(n)\,T} \mathbf{U}^{(n)} = \mathbf{I}_{R_n}, \ \forall n$
Multilinear ICA $\min_{\mathbf{B}^{(n)}} \ \|\underline{\mathbf{X}} - \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)}\|_F^2$	Vectors of $\mathbf{B}^{(n)}$ statistically as independent as possible
Nonnegative CP/Tucker Decomposition (NTF/NTD) [53] $\min_{\mathbf{B}^{(n)}} \ \|\underline{\mathbf{X}} - \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \cdots \times_N \mathbf{B}^{(N)}\|_F^2$ $+\gamma \sum_{n=1}^{N} \|\mathbf{B}^{(n)}\|_1$	Entries of $\underline{\mathbf{G}}$ and $\mathbf{B}^{(n)}, \ \forall n$ are nonnegative
Sparse CP/Tucker decomposition $\min_{\mathbf{B}^{(n)}} \ \|\underline{\mathbf{X}} - \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \cdots \times_N \mathbf{B}^{(N)}\|_F^2$ $+\gamma \sum_{n=1}^{N} \|\mathbf{B}^{(n)}\|_1$	Sparsity constraints imposed on $\mathbf{B}^{(n)}$
Smooth CP/Tucker (SmCP/SmTD) [286] $\min_{\mathbf{B}^{(n)}} \ \|\underline{\mathbf{X}} - \underline{\mathbf{\Lambda}} \times_1 \mathbf{B}^{(1)} \cdots \times_N \mathbf{B}^{(N)}\|_F^2$ $+\gamma \sum_{n=1}^{N} \sum_{r=1}^{R} \|\mathbf{L}\mathbf{b}_r^{(n)}\|_2$	Smoothness imposed on vectors $\mathbf{b}_r^{(n)}$ of $\mathbf{B}^{(n)} \in \mathbb{R}^{I_n \times R}, \quad \forall n$

Algorithm 3: **Sequentially Truncated HOSVD [265]**

Input: Nth-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and approximation accuracy ε

Output: HOSVD in the Tucker format $\hat{\underline{\mathbf{X}}} = [\![\underline{\mathbf{S}}; \mathbf{U}^{(1)}, \ldots, \underline{\mathbf{U}}^{(N)}]\!]$, such that $||\underline{\mathbf{X}} - \hat{\underline{\mathbf{X}}}||_F \leqslant \varepsilon$

1: $\underline{\mathbf{S}} \leftarrow \underline{\mathbf{X}}$
2: **for** $n = 1$ to N **do**
3: $\quad [\mathbf{U}^{(n)}, \mathbf{\Sigma}, \mathbf{V}] = \texttt{truncated\_svd}(\mathbf{S}_{(n)}, \frac{\varepsilon}{\sqrt{N}})$
4: $\quad \underline{\mathbf{S}} \leftarrow \mathbf{V}\mathbf{\Sigma}$
5: **end for**
6: $\underline{\mathbf{S}} \leftarrow \texttt{reshape}(\underline{\mathbf{S}}, [R_1, \ldots, R_N])$
7: **return** Core tensor $\underline{\mathbf{S}}$ and orthogonal factor matrices $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times R_n}$.

For very large-scale problems, it is useful to divide a data tensor $\underline{\mathbf{X}}$ into small blocks $\underline{\mathbf{X}}_{[k_1,k_2,\ldots,k_N]}$. In a similar way, we can divide the orthogonal factor matrices $\mathbf{U}^{(n)T}$ into corresponding blocks of matrices $\mathbf{U}^{(n)T}_{[k_n,p_n]}$, as illustrated in Figure 37 (c) for 3rd-order tensors [276]. For example, the blocks within the resulting tensor $\mathbf{G}^{(n)}$ can be computed sequentially or in parallel, as follows:

$$\underline{\mathbf{G}}^{(n)}_{[k_1,k_2,\ldots,q_n,\ldots,k_N]} = \sum_{k_n=1}^{K_n} \mathbf{X}_{[k_1,k_2,\ldots,k_n,\ldots,k_N]} \times_n \mathbf{U}^{(n)\,T}_{[k_n,q_n]}. \tag{76}$$

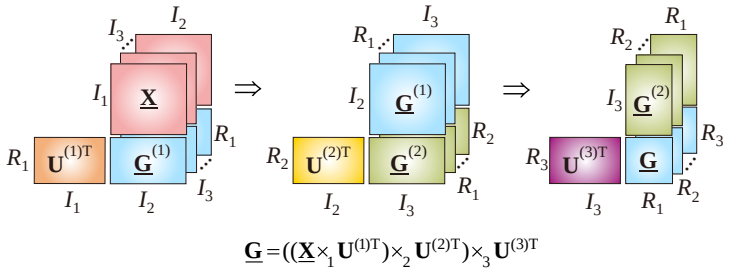
Applications. We have shown that the Tucker/HOSVD decomposition may be considered as a multilinear extension of PCA [160]; it therefore generalizes signal subspace techniques and finds application in areas including multilinear blind source separation, classification, feature extraction, and subspace-based harmonic retrieval [111, 177, 218, 266]. In this way, a low multilinear rank approximation achieved through Tucker decomposition may yield higher Signal-to-Noise Ratio (SNR) than the SNR for the original raw data tensor, which makes Tucker decomposition a very natural tool for signal compression and enhancement [59, 160, 239].

Recently, it was also proved that HOSVD can perform simultaneous subspace selection (data compression) and K-means clustering, widely used for unsupervised learning tasks [125, 212]. This is important, as by a combination of these methods, "relevant" data can be both identified and classified, thus not only revealing information but also simplifying feature extraction.

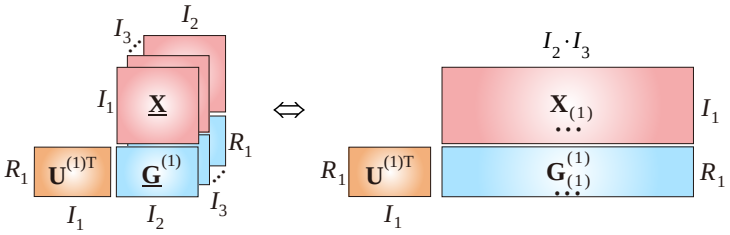
Anomaly detection using HOSVD. Anomaly detection refers to the discrimination of some specific patterns, signals, outliers or features that do not conform to certain expected behaviors, trends or properties [42, 96]. While such analysis can be performed in different domains, it is most frequently based on

matrices are available in [17, 171].

(a) Sequential computation



(b) Distributed computation



(c) Divide-and-conquer approach

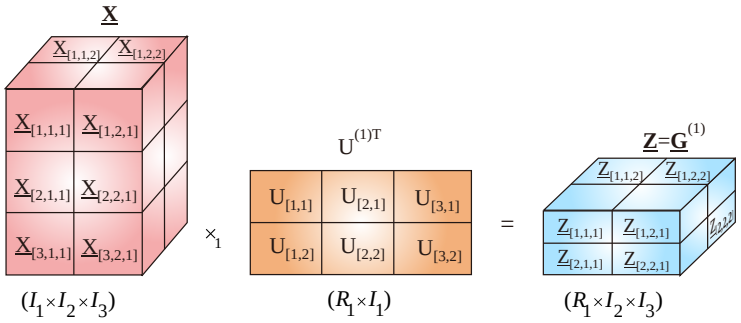


Figure 37: Computation of a multilinear (Tucker) product for large-scale HOSVD. (a) Standard sequential computing of multilinear products (TTM) $\underline{\mathbf{G}} = \underline{\mathbf{S}} = (((\underline{\mathbf{X}} \times_1 \mathbf{U}^{(1)T}) \times_2 \mathbf{U}^{(2)T}) \times_3 \mathbf{U}^{(3)T})$. (b) Distributed implementation through fast matrix-by-matrix multiplications. (c) An alternative method for large-scale problems using the "divide and conquer" approach, whereby a data tensor, $\underline{\mathbf{X}}$, and factor matrices, $\mathbf{U}^{(n)T}$, are partitioned into suitable small blocks: Subtensors $\underline{\mathbf{X}}_{[k_1,k_2,k_3]}$ and block matrices $\mathbf{U}^{(1)T}_{[k_1,p_1]}$. The blocks of a tensor, $\underline{\mathbf{Z}} = \underline{\mathbf{G}}^{(1)} = \underline{\mathbf{X}} \times_1 \mathbf{U}^{(1)T}$, are computed as $\underline{\mathbf{Z}}_{[q_1,k_2,k_3]} = \sum_{k_1=1}^{K_1} \mathbf{X}_{[k_1,k_2,k_3]} \times_1 \mathbf{U}^{(1)T}_{[k_1,q_1]}$ (see Eq. (76) for a general case).

Algorithm 4: **Truncated HOSVD using sequentially symmetric EVD [10, 73]**

Input: Nth-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and approximation accuracy ε

Output: Approximative representation of a tensor in Tucker format with orthogonal factor matrices $\mathbf{U}^{(n)}$, such that $\hat{\underline{\mathbf{X}}} = [\![\underline{\mathbf{S}}; \mathbf{U}^{(1)}, \ldots, \underline{\mathbf{U}}^{(N)}]\!]$ and $||\underline{\mathbf{X}} - \hat{\underline{\mathbf{X}}}||_F \leqslant \varepsilon$

1: $\underline{\mathbf{Z}} \leftarrow \underline{\mathbf{X}}$
2: **for** $n = 1$ to N **do**
3: $\quad \mathbf{C} \leftarrow \mathbf{Z}_{(n)} \mathbf{Z}_{(n)}^T \in \mathbb{R}^{R_n \times R_n}$
4: $\quad$ Perform EVD: $\mathbf{C} = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^T$, with non-increasing eigenvalues $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_{R_n}$
5: $\quad$ Estimate $R_n \leftarrow \min \tilde{R}$ such that $\sum_{r>\tilde{R}} \lambda_r(\mathbf{C}) \leqslant \varepsilon^2 / N$
6: $\quad \mathbf{U}^{(n)} \leftarrow$ leading R_n eigenvectors of $\mathbf{C}$
7: $\quad \underline{\mathbf{Z}} \leftarrow \underline{\mathbf{Z}} \times_n \mathbf{U}^{(n)\,T}$
8: **end for**
9: Construct the all-orthogonal core tensor as $\underline{\mathbf{S}} \leftarrow \underline{\mathbf{Z}}$
10: **return** All orthogonal core tensor $\underline{\mathbf{S}} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$ and orthogonal factor matrices $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times R_n}$.

Algorithm 5: **Higher Order Orthogonal Iteration (HOOI) [10, 74]**

Input: Nth-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ (usually in Tucker/HOSVD format)

Output: Improved Tucker approximation using ALS approach, with orthogonal factor matrices $\mathbf{U}^{(n)}$

1: Initialization via the standard HOSVD (see Algorithm 3)
2: **repeat**
3: $\quad$ **for** $n = 1$ to N **do**
4: $\quad\quad \underline{\mathbf{Z}} \leftarrow \underline{\mathbf{X}} \times_{p \neq n} \{\mathbf{U}^{(p)\,T}\}$
5: $\quad\quad \mathbf{C} \leftarrow \mathbf{Z}_{(n)} \mathbf{Z}_{(n)}^T \in \mathbb{R}^{R \times R}$
6: $\quad\quad \mathbf{U}^{(n)} \leftarrow$ leading R_n eigenvectors of $\mathbf{C}$
7: $\quad$ **end for**
8: $\quad \underline{\mathbf{G}} \leftarrow \underline{\mathbf{Z}} \times_N \mathbf{U}^{(N)\,T}$
9: **until** the cost function $(||\underline{\mathbf{X}}||_F^2 - ||\underline{\mathbf{G}}||_F^2)$ ceases to decrease
10: **return** $[\![\underline{\mathbf{G}}; \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(N)}]\!]$

spectral methods such as PCA, whereby high dimensional data are projected onto a lower-dimensional subspace in which the anomalies may be identified more easier [42]. The main assumption within such approaches is that the normal and abnormal patterns, which may be difficult to distinguish in the original space, appear significantly different in the projected subspace. When con-

Algorithm 6: **HOOI using randomization for large-scale data [301]**

Input: Nth-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and multilinear rank $\{R_1, R_2, \ldots, R_N\}$

Output: Approximative representation of a tensor in Tucker format, with orthogonal factor matrices $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times R_n}$

Initialize factor matrices $\mathbf{U}^{(n)}$ as random Gaussian matrices
Repeat steps (2)-(6) only two times:
for $n = 1$ to N **do**
  $\underline{\mathbf{Z}} = \underline{\mathbf{X}} \times_{p \neq n} \{\mathbf{U}^{(n)\,T}\}$
  Compute $\tilde{\mathbf{Z}}^{(n)} = \mathbf{Z}_{(n)} \boldsymbol{\Omega}^{(n)} \in \mathbb{R}^{I_n \times R_n}$, where $\boldsymbol{\Omega}^{(n)} \in \mathbb{R}^{\prod_{p \neq n} R_p \times R_n}$ is a random matrix drawn from Gaussian distribution
  Compute $\mathbf{U}^{(n)}$ as an orthonormal basis of $\tilde{\mathbf{Z}}^{(n)}$, e.g., by using QR decomposition
end for
Construct the core tensor as $\underline{\mathbf{G}} = \underline{\mathbf{X}} \times_1 \mathbf{U}^{(1)\,T} \times_2 \mathbf{U}^{(2)\,T} \cdots \times_N \mathbf{U}^{(N)\,T}$
return $\underline{\mathbf{X}} \cong [\![\underline{\mathbf{G}}; \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \underline{\mathbf{U}}^{(N)}]\!]$

Algorithm 7: **Tucker decomposition with constrained factor matrices via 2-way CA /LRMF**

Input: Nth-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, multilineal rank $\{R_1, \ldots, R_N\}$ and desired constraints imposed on factor matrices $\mathbf{B}^{(n)} \in \mathbb{R}^{I_n \times R_n}$

Output: Tucker decomposition with constrained factor matrices $\mathbf{B}^{(n)}$ using LRMF and a simple unfolding approach

Initialize randomly or via standard HOSVD (see Algorithm 3)
for $n = 1$ to N **do**
  Compute specific LRMF or 2-way CA (e.g., RPCA, ICA, NMF) of unfolding $\mathbf{X}_{(n)}^T \cong \mathbf{A}^{(n)} \mathbf{B}^{(n)\,T}$ or $\mathbf{X}_{(n)} \cong \mathbf{B}^{(n)} \mathbf{A}^{(n)\,T}$
end for
Compute core tensor $\underline{\mathbf{G}} = \underline{\mathbf{X}} \times_1 [\mathbf{B}^{(1)}]^\dagger \times_2 [\mathbf{B}^{(2)}]^\dagger \cdots \times_N [\mathbf{B}^{(N)}]^\dagger$
return Constrained Tucker decomposition $\underline{\mathbf{X}} \cong [\![\underline{\mathbf{G}}, \mathbf{B}^{(1)}, \ldots, \underline{\mathbf{B}}^{(N)}]\!]$

sidering very large datasets, since the basic Tucker decomposition model generalize PCA and SVD, it offers a natural framework for anomaly detection via HOSVD, as illustrated Figure 38. To handle the exceedingly large dimensionality, we may first compute tensor decompositions for sampled (pre-selected) small blocks of the original large-scale 3rd order tensor, followed by the analysis of changes in specific factor matrices $\mathbf{U}^{(n)}$. A simpler form is straightforwardly obtained by fixing the core tensor and some factor matrices while monitoring the changes along one or more specific modes, as the block tensor moves from left to right as shown in Figure 38.

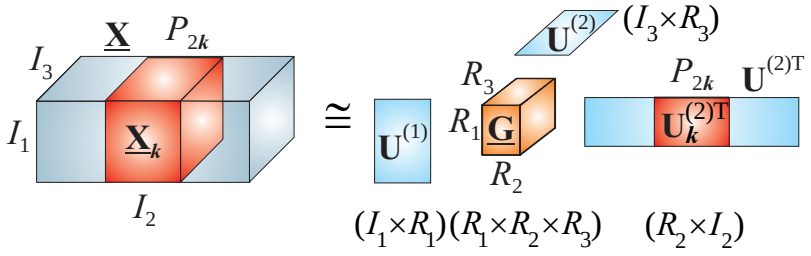


Figure 38: Conceptual model for performing the HOSVD for a very large-scale 3rd-order data tensor. This is achieved by dividing the tensor into blocks $\underline{\mathbf{X}}_k \cong \underline{\mathbf{G}} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}_k^{(2)} \times_3 \mathbf{U}^{(3)}$, $(k = 1, 2 \ldots, K)$. It assumed that the data tensor $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is sampled by sliding the block $\underline{\mathbf{X}}_k$ from left to right. The model can be used for anomaly detection by fixing the core tensor and some factor matrices while monitoring the changes along one or more specific modes (in our case mode two). Tensor decomposition is then first performed for a sampled (pre-selected) small block, followed by the analysis of changes in specific smaller–dimensional factor matrices $\mathbf{U}^{(n)}$.

4.5 Matrix/Tensor Cross-Approximation (MCA/TCA) for Dimensionality Reduction and Compression

Huge-scale matrices, can be factorized using the Matrix Cross-Approximation (MCA) method, which is also known under the names of Pseudo-Skeleton or CUR matrix decompositions [21, 22, 103, 104, 146, 185, 210]. The main idea behind the MCA is to provide reduced dimensionality of data through a linear combination of only a few "meaningful" components, which are exact replicas of columns and rows of the original data matrix [184]. Such an approach is based on the fundamental assumption that large datasets are highly redundant and can therefore be approximated by low-rank matrices, which significantly reduces computational complexity at the cost of a marginal loss of information. Consequently, a sufficiently precise low-rank TN approximation is obtained for data which exhibits internal structure or smoothness.

The MCA method factorizes a data matrix $\mathbf{X} \in \mathbb{R}^{I \times J}$ as [103,104] (see Figure 39):

$$\mathbf{X} = \mathbf{CUR} + \mathbf{E}, \tag{77}$$

where $\mathbf{C} \in \mathbb{R}^{I \times C}$ is a matrix constructed from C suitably selected columns of the data matrix $\mathbf{X}$, matrix $\mathbf{R} \in \mathbb{R}^{R \times J}$ consists of R rows of $\mathbf{X}$, and matrix $\mathbf{U} \in \mathbb{R}^{C \times R}$ is chosen so as to minimize the norm of the error $\mathbf{E} \in \mathbb{R}^{I \times J}$.

A simple modification of this formula yields the so-called CR matrix factorization:

$$\mathbf{X} \cong \mathbf{C}\tilde{\mathbf{R}} = \tilde{\mathbf{C}}\mathbf{R} \tag{78}$$

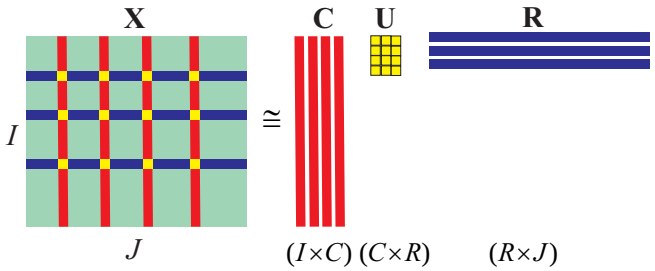


Figure 39: Principle of the matrix cross-approximation decomposition of a huge matrix $\mathbf{X}$ into a product of three matrices, whereby only a small-size core matrix $\mathbf{U}$ needs to be computed.

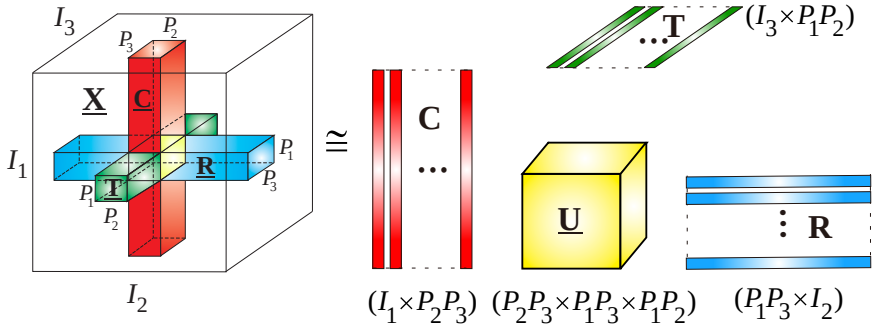
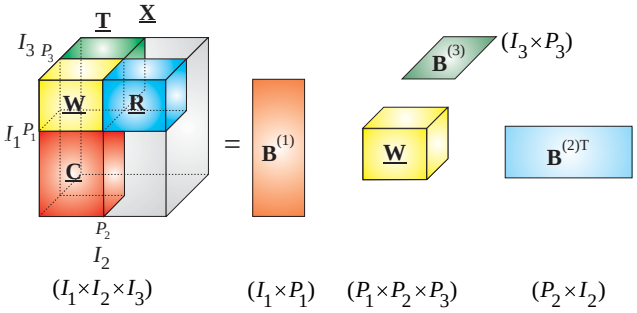


Figure 40: The principle of the tensor cross-approximation (TCA) for a large-scale 3rd-order tensor $\underline{\mathbf{X}} \cong \underline{\mathbf{U}} \times_1 \mathbf{C} \times_2 \mathbf{R} \times_3 \mathbf{T} = [\![\underline{\mathbf{U}}; \mathbf{C}, \mathbf{R}, \mathbf{T}]\!]$, where $\underline{\mathbf{U}} = \underline{\mathbf{W}} \times_1 \mathbf{W}_{(1)}^{\dagger} \times_2 \mathbf{W}_{(2)}^{\dagger} \times_3 \mathbf{W}_{(3)}^{\dagger} = [\![\underline{\mathbf{W}}; \mathbf{W}_{(1)}^{\dagger}, \mathbf{W}_{(2)}^{\dagger}, \mathbf{W}_{(3)}^{\dagger}]\!] \in \mathbb{R}^{P_2P_3 \times P1P_3 \times P_1P_2}$ and $\underline{\mathbf{W}} \in \mathbb{R}^{P_1 \times P_2 \times P_3}$. For simplicity of illustration, we assumed that the selected fibers are permuted in a such way so as to become clustered as subtensors, $\underline{\mathbf{C}} \in \mathbb{R}^{I_1 \times P_2 \times P_3}$, $\underline{\mathbf{R}} \in \mathbb{R}^{P_1 \times I_2 \times P_3}$ and $\underline{\mathbf{T}} \in \mathbb{R}^{P_1 \times P_2 \times I_3}$.

for which the bases can be either the columns, $\mathbf{C}$, or rows, $\mathbf{R}$, while $\tilde{\mathbf{R}} = \mathbf{U}\mathbf{R}$ and $\tilde{\mathbf{C}} = \mathbf{C}\mathbf{U}$.

For dimensionality reduction, $C \ll J$ and $R \ll I$, and the columns and rows of $\mathbf{X}$ should be chosen optimally, in the sense of providing a high "statistical leverage" and the best low-rank fit to the data matrix, while at the same time minimizing the cost function $||\mathbf{E}||_F^2$. For a given set of columns, $\mathbf{C}$, and rows, $\mathbf{R}$, the optimal choice for the core matrix is $\mathbf{U} = \mathbf{C}^{\dagger}\mathbf{X}(\mathbf{R}^{\dagger})^T$. This requires access to all the entries of $\mathbf{X}$ and is not practical or feasible for large-scale data. In such cases, a pragmatic choice for the core matrix would be $\mathbf{U} = \mathbf{W}^{\dagger}$, where the matrix $\mathbf{W} \in \mathbb{R}^{R \times C}$ is composed from the intersections of the selected rows and columns. It should be noted that, if $\text{rank}(\mathbf{X}) \leqslant \max\{C, R\}$, then

(a)



(b)

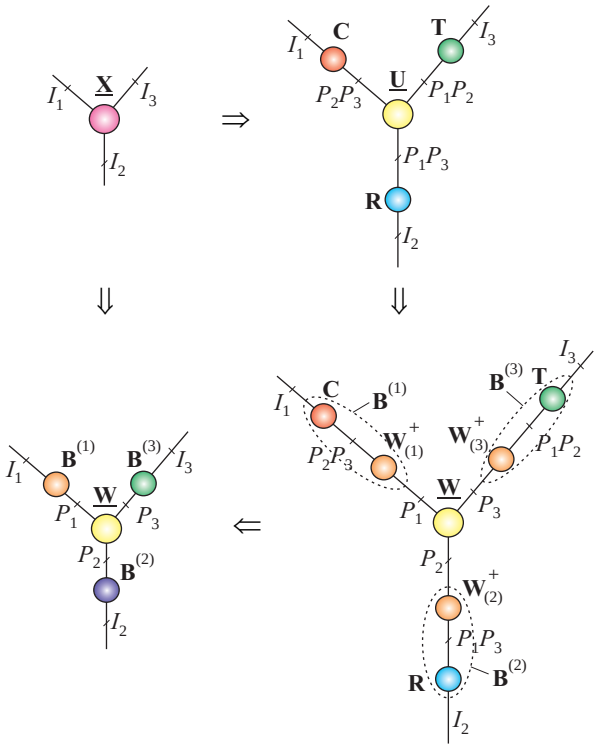


Figure 41: The Tucker decomposition of 3rd-order tensor using cross-approximation approach for a low-rank tensor. (a) Standard block diagram. (b) Transformation from the TCA in the Tucker format, $\mathbf{X} \cong \underline{\mathbf{U}} \times_1 \mathbf{C} \times_2 \mathbf{R} \times_3 \mathbf{T}$, to a standard Tucker representation, $\underline{\mathbf{X}} \cong \underline{\mathbf{W}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \times_3 \mathbf{B}^{(3)} = [\![\underline{\mathbf{W}}; \mathbf{C}\mathbf{W}^{\dagger}_{(1)}, \mathbf{R}\mathbf{W}^{\dagger}_{(2)}, \mathbf{T}\mathbf{W}^{\dagger}_{(3)}]\!]$, with a prescribed core tensor.

the cross-approximation is exact. For the general case, it has been proven that when the intersection sub-matrix $\mathbf{W}$ is of maximum volume[19], the matrix cross-approximation is close to the optimal SVD solution [103]. Finding a submatrix with maximum volume has exponential complexity, however, suboptimal matrices can be found using fast greedy algorithms [8, 188, 227, 277].

The concept of MCA can be generalized to tensors cross-approximation (TCA) (see Figure 40) through several approaches, including:

- Applying the MCA decomposition to one matricized version of the tensor data [185];
- Operating directly on fibers of a data tensor which admits a low-rank Tucker approximation, an approach termed the Fiber Sampling Tucker Decomposition (FSTD) [33–35].

Real-life structured data often admit good low-multilinear rank approximations, and the FSTD provides such a low-rank Tucker decomposition which is practical as it is directly expressed in terms of a relatively small number of fibers of the data tensor.

For example, for a 3rd-order tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, for which an exact rank-(R_1, R_2, R_3) Tucker representation exists, the FSTD selects $P_n \geqslant R_n$, $n = 1, 2, 3$, indices in each mode; this determines an intersection sub-tensor, $\underline{\mathbf{W}} \in \mathbb{R}^{P_1 \times P_2 \times P_3}$, so that the following exact Tucker representation can be obtained (see Figure 41):

$$\underline{\mathbf{X}} = [\![\underline{\mathbf{U}}; \mathbf{C}, \mathbf{R}, \mathbf{T}]\!], \tag{79}$$

where the core tensor is computed as $\underline{\mathbf{U}} = \underline{\mathbf{G}} = [\![\underline{\mathbf{W}}; \mathbf{W}_{(1)}^{\dagger}, \mathbf{W}_{(2)}^{\dagger}, \mathbf{W}_{(3)}^{\dagger}]\!]$, while the factor matrices, $\mathbf{C} \in \mathbb{R}^{I_1 \times P_2 P_3}$, $\mathbf{R} \in \mathbb{R}^{I_2 \times P_1 P_3}$, $\mathbf{T} \in \mathbb{R}^{I_3 \times P_1 P_2}$, contain the fibers which are the respective subsets of the columns $\underline{\mathbf{C}}$, rows $\underline{\mathbf{R}}$ and tubes $\underline{\mathbf{T}}$. An equivalent Tucker representation is then given by:

$$\underline{\mathbf{X}} = [\![\underline{\mathbf{W}}; \mathbf{C}\mathbf{W}_{(1)}^{\dagger}, \mathbf{R}\mathbf{W}_{(2)}^{\dagger}, \mathbf{T}\mathbf{W}_{(3)}^{\dagger}]\!]. \tag{80}$$

Observe that for $N = 2$, the TCA model simplifies into the MCA for a matrix case, $\mathbf{X} = \mathbf{CUR}$, for which the core matrix is $\mathbf{U} = [\![\mathbf{W}; \mathbf{W}_{(1)}^{\dagger}, \mathbf{W}_{(2)}^{\dagger}]\!] = \mathbf{W}^{\dagger}\mathbf{W}\mathbf{W}^{\dagger} = \mathbf{W}^{\dagger}$.

For a general case of an Nth-order tensor, we can show that [33] a tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, with a low multilinear rank $\{R_1, R_2, \ldots, R_N\}$, where $R_n \leqslant I_n, \ \forall n$, can be fully reconstructed via the TCA FSTD, $\underline{\mathbf{X}} = [\![\underline{\mathbf{U}}; \mathbf{C}^{(1)}, \mathbf{C}^{(2)}, \ldots, \mathbf{C}^{(N)}]\!]$, using only N factor matrices $\mathbf{C}^{(n)} \in \mathbb{R}^{I_n \times P_n}$, $(n = 1, 2, \ldots, N)$, built up from the fibers of the data and core tensors, $\underline{\mathbf{U}} = \underline{\mathbf{G}} = [\![\underline{\mathbf{W}}; \mathbf{W}_{(1)}^{\dagger}, \mathbf{W}_{(2)}^{\dagger}, \ldots, \mathbf{W}_{(N)}^{\dagger}]\!]$, under the condition that the subtensor $\underline{\mathbf{W}} \in \mathbb{R}^{P_1 \times P_2 \times \cdots \times P_N}$ with $P_n \geqslant R_n, \ \forall n$, has the multilinear rank $\{R_1, R_2, \ldots, R_N\}$.

[19] The volume of a sub-matrix $\mathbf{W}$ is defined as $|\det(\mathbf{W})|$.

The selection of a minimum number of suitable fibers depends upon a chosen optimization criterion. A strategy which requires access to only a small subset of entries of a data tensor, achieved by selecting the entries with maximum modulus within each single fiber, is given in [33]. These entries are selected sequentially using a deflation approach, thus making the tensor cross-approximation FSTD algorithm suitable for approximation of very large-scale but relatively low-order tensors (including tensors with missing fibers or entries).

4.6 Multiway Component Analysis (MWCA) Using Constrained Tensor Decompositions

4.6.1 Multilinear component analysis using Tucker decomposition

The great success of 2-way component analyses (PCA, ICA, NMF, SCA) is largely due to the existence of very efficient algorithms and the possibility to extract components with a desired physical meaning, provided by the various flexible constraints exploited in these methods. Without these constraints, matrix factorizations would be less useful in practice, as the components would have only mathematical but not physical meaning.

Similarly, for tensor factorization/decompositions, we need to impose suitable constraints to the desired components. In fact, there is much more flexibility for tensors, since different constraints can be imposed for factorization of matricized tensor $\mathbf{X}_{(n)}$ in every mode n. Indeed, seldom do the components of a data tensor exhibit the same properties in all modes, and for physically meaningful representation different constraints can be imposed in different modes (see Algorithm 7). This underpins the concept of multiway component analysis (MWCA) and its flexibility in choosing the mode-wise constraints; a Tucker representation of MWCA naturally accommodates such diversities in different modes [51]. Besides the orthogonality, alternative constraints in the Tucker format include statistical independence, sparsity, smoothness and nonnegativity [53, 266, 297] (see Table 7).

The multiway component analysis (MWCA) based on the Tucker-N model can be computed directly in two or three steps:

1. For each mode n $(n = 1, 2, \ldots, N)$ perform model reduction and matricization of data tensors sequentially, then apply a suitable set of 2-way CA/BSS algorithms to the so reduced unfolding matrices, $\tilde{\mathbf{X}}_{(n)}$. In each mode, we can apply different constraints and a different 2-way CA algorithm [51].

2. Compute the core tensor using, e.g., the inversion formula, $\hat{\underline{\mathbf{G}}} = \underline{\mathbf{X}} \times_1 \mathbf{B}^{(1)\dagger} \times_2 \mathbf{B}^{(2)\dagger} \cdots \times_N \mathbf{B}^{(N)\dagger}$ [297]. This step is quite important because core tensors often model the complex links among the multiple components in different modes [53].

3. Optionally, perform fine tuning of factor matrices and the core tensor by ALS minimizing of a suitable cost function, e.g., $||\underline{\mathbf{X}} - [\![\underline{\mathbf{G}}; \mathbf{B}^{(1)}, \ldots, \mathbf{B}^{(N)}]\!]||_F^2$, subject to specific imposed constraints.

4.6.2 Analysis of coupled multi-block matrix/tensors – Linked Multiway Component Analysis (LMWCA)

We have shown that TDs provide natural extensions of blind source separation (BSS) and 2-way (matrix) Component Analysis to multi-way component analysis (MWCA) methods.

In addition, TDs are suitable for the coupled multiway analysis of multi-block datasets, possibly with missing values and corrupted by noise. To illustrate the simplest scenario for multi-block analysis, consider the block matrices, $\mathbf{X}^{(k)} \in \mathbb{R}^{I \times J}$, which need to be approximately jointly factorized as

$$\mathbf{X}^{(k)} \cong \mathbf{A}\mathbf{G}^{(k)}\mathbf{B}^T, \quad (k = 1, 2, \ldots, K), \tag{81}$$

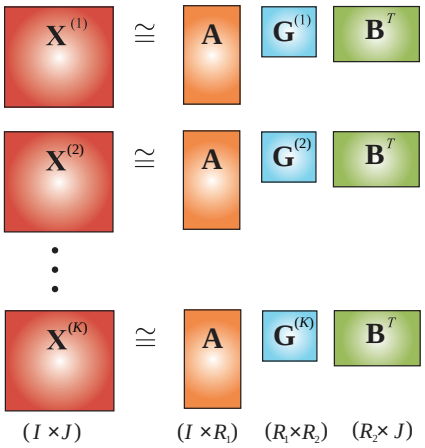
where $\mathbf{A} \in \mathbb{R}^{I \times R_1}$ and $\mathbf{B} \in \mathbb{R}^{J \times R_2}$ are common factor matrices and $\mathbf{G}^{(k)} \in \mathbb{R}^{R_1 \times R_2}$ are reduced-size matrices, while the number of data matrices K can be huge (hundreds of millions or more matrices). Such a simple model is referred to as the Population Value Decomposition (PVD) [63]. Note that the PVD is equivalent to the unconstrained or constrained Tucker-2 model, as illustrated in Figure 42. In a special cases with square diagonal matrices, $\mathbf{G}^{(k)}$, the model is equivalent to CP decomposition and is related to joint matrix diagonalization [41, 76, 255]. Furthermore, if $\mathbf{A} = \mathbf{B}$ then the PVD model is equivalent to the RESCAL model [197].

Observe that the PVD/Tucker-2 model is quite general and flexible, since any high-order tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ (with $N > 3$), can be reshaped into a "thin and deep" or equivalently "skinny and tall" 3rd-order tensor, $\underline{\widetilde{\mathbf{X}}} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \cdots I_N}$, for which Tucker-2/PVD algorithms can be applied. As previously mentioned, various constraints, including sparsity, nonnegativity or smoothness can be imposed on the factor matrices, $\mathbf{A}$ and $\mathbf{B}$, to obtain physically meaningful and unique components.

A simple SVD/QR based algorithm for the PVD, with orthogonality constraints, is presented in Algorithm 8 [61, 63, 274]. However, it should be noted that this algorithm does not provide an optimal solution in the sense of the absolute minimum of the cost function, $\sum_{k=1}^{K} ||\mathbf{X}_k - \mathbf{A}\mathbf{G}_k\mathbf{B}^T||_F^2$, and for data with Gaussian noise, better performances can be achieved by applying the HOOI-2 Algorithm 5, for $N = 2$.

Linked MWCA. Consider the analysis of multi-modal high-dimensional data collected under the same or very similar conditions, for example, a set of EEG and MEG or EEG and fMRI signals recorded for different subjects over many trials and under the same experimental configurations and mental tasks. Such data share some common latent (hidden) components but can also have their

(a)



(b)

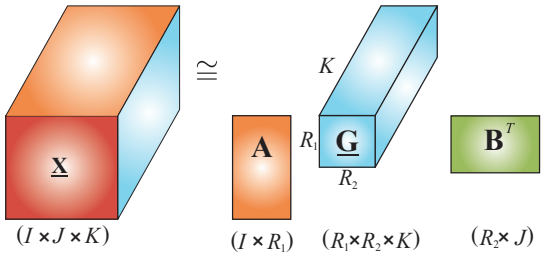


Figure 42: Principle of the Population Value Decomposition (PVD). (a) Simultaneous multi-block matrix factorizations. (b) Equivalent representation of the PVD as the constrained or unconstrained Tucker-2 decomposition, $\underline{\mathbf{X}} \cong \underline{\mathbf{G}} \times_1 \mathbf{A} \times_2 \mathbf{B}$. The objective is to find the common factor matrices, $\mathbf{A}$, $\mathbf{B}$ and the core tensor, $\underline{\mathbf{G}} \in \mathbb{R}^{R_1 \times R_2 \times K}$.

own independent features. As a result, it is advantageous and natural to analyze such data in a linked way instead of independently. In such a scenario, the PVD model can be straightforwardly generalized to multi-block matrix/tensor datasets [48, 300, 302].

The linked multiway component analysis (LMWCA) for multi-block tensors data can therefore be formulated as a set of approximate simultaneous (joint) Tucker-$(1, N)$ decompositions of a set of data tensors, $\underline{\mathbf{X}}^{(k)} \in \mathbb{R}^{I_1^{(k)} \times I_2^{(k)} \times \cdots \times I_N^{(k)}}$,

Algorithm 8: **Population Value Decomposition (PVD) with orthogonality constraints**

Input: A set of matrices $\mathbf{X}_k \in \mathbb{R}^{I \times J}$, for $k = 1, \ldots, K$, $(K \gg \max\{I, J\})$
Output: Factor matrices $\mathbf{A} \in \mathbb{R}^{I \times R_1}$, $\mathbf{B} \in \mathbb{R}^{J \times R_2}$ and $\mathbf{G}_k \in \mathbb{R}^{R_1 \times R_2}$, with orthogonality constraints $\mathbf{A}^T\mathbf{A} = \mathbf{I}_{R_1}$ and $\mathbf{B}^T\mathbf{B} = \mathbf{I}_{R_2}$

1: **for** $k = 1$ to K **do**
2: Perform truncated SVD, $\mathbf{X}_k = \mathbf{U}_k\mathbf{S}_k\mathbf{V}_k^T$, using R largest singular values
3: **end for**
4: Construct long and thin matrices: $\mathbf{U} = [\mathbf{U}_1\mathbf{S}_1, \ldots, \mathbf{U}_K\mathbf{S}_K] \in \mathbb{R}^{I \times KR}$ and $\mathbf{V} = [\mathbf{V}_1\mathbf{S}_1, \ldots, \mathbf{V}_K\mathbf{S}_K] \in \mathbb{R}^{J \times KR}$
5: Perform SVD (or QR) for the matrices $\mathbf{U}$ and $\mathbf{V}$
Obtain common orthogonal matrices $\mathbf{A}$ and $\mathbf{B}$ as left-singular matrices of $\mathbf{U}$ and $\mathbf{V}$, respectively
6: **for** $k = 1$ to K **do**
7: Compute $\mathbf{G}_k = \mathbf{A}^T\mathbf{X}_k\mathbf{B}$
8: **end for**

with $I_1^{(k)} = I_1$ and $k = 1, 2, \ldots, K$, in the form (see Figure 43):

$$\underline{\mathbf{X}}^{(k)} = \underline{\mathbf{G}}^{(k)} \times_1 \mathbf{B}^{(1,k)}, \quad (k = 1, 2, \ldots K) \tag{82}$$

where each factor (component) matrix, $\mathbf{B}^{(1,k)} = [\mathbf{B}_C^{(1)}, \mathbf{B}_I^{(1,k)}] \in \mathbb{R}^{I_1 \times R_1}$, comprises two sets of components: (1) Components $\mathbf{B}_C^{(1)} \in \mathbb{R}^{I_1 \times C}$ (with $0 \leqslant C \leqslant R_1$), which are common for all the available blocks and correspond to identical or maximally correlated components, and (2) components $\mathbf{B}_I^{(1,k)} \in \mathbb{R}^{I_1 \times (R_1 - C)}$, which are different independent processes for each block, k, these can be for example, latent variables independent of excitations or stimuli/tasks. The objective is therefore to estimate the common components, $\mathbf{B}_C^{(1)}$, and independent (individual) components, $\mathbf{B}_I^{(1,k)}$ (see Figure 43) [48].

If $\mathbf{B}^{(n,k)} = \mathbf{B}_C^{(n)} \in \mathbb{R}^{I_n \times R_n}$ for a specific mode n (in our case $n = 1$), and under the additional assumption that the block tensors are of the same order and size, the problem simplifies into generalized Common Component Analysis or tensor Population Value Decomposition (PVD) and can be solved by concatenating all data tensors along one mode, followed by constrained Tucker or CP decompositions [218].

In a more general scenario, when $C_n < R_n$, we can unfold each data tensor $\underline{\mathbf{X}}^{(k)}$ in the common mode, and perform a set of simultaneous matrix factorizations, $\mathbf{X}_{(1)}^{(k)} \cong \mathbf{B}_C^{(1)}\mathbf{A}_C^{(1,k)} + \mathbf{B}_I^{(1,k)}\mathbf{A}_I^{(1,k)}$, by solving the constrained optimization

Algorithm 9: **Tucker-2 decomposition with orthogonality constraints**

Input: A 3rd-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$, with $K \gg \max\{I, J\}$) and estimation accuracy ε

Output: A set of orthogonal matrices $\mathbf{A} \in \mathbb{R}^{I \times R_1}$, $\mathbf{B} \in \mathbb{R}^{J \times R_2}$ and core tensor $\underline{\mathbf{G}} \in \mathbb{R}^{R_1 \times R_2 \times K}$, which satisfies constraint $||\underline{\mathbf{X}} - \underline{\mathbf{G}} \times_1 \mathbf{A} \times \mathbf{B}||_F^2 \leqslant \varepsilon^2$, s.t, $\mathbf{A}^T\mathbf{A} = \mathbf{I}_{R_1}$ and $\mathbf{B}^T\mathbf{B} = \mathbf{I}_{R_2}$.

1: Initialize $\mathbf{A} = \mathbf{I}_{I_1} \in \mathbb{R}^{I_1 \times I_1}$, $R_1 = I_1$
2: **while** not converged or iteration limit is not hit **do**
3: Compute tensor $\underline{\mathbf{Z}}^{(1)} = \underline{\mathbf{X}} \times_1 \mathbf{A}^T \in \mathbb{R}^{R_1 \times I_2 \times K}$
4: Unfold the tensor $\mathbf{Z}_{(2)}^{(1)} = \text{reshape}\left(\underline{\mathbf{Z}}^{(1)}, I_2, R_1 K\right) \in \mathbb{R}^{I_2 \times R_1 K}$
5: Compute EVD of a small matrix $\mathbf{Q}_1 = \mathbf{Z}_{(2)}^{(1)} \mathbf{Z}_{(2)}^{(1)\,T} \in \mathbb{R}^{I_2 \times I_2}$ as $\mathbf{Q}_1 = \mathbf{B}\ \text{diag}\left(\lambda_1, \cdots, \lambda_{R_2}\right)\ \mathbf{B}^T$, such that $\sum_{r_2=1}^{R_2} \lambda_{r_2} \geqslant ||\underline{\mathbf{X}}||_F^2 - \varepsilon^2 \geqslant \sum_{r_2=1}^{R_2-1} \lambda_{r_2}$
6: Compute tensor $\underline{\mathbf{Z}}^{(2)} = \underline{\mathbf{X}} \times_2 \mathbf{B}^T \in \mathbb{R}^{I_1 \times R_2 \times K}$
7: Unfold the tensor $\mathbf{Z}_{(1)}^{(2)} = \text{reshape}\left(\underline{\mathbf{Z}}^{(2)}, I_1, R_2 K\right) \in \mathbb{R}^{I_1 \times R_2 K}$
8: Compute EVD of a small matrix $\mathbf{Q}_2 = \mathbf{Z}_{(1)}^{(2)} \mathbf{Z}_{(1)}^{(2)\,T} \in \mathbb{R}^{I_1 \times I_1}$ as $\mathbf{Q}_2 = \mathbf{A}\ \text{diag}\left(\lambda_1, \cdots, \lambda_{R_1}\right)\ \mathbf{A}^T$, such that $\sum_{r_1=1}^{R_1} \lambda_{r_1} \geqslant ||\underline{\mathbf{X}}||_F^2 - \varepsilon^2 \geqslant \sum_{r_1=1}^{R_1-1} \lambda_{r_1}$
9: **end while**
10: Compute core tensor $\underline{\mathbf{G}} = \underline{\mathbf{X}} \times_1 \mathbf{A}^T \times \mathbf{B}^T$
11: **return** $\mathbf{A}$, $\mathbf{B}$ and $\underline{\mathbf{G}}$.

problems:

$$\begin{aligned} \min \sum_{k=1}^{K} \|\mathbf{X}_{(1)}^{(k)} - \mathbf{B}_C^{(1)} \mathbf{A}_C^{(1,k)} - \mathbf{B}_I^{(1,k)} \mathbf{A}_I^{(1,k)}\|_F \\ + P(\mathbf{B}_C^{(1)}),\ \ s.t.\ \ \mathbf{B}_C^{(1)\,T} \mathbf{B}_I^{(1,k)} = \mathbf{0}\ \ \forall k, \end{aligned} \tag{83}$$

where the symbol P denotes the penalty terms which impose additional constraints on the common components, $\mathbf{B}_C^{(1)}$, in order to extract as many common components as possible. In the special case of orthogonality constraints, the problem can be transformed into a generalized eigenvalue problem which could be solved by the power method [300]. The key point is to assume that common factor sub-matrices, $\mathbf{B}_C^{(1)}$, are present in all data blocks and hence reflect structurally complex latent (hidden) and intrinsic links between the data blocks. In practice, the number of common components, C is unknown and should be estimated, e.g. by using method proposed in [300].

The linked multiway component analysis (LMWCA) model thus provides a flexible and general framework, and complements currently available tech-

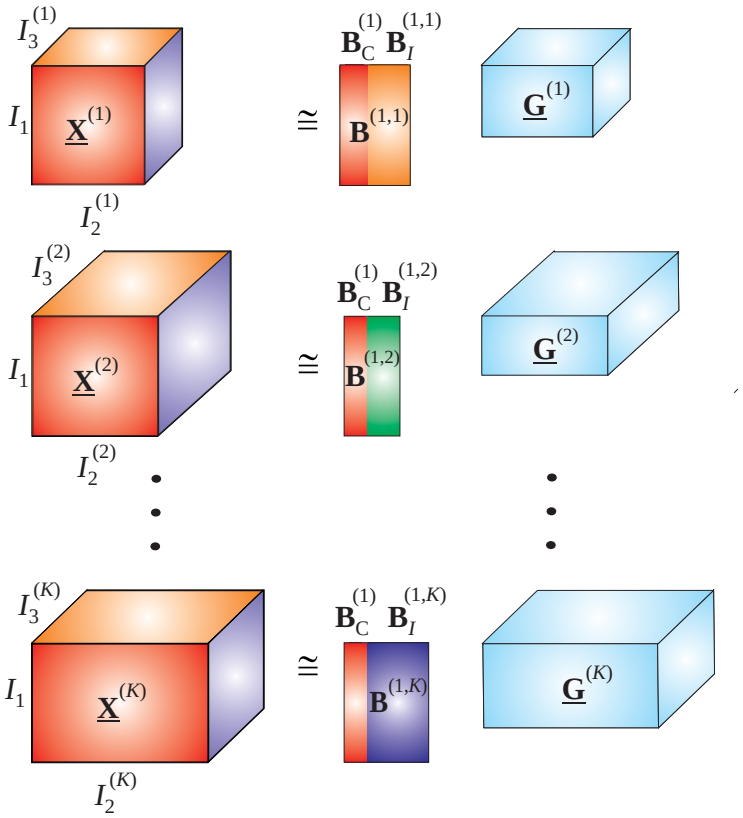


Figure 43: Linked Multiway Component Analysis (LMWCA) for coupled 3rd-order data tensors; these can have different dimensions in each mode, except for the first mode. The objective is to find the common components, $\mathbf{B}_C^{(1)} \in \mathbb{R}^{I_1 \times C}$, and individual components, $\mathbf{B}_I^{(1,k)} \in \mathbb{R}^{I_1 \times (R_1 - C)}$, where $C \leqslant R_1$ is the number of common components in mode-1.

niques for group component analysis and feature extraction from multi-block datasets. In this sense, LMWCA attempts to estimate both common and individual components, and is a natural extension of group ICA, PVD, and CCA/PLS methods (see [48, 289, 300, 302] and references therein). Moreover, the concept of LMWCA can be generalized to tensor networks, as illustrated in Figure 44.

4.7 Nonlinear Tensor Decompositions – Infinite Tucker

The Infinite Tucker model and its modification, the Distributed Infinite Tucker (DinTucker), generalize the standard Tucker decomposition to infinite feature spaces using the kernel and Bayesian approaches [252, 284, 294].

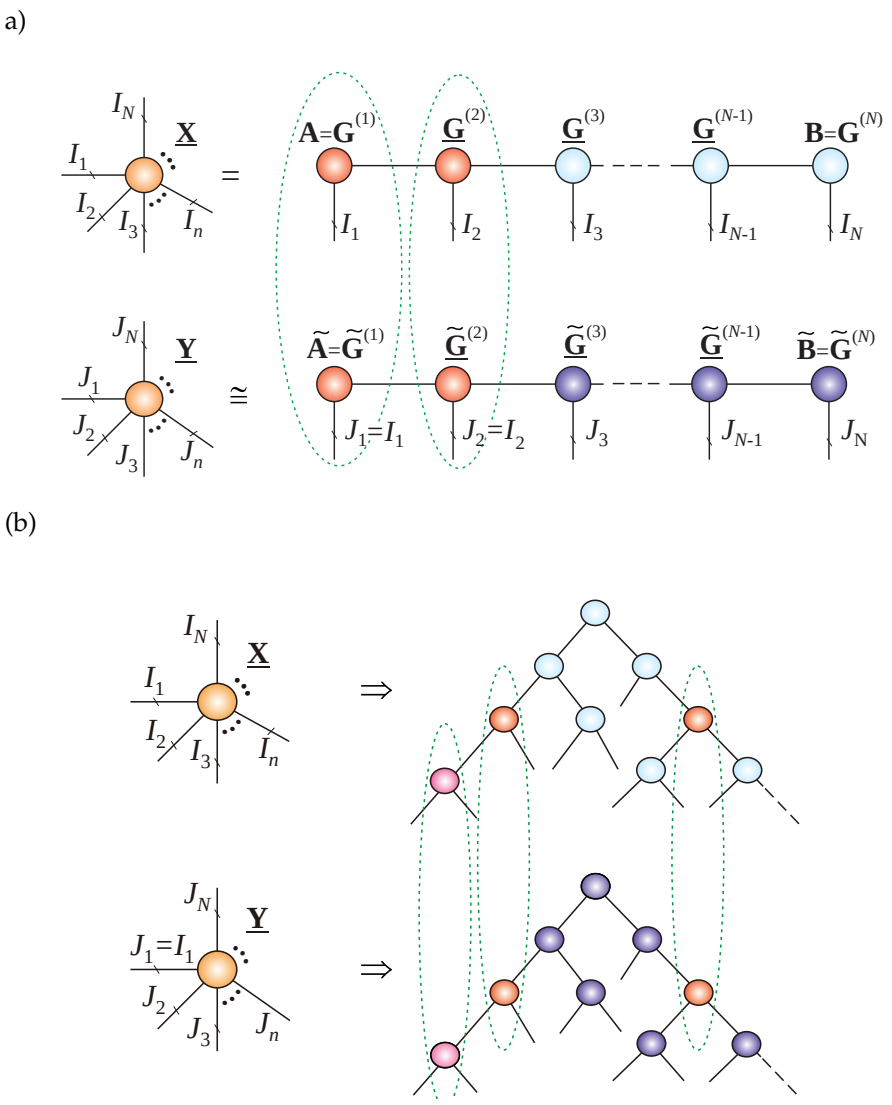


Figure 44: Conceptual models of generalized Linked Multiway Component Analysis (LMWCA) applied to cores of high-order TNs. The objective is to find a suitable tensor decomposition such that the maximal number of cores (components) will maximally correlated. (a) Linked Tensor Train (TT) networks. (b) Linked Hierarchical Tucker (HT) networks with some correlated cores indicated by ellipses in broken lines.

Consider the classic Tucker-N model for an Nth-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, given by

$$\begin{aligned} \underline{\mathbf{X}} &= \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \times_2 \cdots \times_N \mathbf{B}^{(N)} \\ &= [\![\underline{\mathbf{G}}; \mathbf{B}^{(1)}, \ldots, \mathbf{B}^{(N)}]\!] \end{aligned} \tag{84}$$

and its vectorized version

$$\text{vec}(\underline{\mathbf{X}}) = (\mathbf{B}^{(1)} \otimes_L \cdots \otimes_L \mathbf{B}^{(N)}) \, \text{vec}(\underline{\mathbf{G}}).$$

Furthermore, let assume that the noisy data tensor is modeled as

$$\underline{\mathbf{Y}} = \underline{\mathbf{X}} + \underline{\mathbf{E}}, \tag{85}$$

where $\underline{\mathbf{E}}$ represents the tensor of additive Gaussian noise.

Using the Bayesian framework and tensor-variate Gaussian processes (TGP) for Tucker decomposition, a standard normal prior can be assigned over each entry, $g_{r_1, r_2, \ldots, r_N}$, of an Nth-order core tensor, $\underline{\mathbf{G}} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$, in order to marginalize out $\underline{\mathbf{G}}$ and express the probability density function of tensor $\underline{\mathbf{X}}$ [47, 284, 294] in the form

$$\begin{aligned} & p\left(\underline{\mathbf{X}} \,|\, \mathbf{B}^{(1)}, \ldots, \mathbf{B}^{(N)}\right) = \mathcal{N}\left(\text{vec}(\underline{\mathbf{X}}); \mathbf{0}, \mathbf{C}^{(1)} \otimes_L \cdots \otimes_L \mathbf{C}^{(N)}\right) \\ & = \frac{\exp\left(-\frac{1}{2} || [\![\underline{\mathbf{X}}; (\mathbf{C}^{(1)})^{-1/2}, \ldots, (\mathbf{C}^{(N)})^{-1/2}]\!] ||^2\right)}{(2\pi)^{I/2} \prod_{n=1}^{N} |\mathbf{C}^{(n)}|^{-I/(2I_n)}} \end{aligned} \tag{86}$$

where $I = \prod_n I_n$ and $\mathbf{C}^{(n)} = \mathbf{B}^{(n)} \, \mathbf{B}^{(n)\,T} \in \mathbb{R}^{I_n \times I_n}$ for $n = 1, 2, \ldots, N$.

In order to model unknown, complex, and potentially nonlinear interactions between the latent factors, each row, $\bar{\mathbf{b}}_{i_n}^{(n)} \in \mathbb{R}^{1 \times R_n}$, within $\mathbf{B}^{(n)}$, is replaced by a nonlinear feature transformation $\Phi(\bar{\mathbf{b}}_{i_n}^{(n)})$ using the kernel trick [292], whereby the nonlinear covariance matrix $\mathbf{C}^{(n)} = k(\mathbf{B}^{(n)}, \, \mathbf{B}^{(n)})$ replaces the standard covariance matrix, $\mathbf{B}^{(n)} \, \mathbf{B}^{(n)\,T}$. Using such a nonlinear feature mapping, the original Tucker factorization is performed in an infinite feature space, while Eq. (86) defines a Gaussian process (GP) on a tensor, called the TGP, where the inputs come from a set of factor matrices $\{\mathbf{B}^{(1)}, \ldots, \mathbf{B}^{(N)}\} = \{\mathbf{B}^{(n)}\}$.

For a noisy data tensor, the joint probability density function is given by

$$p(\underline{\mathbf{Y}}, \underline{\mathbf{X}}, \{\mathbf{B}^{(n)}\}) = p(\{\mathbf{B}^{(n)}\}) \; p(\underline{\mathbf{X}} \,|\, \{\mathbf{B}^{(n)}\}) \; p(\underline{\mathbf{Y}} | \underline{\mathbf{X}}). \tag{87}$$

To improve scalability, the observed noisy tensor $\underline{\mathbf{Y}}$ can be split into K subtensors $\{\underline{\mathbf{Y}}_1, \ldots, \underline{\mathbf{Y}}_K\}$, whereby each subtensor $\underline{\mathbf{Y}}_k$ is sampled from its own GP based model on factor matrices, $\{\tilde{\mathbf{B}}_k^{(n)}\} = \{\tilde{\mathbf{B}}_k^{(1)}, \ldots, \tilde{\mathbf{B}}_k^{(N)}\}$. The factor matrices

can then be merged via a prior distribution:

$$
\begin{aligned}
p(\{\tilde{\mathbf{B}}_k^{(n)}\}|\{\mathbf{B}^{(n)}\}) &= \prod_{n=1}^{N} p(\tilde{\mathbf{B}}_k^{(n)}|\mathbf{B}^{(n)}) \\
&= \prod_{n=1}^{N} \mathcal{N}(\text{vec}(\tilde{\mathbf{B}}_k^{(n)})|\text{vec}(\mathbf{B}^{(n)})), \lambda\mathbf{I}), \qquad (88)
\end{aligned}
$$

where $\lambda > 0$ is a variance parameter which controls the similarity between the corresponding factor matrices. The above model is referred to as DinTucker [294].

The full covariance matrix, $\mathbf{C}^{(1)} \otimes \cdots \otimes \mathbf{C}^{(N)} \in \mathbb{R}^{\prod_n I_n \times \prod_n I_n}$, can have a prohibitively large size and be extremely sparse. For such cases, an alternative nonlinear tensor decomposition model has been recently developed, which does not either explicitly or implicitly, exploit the Kronecker structure of covariance matrices [295]. Within this model, for each tensor entry, $x_{i_1,\ldots,i_N} = x_{\mathbf{i}}$, with $\mathbf{i} = (i_1, i_2, \ldots, i_N)$, an input vector $\mathbf{b}_{\mathbf{i}}$ is constructed by concatenating the corresponding row vectors of factor (latent) matrices, $\mathbf{B}^{(n)}$, for all N modes, as follows:

$$
\mathbf{b}_{\mathbf{i}} = [\bar{\mathbf{b}}_{i_1}^{(1)}, \ldots, \bar{\mathbf{b}}_{i_N}^{(N)}] \in \mathbb{R}^{1 \times \prod_{n=1}^{N} R_n}. \qquad (89)
$$

This allows us to formalize the process through an (unknown) nonlinear transformation:

$$
x_{\mathbf{i}} = f(\mathbf{b}_{\mathbf{i}}) = f([\bar{\mathbf{b}}_{i_1}^{(1)}, \ldots, \bar{\mathbf{b}}_{i_N}^{(N)}]) \qquad (90)
$$

for which a zero-mean multivariate Gaussian distribution is determined by $\mathbf{B}_{\mathcal{S}} = \{\mathbf{b}_{\mathbf{i}_1}, \ldots, \mathbf{b}_{\mathbf{i}_M}\}$ and $\mathbf{f}_{\mathcal{S}} = \{f(\mathbf{b}_{\mathbf{i}_1}), \ldots, f(\mathbf{b}_{\mathbf{i}_M})\}$. This allows us to construct the following probability function

$$
p\left(\mathbf{f}_{\mathcal{S}}|\{\mathbf{B}^{(n)}\}\right) = \mathcal{N}\left(\mathbf{f}_{\mathcal{S}}|\mathbf{0},\ k(\mathbf{B}_{\mathcal{S}},\ \mathbf{B}_{\mathcal{S}})\right), \qquad (91)
$$

where $k(\cdot,\cdot)$ is a nonlinear covariance function which can be expressed as $k(\mathbf{b}_{\mathbf{i}}, \mathbf{b}_{\mathbf{j}}) = k(([\bar{\mathbf{b}}_{i_1}^{(1)}, \ldots, \bar{\mathbf{b}}_{i_N}^{(N)}]), ([\bar{\mathbf{b}}_{j_1}^{(1)}, \ldots, \bar{\mathbf{b}}_{j_N}^{(N)}]))$ and $\mathcal{S} = [\mathbf{i}_1, \ldots, \mathbf{i}_M]$.

In order to assign a standard normal prior over the factor matrices, $\{\mathbf{B}^{(n)}\}$, we assume that for selected entries, $\mathbf{x} = [x_{\mathbf{i}_1}, \ldots, x_{\mathbf{i}_M}]$, of a tensor $\underline{\mathbf{X}}$, the observed noisy entries, $\mathbf{y} = [y_{\mathbf{i}_1}, \ldots, y_{\mathbf{i}_M}]$, of the observed tensor $\underline{\mathbf{Y}}$, are sampled from the following joint probability model

$$
\begin{aligned}
&p(\mathbf{y}, \mathbf{x}, \{\mathbf{B}^{(n)}\}) = \qquad (92) \\
&= \prod_{n=1}^{N} \mathcal{N}(\text{vec}(\mathbf{B}^{(n)})|\mathbf{0}, \mathbf{I})\ \mathcal{N}(\mathbf{x}|\mathbf{0}, k(\mathbf{B}_{\mathcal{S}},\ \mathbf{B}_{\mathcal{S}}))\ \mathcal{N}(\mathbf{y}|\mathbf{x}, \beta^{-1}\mathbf{I}),
\end{aligned}
$$

where β represents noise variance. Such a model no longer corresponds to the Tucker model.

These nonlinear and probabilistic models can be potentially applied for data tensors or function-related tensors comprising large number of entries, typically with millions of non-zero entries and billions of zero entries. Even if only nonzero entries are used, exact inference of the above nonlinear tensor decomposition models may still be intractable. To alleviate this problem, a distributed variational inference algorithm has been developed with many potential applications for big data [294]. The algorithm is based on sparse GP, an efficient MapReduce framework which uses a small set of inducing points to break the dependencies between random function values [256].

5 Discussion and Conclusions (for Part 1 and Part 2)

We have provided a systematic and example-rich guide to the properties and applications of tensor network (TNs) methodologies, and have demonstrated their promise as a tool for the analysis of extreme-scale multidimensional data. Our main aim has been to illustrate that, owing to the compression ability which stems from the distributed way in which they represent data and process information, TNs can be naturally employed for a wide range of large-scale optimization problems - especially those based on linear/multilinear dimensionality reduction. Indeed, current applications of TNs include generalized multivariate regression, compressed sensing, multi-way blind source separation, sparse representation and coding, feature extraction, classification, clustering and data fusion.

With multilinear algebra as their mathematical backbone, TNs have been shown to have intrinsic advantages over the flat two-dimensional view provided by matrices, including the ability to model both the strong and weak couplings among multiple variables, and to cater for multimodal, incomplete and noisy data. We have discussed a scalable framework for the distributed implementation of optimization algorithms, which transforms huge-scale optimization problems into linked small-scale optimization sub-problems of the same type. In that sense, TNs can be seen as a natural bridge between small-scale and very large-scale optimization paradigms, which allows for any efficient numerical algorithm to be applied to such local optimization sub-problems. We have illustrated this with concrete examples using graphical approaches.

Although the research on tensor networks for dimensionality reduction and optimization problems is still in its infancy, recent works indicate that many challenging problems for extremely large-scale data can be approximately solved through tensor networks. Our focus has been on illustrating that the distributed representations performed through low-rank TNs provide a linearly or even sub-linearly scalable framework for the implementation of optimization algorithms, and a vehicle to convert a mathematically or computationally intractable huge-scale paradigm into a set of feasible sub-tasks. We hope that the readers will find the approaches presented in this monograph helpful in ad-

vancing seamlessly from numerical linear algebra to numerical multilinear algebra. Given that in many modern applications multiway arrays (tensors) arise either explicitly or indirectly through the tensorization of vectors and matrices, we foresee this material serving as a useful foundation for further studies on many machine learning problems for data of otherwise prohibitively large volume, variety, or veracity.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: A system for large-scale machine learning. *ArXiv e-prints*, May 2016.

[2] P.-A. Absil and I. V. Oseledets. Low-rank retractions: A survey and new results. *Computational Optimization and Applications*, 62(1):5–29, 2015.

[3] P.A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2009.

[4] E. Acar, D. M. Dunlavy, and T.G. Kolda. A scalable optimization approach for fitting canonical tensor decompositions. *Journal of Chemometrics*, 25(2):67–86, February 2011.

[5] E. Acar and B. Yener. Unsupervised multiway data analysis: A literature survey. *IEEE Transactions on Knowledge and Data Engineering*, 21:6–20, 2009.

[6] I. Affleck, T. Kennedy, E.H. Lieb, and H. Tasaki. Rigorous results on valence-bond ground states in antiferromagnets. *Physical Review Letters*, 59(7):799, 1987.

[7] G.I. Allen and M. Maletic-Savatic. Sparse non-negative generalized PCA with applications to metabolomics. *Bioinformatics*, 27 (21):3029–3035, 2011.

[8] D. Anderson, S. Du, M. Mahoney, C. Melgaard, K. Wu, and M. Gu. Spectral gap error bounds for improving CUR matrix decomposition and the Nyström. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, pages 19–27, 2015.

[9] C.A. Andersson and R. Bro. The N-way toolbox for MATLAB. *Chemometrics and Intelligent Laboratory Systems*, 52(1):1–4, 2000.

[10] W. Austin, G. Ballard, and T.G. Kolda. Parallel tensor compression for large-scale scientific data. *arXiv preprint arXiv:1510.06689*, 2015.

[11] F.R. Bach and M.I. Jordan. Kernel independent component analysis. *The Journal of Machine Learning Research*, 3:1–48, 2003.

[12] F.R. Bach and M.I. Jordan. A probabilistic interpretation of canonical correlation analysis. Technical report, Department of Statistics, Unversity of California, Berkeley, 2005.

[13] B.W. Bader and T.G. Kolda. Matlab tensor classes for fast algorithm prototyping. Technical report, ACM Transactions on Mathematical Software, 2004.

[14] B.W. Bader and T.G. Kolda. MATLAB tensor toolbox, 2015.

[15] J. Ballani and L. Grasedyck. Tree adaptive approximation in the hierarchical tensor format. *SIAM Journal on Scientific Computing*, 36(4):A1415–A1431, 2014.

[16] J. Ballani, L. Grasedyck, and M. Kluge. A review on adaptive low-rank approximation techniques in the hierarchical tensor format. In *Extraction of Quantifiable Information from Complex Systems*, pages 195–210. Springer, 2014.

[17] G. Ballard, A.R. Benson, A. Druinsky, B. Lipshitz, and O. Schwartz. Improving the numerical stability of fast matrix multiplication algorithms. *arXiv preprint arXiv:1507.00687*, 2015.

[18] G. Ballard, A. Druinsky, N. Knight, and O. Schwartz. Brief announcement: Hypergraph partitioning for parallel sparse matrix-matrix multiplication. In *Proceedings of the 27th ACM on Symposium on Parallelism in Algorithms and Architectures*, pages 86–88. ACM, 2015.

[19] K. Batselier, H. Liu, and N. Wong. A constructive algorithm for decomposing a tensor into a finite sum of orthonormal rank-1 terms. *ArXiv e-prints*, July 2014.

[20] K. Batselier and N. Wong. A constructive arbitrary-degree Kronecker product decomposition of tensors. *ArXiv e-prints*, July 2016.

[21] M. Bebendorf. Adaptive cross-approximation of multivariate functions. *Constructive Approximation*, 34(2):149–179, 2011.

[22] M. Bebendorf, C. Kuske, and R. Venn. Wideband nested cross–approximation for Helmholtz problems. *Numerische Mathematik*, 130(1):1–34, 2015.

[23] R.E. Bellman. *Adaptive Control Processes*. Princeton University Press, Princeton, NJ, 1961.

[24] P. Benner, V. Khoromskaia, and B.N. Khoromskij. A reduced basis approach for calculation of the Bethe–Salpeter excitation energies by using low-rank tensor factorisations. *Molecular Physics*, 114(7-8):1148–1161, 2016.

[25] A.R. Benson, D.F. Gleich, and J.Leskovec. Tensor spectral clustering for partitioning higher-order network structures. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 118–126, 2015.

[26] A.R. Benson, J.D. Lee, B. Rajwa, and D.F. Gleich. Scalable methods for nonnegative matrix factorizations of near-separable tall-and-skinny matrices. In *Proceedings of Neural Information Processing Systems (NIPS)*, pages 945–953, 2014.

[27] D. Bini. Tensor and border rank of certain classes of matrices and the fast evaluation of determinant inverse matrix and eigenvalues. *Calcolo*, 22(1):209–228, 1985.

[28] M. Bolten, K. Kahl, D. Kressner, and S. Macedo, F. andSokolović. Multigrid methods combined with low-rank approximation for tensor structured Markov chains. *ArXiv e-prints*, 2016.

[29] N. Boumal and P.-A. Absil. RTRMC: A Riemannian trust-region method for low-rank matrix completion. In J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24 (NIPS)*, pages 406–414. MIT, 2011.

[30] M. Boussé, O. Debals, and L. De Lathauwer. A tensor-based method for large-scale blind source separation using segmentation. Technical Report Tech. Report 15-59, ESAT-STADIUS, KU Leuven, Leuven, Belgium, 2015. submitted.

[31] S. 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*, 3(1):1–122, 2011.

[32] A. Bruckstein, D. Donoho, and M. Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Review*, 51(1):34–81, 2009.

[33] C. Caiafa and A. Cichocki. Generalizing the column-row matrix decomposition to multi-way arrays. *Linear Algebra and its Applications*, 433(3):557–573, 2010.

[34] C. Caiafa and A Cichocki. Computing sparse representations of multidimensional signals using Kronecker bases. *Neural Computaion*, 25(1):186–220, 2013.

[35] C. Caiafa and A. Cichocki. Stable, robust, and super–fast reconstruction of tensors using multi-way projections. *IEEE Transactions on Signal Processing*, 63(3):780–793, 2015.

[36] L. Cambier and P.-A. Absil. Robust low-rank matrix completion by Riemannian optimization. (in press), 2016.

[37] E.J. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080, 2010.

[38] E.J. Candes, M.B. Wakin, and S.P. Boyd. Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier Analysis and Applications*, 14(5-6):877–905, 2008.

[39] J. Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an N-way generalization of 'Eckart-Young' decomposition. *Psychometrika*, 35(3):283–319, 1970.

[40] V. Cevher, S. Becker, and M. Schmidt. Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics. *IEEE Signal Processing Magazine*, 31(5):32–43, 2014.

[41] G. Chabriel, M. Kleinsteuber, E. Moreau, H. Shen, P. Tichavsky, and A. Yeredor. Joint matrix decompositions and blind source separation: A survey of methods, identification, and applications. *IEEE Signal Processing Magazine*, 31(3):34–43, 2014.

[42] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.

[43] C. Chen, J. Huang, L. He, and H. Li. Preconditioning for accelerated iteratively reweighted least squares in structured sparsity reconstruction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[44] H. Cho, D. Venturi, and G.E. Karniadakis. Numerical methods for high-dimensional probability density function equations. *Journal of Computational Physics*, 305:817–837, 2016.

[45] J.H. Choi and S. Vishwanathan. Dfacto: Distributed factorization of tensors. In *Advances in Neural Information Processing Systems*, pages 1296–1304, 2014.

[46] D. Chu, L.-Z. Liao, M.K. Ng, and X. Zhang. Sparse canonical correlation analysis: New formulation and algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):3050–3065, 2013.

[47] W. Chu and Z. Ghahramani. Probabilistic models for incomplete multi-dimensional arrays. In *JMLR Workshop and Conference Proceedings Volume 5: AISTATS 2009*, volume 5, pages 89–96. Microtome Publishing (paper) Journal of Machine Learning Research, 2009.

[48] A. Cichocki. Tensor decompositions: New concepts for brain data analysis? *Journal of Control, Measurement, and System Integration (SICE)*, 47(7):507–517, 2011.

[49] A. Cichocki. Era of big data processing: A new approach via tensor networks and tensor decompositions, (invited). In *Proceedings of the International Workshop on Smart Info-Media Systems in Asia (SISA2013)*, September 2013.

[50] A. Cichocki. Tensor networks for big data analytics and large-scale optimization problems. *arXiv preprint arXiv:1407.3124*, 2014.

[51] A. Cichocki, D. Mandic, C. Caiafa, A.H. Phan, G. Zhou, Q. Zhao, and L. De Lathauwer. Tensor decompositions for signal processing applications: From two-way to multiway component analysis. *IEEE Signal Processing Magazine*, 32(2):145–163, 2015.

[52] A. Cichocki and A. H. Phan. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E92-A(3):708–721, 2009.

[53] A. Cichocki, R Zdunek, A.-H. Phan, and S. Amari. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. Wiley, Chichester, 2009.

[54] N. Cohen, O. Sharir, and A. Shashua. On the expressive power of deep learning: A tensor analysis. *ArXiv e-prints*, September 2015.

[55] N. Cohen and A. Shashua. Convolutional Rectifier Networks as Generalized Tensor Decompositions. *ArXiv e-prints*, 2016.

[56] P. Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3):287–314, 1994.

[57] P. Comon, G. Golub, L.H. Lim, and B. Mourrain. Symmetric tensors and symmetric tensor rank. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1254–1279, 2008.

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

[59] P. Comon, X. Luciani, and A. L. F. de Almeida. Tensor decompositions, alternating least squares and other tales. *Journal of Chemometrics*, 23:393–405, 2009.

[60] P.G. Constantine and D.F. Gleich. Tall and skinny QR factorizations in MapReduce architectures. In *Proceedings of the Second International Workshop on MapReduce and its Applications*, pages 43–50. ACM, 2011.

[61] P.G. Constantine, D.F Gleich, Y. Hou, and J. Templeton. Model reduction with MapReduce-enabled tall and skinny singular value decomposition. *SIAM Journal on Scientific Computing*, 36(5):S166–S191, 2014.

[62] E. Corona, A. Rahimian, and D. Zorin. A Tensor-Train accelerated solver for integral equations in complex geometries. *ArXiv e-prints*, November 2015.

[63] C. Crainiceanu, B. Caffo, S. Luo, V. Zipunnikov, and N. Punjabi. Population value decomposition, a framework for the analysis of image populations. *Journal of the American Statistical Association, with Discussion and Rejoinder*, 106(495):775–790, 2011.

[64] A. Critch and J. Morton. Algebraic geometry of matrix product states. *ArXiv e-prints*, 2014.

[65] A.J. Critch. *Algebraic Geometry of Hidden Markov and Related Models*. PhD thesis, University of California, Berkeley, 2013.

[66] J.P. Cunningham and Z. Ghahramani. Linear dimensionality reduction: Survey, insights, and generalizations. *Journal of Machine Learning Research*, 16:2859–2900, 2015.

[67] O. Curtef, G. Dirr, and U. Helmke. Riemannian optimization on tensor products of Grassmann manifolds: Applications to generalized Rayleigh-quotients. *SIAM Journal on Matrix Analysis and Applications*, 33(1):210–234, 2012.

[68] A. d'Aspremont, L. El Ghaoui, M.I. Jordan, and G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. *SIAM Review*, 49(3):434–448, 2007.

[69] A. De Almeida, G. Favier, J. Paulo C.L. Costa, and J. Mota. Overview of tensor decompositions with applications to communications. *Signals and Images: Advances and Results in Speech, Estimation, Compression, Recognition, Filtering, and Processing*, pages 325–356, 2016.

[70] F. De la Torre. A least-squares framework for component analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 34(6):1041–1055, 2012.

[71] L. De Lathauwer. A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization. *SIAM Journal on Matrix Analysis and Applications*, 28:642–666, 2006.

[72] L. De Lathauwer. Decompositions of a higher-order tensor in block terms – Part I and II. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1022–1066, 2008. Special Issue on Tensor Decompositions and Applications.

[73] L. De Lathauwer, B. De Moor, and J. Vandewalle. A Multilinear Singular Value Decomposition. *SIAM Journal on Matrix Analysis Applications*, 21:1253–1278, 2000.

[74] L. De Lathauwer, B. De Moor, and J. Vandewalle. On the best rank-1 and rank-$(R_1, R_2, ..., R_N)$ approximation of higher-order tensors. *SIAM Journal of Matrix Analysis and Applications*, 21(4):1324–1342, 2000.

[75] L. De Lathauwer and D. Nion. Decompositions of a higher-order tensor in block terms – Part III: Alternating least squares algorithms. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1067–1083, 2008.

[76] De Lathauwer, L. A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization. *SIAM Journal on Matrix Analysis and Applications*, 28:642–666, 2006.

[77] W. de Launey and J. Seberry. The strong Kronecker product. *J. Comb. Theory, Ser. A*, 66(2):192–213, 1994.

[78] V. de Silva and L-H. Lim. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM Journal on Matrix Analysis and Applications*, 30:1084–1127, 2008.

[79] O. Debals, M. Van Barel, and L. De Lathauwer. Löwner-based blind signal separation of rational functions with applications. *IEEE Transactions on Signal Processing*, 64(8):1909–1918, 2016.

[80] O. Debals and L. Lathauwer. *Latent Variable Analysis and Signal Separation: 12th International Conference, LVA/ICA 2015, Liberec, Czech Republic, August 25-28, 2015, Proceedings*, chapter Stochastic and Deterministic Tensorization for Blind Signal Separation, pages 3–13. Springer International Publishing, Cham, 2015.

[81] I.S. Dhillon. Fast Newton-type methods for nonnegative matrix and tensor approximation. Future Directions in Tensor-Based Computation and Modeling, 2009.

[82] E. Di Napoli, D. Fabregat-Traver, G. Quintana-Ortí, and P. Bientinesi. Towards an efficient use of the blas library for multilinear tensor contractions. *Applied Mathematics and Computation*, 235:454–468, 2014.

[83] S.V. Dolgov. TT-GMRES: Solution to a linear system in the structured tensor format. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 28(2):149–172, 2013.

[84] S.V. Dolgov. *Tensor Product Methods in Numerical Simulation of High-dimensional Dynamical Problems*. PhD thesis, Faculty of Mathematics and Informatics, University Leipzig, Germany, Leipzig, Germany, 2014.

[85] S.V. Dolgov and B.N. Khoromskij. Two-level QTT-Tucker format for optimized tensor calculus. *SIAM Journal on Matrix Analysis and Applications*, 34(2):593–623, 2013.

[86] S.V. Dolgov and B.N. Khoromskij. Simultaneous state-time approximation of the chemical master equation using tensor product formats. *Numerical Linear Algebra with Applications*, 22(2):197–219, 2015.

[87] S.V. Dolgov, B.N. Khoromskij, I.V. Oseledets, and D.V. Savostyanov. Computation of extreme eigenvalues in higher dimensions using block tensor train format. *Computer Physics Communications*, 185(4):1207–1216, 2014.

[88] S.V. Dolgov and I.V. Oseledets. Solution of linear systems and matrix inversion in the TT-format. *SIAM Journal on Scientific Computing*, 34(5):A2718–A2739, 2011.

[89] S.V. Dolgov, J.W. Pearson, D.V. Savostyanov, and M. Stoll. Fast tensor product solvers for optimization problems with fractional differential equations as constraints. *Applied Mathematics and Computation*, 273:604–623, 2016.

[90] S.V. Dolgov and D.V. Savostyanov. Alternating minimal energy methods for linear systems in higher dimensions. *SIAM Journal on Scientific Computing*, 36(5):A2248–A2271, 2014.

[91] P. Drineas and M.W. Mahoney. A randomized algorithm for a tensor-based generalization of the singular value decomposition. *Linear Algebra and its Applications*, 420(2):553–571, 2007.

[92] A. El Alaoui, X. Cheng, A. Ramdas, M. J. Wainwright, and M. I. Jordan. Asymptotic behavior of ℓ_p-based Laplacian regularization in semi-supervised learning. *ArXiv e-prints*, March 2016.

[93] M Espig, M Schuster, A Killaitis, N Waldren, P Wähnert, S Handschuh, and H Auer. TensorCalculus library, 2012.

[94] F. Esposito, T. Scarabino, A. Hyvärinen, J. Himberg, E. Formisano, S. Comani, G. Tedeschi, R. Goebel, E. Seifritz, and F. Di Salle. Independent component analysis of fMRI group studies by self-organizing clustering. *Neuroimage*, 25(1):193–205, 2005.

[95] G. Evenbly and G. Vidal. Algorithms for entanglement renormalization. *Physical Review B*, 79(14):144108, 2009.

[96] Hadi Fanaee-T and João Gama. Tensor-based anomaly detection: An interdisciplinary survey. *Knowledge-Based Systems*, 2016.

[97] G. Favier and A. de Almeida. Overview of constrained PARAFAC models. *EURASIP Journal on Advances in Signal Processing*, 2014(1):1–25, 2014.

[98] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical Lasso. *Biostatistics*, 9(3):432–441, 2008.

[99] K. Fukumizu, F.R. Bach, and M.I. Jordan. Dimensionality reduction for supervised learning with reproducing kernel Hilbert spaces. *Journal of Machine Learning Research*, 5:73–99, 2004.

[100] S. Garreis and M. Ulbrich. Constrained optimization with low-rank tensors and applications to parametric problems with PDEs. Technical report, Technical Report 5301, Optimization Online, 2016.

[101] V. Giovannetti, S. Montangero, and R. Fazio. Quantum multiscale entanglement renormalization ansatz channels. *Physical Review Letters*, 101(18):180503, 2008.

[102] D.F. Gleich, L-H. Lim, and Y. Yu. Multilinear PageRank. *SIAM Journal on Matrix Analysis and Applications*, 36(4):1507–1541, 2015.

[103] S.A. Goreinov, E.E. Tyrtyshnikov, and N.L. Zamarashkin. A theory of pseudo-skeleton approximations. *Linear Algebra and its Applications*, 261:1–21, 1997.

[104] S.A. Goreinov, N.L. Zamarashkin, and E.E. Tyrtyshnikov. Pseudo-skeleton approximations by matrices of maximum volume. *Mathematical Notes*, 62(4):515–519, 1997.

[105] J. Goulart, M. Boizard, R. Boyer, G. Favier, and P. Comon. Tensor CP decomposition with structured factor matrices: Algorithms and performance. *IEEE Journal of Selected Topics in Signal Processing*, 10(4):757–769, 2016.

[106] L. Grasedyck. Hierarchical singular value decomposition of tensors. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2029–2054, 2010.

[107] L. Grasedyck and W. Hackbusch. An introduction to hierarchical (h-) rank and TT-rank of tensors with examples. *Computational Methods in Applied Mathematics*, 11(3):291–304, 2011.

[108] L. Grasedyck, D Kessner, and C. Tobler. A literature survey of low-rank tensor approximation techniques. *CGAMM-Mitteilungen*, 36:53–78, 2013.

[109] L. Grasedyck, M. Kluge, and S. Krämer. Variants of alternating least squares tensor completion in the tensor train format. *SIAM Journal on Scientific Computing*, 37(5):A2424–A2450, 2015.

[110] A.R. Groves, C.F. Beckmann, S.M. Smith, and M.W. Woolrich. Linked independent component analysis for multimodal data fusion. *NeuroImage*, 54(1):2198 – 21217, 2011.

[111] M. Haardt, F. Roemer, and G. Del Galdo. Higher-order SVD based subspace estimation to improve the parameter estimation accuracy in multidimensional harmonic retrieval problems. *IEEE Transactions on Signal Processing*, 56:3198 – 3213, July 2008.

[112] W. Hackbusch. *Tensor Spaces and Numerical Tensor Calculus*, volume 42 of *Springer series in Computational Mathematics*. Springer, Heidelberg, 2012.

[113] W. Hackbusch and S. Kühn. A new scheme for the tensor representation. *Journal of Fourier Analysis and Applications*, 15(5):706–722, 2009.

[114] W. Hackbusch and A. Uschmajew. On the interconnection between the higher-order singular values of real tensors. *(preprint)*, 2015. INS Preprint No. 1520. Revised version, Apr. 2016.

[115] N. Halko, P. Martinsson, and J. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

[116] S. Handschuh. Changing the topology of tensor networks. *ArXiv e-prints*, 2012.

[117] S. Handschuh. *Numerical Methods in Tensor Networks*. PhD thesis, Facualty of Mathematics and Informatics, University Leipzig, Germany, Leipzig, Germany, 2015.

[118] M. Harandi, R. Hartley, C. Shen, B. Lovell, and C. Sanderson. Extrinsic methods for coding and dictionary learning on Grassmann manifolds. *International Journal of Computer Vision*, 114(2):113–136, 2015.

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

[120] F.L. Hitchcock. Multiple invariants and generalized rank of a p-way matrix or tensor. *Journal of Mathematics and Physics*, 7:39–79, 1927.

[121] J. Ho, Y. Xie, and B. Vemuri. On a nonlinear generalization of sparse coding and dictionary learning. *J. Mach. Learn. Res.: Workshop Conference Proc.*, 28(3):1480–1488, 2013.

[122] S. Holtz, T. Rohwedder, and R. Schneider. The alternating linear scheme for tensor optimization in the tensor train format. *SIAM Journal on Scientific Computing*, 34(2), 2012.

[123] S. Holtz, T. Rohwedder, and R. Schneider. On manifolds of tensors of fixed TT-rank. *Numerische Mathematik*, 120(4):701–731, 2012.

[124] M. Hong, M. Razaviyayn, Z.Q. Luo, and J.S. Pang. A unified algorithmic framework for block-structured optimization involving big data with applications in machine learning and signal processing. *IEEE Signal Processing Magazine*, 33(1):57–77, 2016.

[125] H. Huang, C. Ding, D. Luo, and T. Li. Simultaneous tensor subspace selection and clustering: The equivalence of high order SVD and K-means clustering. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge Discovery and Data mining*, pages 327–335. ACM, 2008.

[126] J.Z. Huang, H. Shen, and A. Buja. The analysis of two-way functional data using two-way regularized singular value decompositions. *Journal of the American Statistical Association*, 104(488):1609–1620, 2009.

[127] R Hübener, V Nebendahl, and W Dür. Concatenated tensor network states. *New Journal of Physics*, 12(2):025004, 2010.

[128] T. Huckle, K. Waldherr, and T. Schulte-Herbriggen. Computations in quantum tensor networks. *Linear Algebra and its Applications*, 438(2):750 – 781, 2013.

[129] A. Hyvärinen. Independent component analysis: Recent advances. *Philosophical Transactions of the Royal Society A*, 371(1984):20110534, 2013.

[130] I. Jeon, E. Papalexakis, U. Kang, and C. Faloutsos. Haten2: Billion-scale tensor decompositions. In *Proceedings of the 31st IEEE International Conference on Data Engineering (ICDE)*, pages 1047–1058. IEEE, 2015.

[131] I. Jeon, E.E. Papalexakis, C. Faloutsos, L. Sael, and U. Kang. Mining billion-scale tensors: Algorithms and discoveries. *The VLDB Journal*, pages 1–26, 2016.

[132] B. Jiang, F. Yang, and S. Zhang. Tensor and its Tucker core: The invariance relationships. *ArXiv e-prints*, January 2016.

[133] M.H. Kamal, B. Heshmat, R. Raskar, P. Vandergheynst, and G. Wetzstein. Tensor low-rank and sparse light field photography. *Computer Vision and Image Understanding*, 145:172–181, 2016.

[134] U. Kang, E.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 (KDD '12)*, pages 316–324, August 2012.

[135] Y-J. Kao, Y. Hsieh, and P. Chen. Uni10: An open-source library for tensor network algorithms. In *Journal of Physics: Conference Series*, volume 640, page 012040. IOP Publishing, 2015.

[136] L. Karlsson and A. Kressner, D.and Uschmajew. Parallel algorithms for tensor completion in the CP format. *Parallel Computing*, (in print), 2015.

[137] J.-P. Kauppi, J. Hahne, K.R. Müller, and A. Hyvärinen. Three-way analysis of spectrospatial electromyography data: Classification and interpretation. *PloS One*, 10(6):e0127231, 2015.

[138] V.A. Kazeev, M. Khammash, M. Nip, and C. Schwab. Direct solution of the chemical master equation using quantized tensor trains. *PLoS Computational Biology*, 10(3):e1003359, 2014.

[139] V.A. Kazeev and B.N. Khoromskij. Low-rank explicit QTT representation of the Laplace operator and its inverse. *SIAM Journal on Matrix Analysis and Applications*, 33(3):742–758, 2012.

[140] V.A. Kazeev, B.N. Khoromskij, and E.E. Tyrtyshnikov. Multilevel Toeplitz matrices generated by tensor-structured vectors and convolution with logarithmic complexity. *SIAM Journal on Scientific Computing*, 35(3):A1511–A1536, 2013.

[141] V.A. Kazeev, O. Reichmann, and C. Schwab. Low-rank tensor structure of linear diffusion operators in the TT and QTT formats. *Linear Algebra and its Applications*, 438(11):4204–4221, 2013.

[142] V. Khoromskaia and B.N. Khoromskij. Fast tensor method for summation of long-range potentials on 3D lattices with defects. *Numerical Linear Algebra with Applications*, 23(2):249–271, 2016.

[143] B.N. Khoromskij. $O(d \log N)$-quantics approximation of N-d tensors in high-dimensional numerical modeling. *Constructive Approximation*, 34(2):257–280, 2011.

[144] B.N. Khoromskij. Tensors-structured numerical methods in scientific computing: Survey on recent advances. *Chemometrics and Intelligent Laboratory Systems*, 110(1):1–19, 2011.

[145] B.N. Khoromskij and S. Miao. Superfast wavelet transform using quantics-TT approximation. I. application to Haar wavelets. *Computational Methods in Applied Mathematics*, 14(4):537–553, 2014.

[146] B.N. Khoromskij and A. Veit. Efficient computation of highly oscillatory integrals by using QTT tensor approximation. *Computational Methods in Applied Mathematics*, 16(1):145–159, 2016.

[147] H.-J. Kim, E. Ollila, V. Koivunen, and H.V. Poor. Robust iteratively reweighted Lasso for sparse tensor factorizations. In *IEEE Workshop on Statistical Signal Processing (SSP)*, pages 420–423, 2014.

[148] H.-J. Kim, E.sa Ollila, and V. Koivunen. New robust Lasso method based on ranks. In *23rd European Signal Processing Conference (EUSIPCO)*, pages 699–703. IEEE, 2015.

[149] T.K. Kim and R. Cipolla. Canonical correlation analysis of video volume tensors for action categorization and detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(8):1415–1428, 2009.

[150] T.K. Kim, S.F. Wong, and R. Cipolla. Tensor canonical correlation analysis for action classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07)*, pages 1–8, 2007.

[151] S. Klus and C. Schütte. Towards tensor-based methods for the numerical approximation of the Perron-Frobenius and Koopman operator. *ArXiv e-prints*, December 2015.

[152] E. Kokiopoulou, J. Chen, and Y. Saad. Trace optimization and eigenproblems in dimension reduction methods. *Numerical Linear Algebra with Applications*, 18(3):565–602, 2011.

[153] T.G. Kolda and B.W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[154] D. Kressner and F. Macedo. Low-rank tensor methods for communicating Markov processes. In *Quantitative Evaluation of Systems*, pages 25–40. Springer, 2014.

[155] D. Kressner, M. Steinlechner, and A. Uschmajew. Low-rank tensor methods with subspace correction for symmetric eigenvalue problems. *SIAM Journal on Scientific Computing*, 36(5):A2346–A2368, 2014.

[156] D. Kressner, M. Steinlechner, and B. Vandereycken. Low-rank tensor completion by Riemannian optimization. *BIT Numerical Mathematics*, 54(2):447–468, 2014.

[157] D. Kressner, M. Steinlechner, and B. Vandereycken. Preconditioned low-rank Riemannian optimization for linear systems with tensor product structure. arXiv:1508.02988, 2015.

[158] D. Kressner and C. Tobler. Algorithm 941: HTucker–A MATLAB toolbox for tensors in hierarchical Tucker format. *ACM Transactions on Mathematical Software*, 40(3):22, 2014.

[159] D. Kressner and A. Uschmajew. On low-rank approximability of solutions to high-dimensional operator equations and eigenvalue problems. *Linear Algebra and its Applications*, 493:556–572, 2016.

[160] P.M. Kroonenberg. *Applied Multiway Data Analysis*. John Wiley & Sons Ltd, New York, 2008.

[161] V. Kuleshov, A.T. Chaganty, and P. Liang. Tensor factorization via matrix factorization. *arXiv preprint arXiv:1501.07320*, 2015.

[162] 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*, 32(4):1451–1474, 2011.

[163] L. De Lathauwer. Tensor methods and blind source separation. Technical Report Tech. Report 16-09, ESAT-STADIUS, KU Leuven, Leuven, Belgium, Jan. 2016.

[164] M. Lee, H. Shen, J.Z. Huang, and J.S. Marron. Biclustering via sparse singular value decomposition. *Biometrics*, 66(4):1087–1095, 2010.

[165] N. Lee and A. Cichocki. Big data matrix singular value decomposition based on low-rank tensor train decomposition. In *Advances in Neural Networks*, pages 121–130. Springer, 2014.

[166] N. Lee and A. Cichocki. Very large-scale singular value decomposition using tensor train networks. *arXiv preprint arXiv:1410.6895*, 2014.

[167] N. Lee and A. Cichocki. Estimating a few extreme singular values and vectors for large-scale matrices in Tensor Train format. *SIAM Journal on Matrix Analysis and Applications*, 36(3):994–1014, 2015.

[168] N. Lee and A. Cichocki. Decompositions for higher order regression with Lasso penalties. In *Workshop on Tensor Decompositions and Applications (TDA2016)*, 2016.

[169] N. Lee and A. Cichocki. Fundamental tensor operations for large-scale data analysis in tensor train formats. *ArXiv e-prints*, May 2016.

[170] N. Lee and A. Cichocki. Regularized computation of approximate pseudoinverse of large matrices using low-rank tensor train decompositions. *SIAM J. Matrix Analysis and Applications*, 37(2):598–623, 2016.

[171] J. Li, C. Battaglino, I. Perros, J. Sun, and R. Vuduc. An input-adaptive and in-place approach to dense tensor-times-matrix multiply. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, page 76. ACM, 2015.

[172] M. Li and V. Monga. Robust video hashing via multilinear subspace projections. *IEEE Transactions on Image Processing*, 21(10):4397–4409, 2012.

[173] S. Liao, T. Vejchodskỳ, and R. Erban. Tensor methods for parameter estimation and bifurcation analysis of stochastic reaction networks. *Journal of The Royal Society Interface*, 12(108):20150233, 2015.

[174] A.P. Liavas and N.D. Sidiropoulos. Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers. *IEEE Transactions on Signal Processing*, 63(20):5450–5463, 2015.

[175] L.H. Lim and P. Comon. Multiarray signal processing: Tensor decomposition meets compressed sensing. *Comptes Rendus Mecanique*, 338(6):311–320, 2010.

[176] M.S. Litsarev and I.V. Oseledets. A low-rank approach to the computation of path integrals. *Journal of Computational Physics*, 305:557–574, 2016.

[177] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos. A survey of multilinear subspace learning for tensor data. *Pattern Recognition*, 44(7):1540–1551, July 2011.

[178] M. Lubasch, J.I. Cirac, and M-C. Bauls. Unifying projected entangled pair state contractions. *New Journal of Physics*, 16(3):033014, 2014.

[179] C. Lubich and I.V. Oseledets. A projector-splitting integrator for dynamical low-rank approximation. *BIT Numerical Mathematics*, 54(1):171–188, 2014.

[180] C. Lubich, I.V. Oseledets, and B. Vandereycken. Time integration of tensor trains. *ArXiv e-prints*, 2014.

[181] C. Lubich, T. Rohwedder, R. Schneider, and B. Vandereycken. Dynamical approximation of hierarchical Tucker and tensor-train tensors. *SIAM Journal on Matrix Analysis and Applications*, 34(2):470–494, 2013.

[182] Y.M. Lui, J.R. Beveridge, and M. Kirby. Action classification on product manifolds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 833–839, 2010.

[183] Y. Luo, D. Tao, K. Ramamohanarao, C. Xu, and Y. Wen. Tensor canonical correlation analysis for multi-view dimension reduction. *IEEE Transactions on Knowledge and Data Engineering*, 27(11):3111–3124, 2015.

[184] M.W. Mahoney and P. Drineas. CUR matrix decompositions for improved data analysis. *Proceedings of the National Academy of Science*, 106:697–702, 2009.

[185] M.W. Mahoney, M. Maggioni, and P. Drineas. Tensor-CUR decompositions for tensor-based data. *SIAM Journal on Matrix Analysis and Applications*, 30(3):957–987, 2008.

[186] F. Marvasti, A. Amini, F. Haddadi, M. Soltanolkotabi, B. Khalaj, A. Aldroubi, and J. Sanei, S.and Chambers. A unified approach to sparse signal processing. *EURASIP Journal on Advances in Signal Processing*, 2012(1):44, 2012.

[187] G. Meyer. *Geometric optimization algorithms for linear regression on fixed-rank matrices*. PhD thesis, Faculty of Applied Sciences, University of Liège, Belgium, June 2011.

[188] A.Y. Mikhalev and I.V. Oseledets. Iterative representing set selection for nested cross–approximation. *Numerical Linear Algebra with Applications*, 2015.

[189] E. Miles Stoudenmire and D. J. Schwab. Supervised Learning with Quantum-Inspired Tensor Networks. *ArXiv e-prints*, 2016.

[190] L. Mirsky. Symmetric gauge functions and unitarily invariant norms. *The Quarterly Journal of Mathematics*, 11:50–59, 1960.

[191] B. Mishra, G. Meyer, F. Bach, and R. Sepulchre. Low-rank optimization with trace norm penalty. *SIAM Journal on Optimization*, 23(4):2124–2149, 2013.

[192] J. Morton. Tensor networks in algebraic geometry and statistics. *Lecture at Networking Tensor Networks, Centro de Ciencias de Benasque Pedro Pascual, Benasque, Spain*, 2012.

[193] M. Mørup. Applications of tensor (multiway array) factorizations and decompositions in data mining. *Wiley Interdisciplinary Review: Data Mining and Knowledge Discovery*, 1(1):24–40, 2011.

[194] V. Murg, F. Verstraete, R. Schneider, P.R. Nagy, and O. Legeza. Tree tensor network state with variable tensor order: An efficient multireference method for strongly correlated systems. *Journal of Chemical Theory and Computation*, 11(3):1027–1036, 2015.

[195] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[196] Y. Nesterov. Subgradient methods for huge-scale optimization problems. *Mathematical Programming*, 146(1-2):275–297, 2014.

[197] M.an Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016.

[198] A. Novikov and R.A. Rodomanov. Putting MRFs on a tensor train. In *Proceedings of the International Conference on Machine Learning (ICML '14)*, 2014.

[199] J. Ogutu and H. Piepho. Regularized group regression methods for genomic prediction: Bridge, MCP, SCAD, Group Bridge, Group Lasso, Sparse Group Lasso, Group MCP and Group SCAD. In *BMC Proceedings*, volume 8, page S7. BioMed Central Ltd, 2014.

[200] A.C. Olivieri. Analytical advantages of multivariate data processing: One, two, three, infinity? *Analytical Chemistry*, 80(15):5713–5720, 2008.

[201] R. Orús. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of Physics*, 349:117–158, 2014.

[202] I.V. Oseledets. Approximation of $2^d \times 2^d$ matrices using tensor decomposition. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2130–2145, 2010.

[203] I.V. Oseledets. DMRG approach to fast linear algebra in the TT-format. *Computational Methods in Applied Mathematics*, 11(3):382–393, 2011.

[204] I.V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[205] I.V. Oseledets. Constructive representation of functions in low-rank tensor formats. *Constructive Approximation*, 37(1):1–18, 2012.

[206] I.V. Oseledets and S.V. Dolgov. Solution of linear systems and matrix inversion in the TT-format. *SIAM Journal on Scientific Computing*, 34(5):A2718–A2739, 2012.

[207] I.V. Oseledets, S.V. Dolgov, V.A. Kazeev, D. Savostyanov, O. Lebedeva, P. Zhlobich, T. Mach, and L. Song. TT-Toolbox, 2012. https://github.com/oseledets/TT-Toolbox.

[208] I.V. Oseledets, D.V. Savostianov, and E.E. Tyrtyshnikov. Tucker dimensionality reduction of three-dimensional arrays in linear time. *SIAM Journal on Matrix Analysis and Applications*, 30(3):939–956, 2008.

[209] I.V. Oseledets and E.E. Tyrtyshnikov. Breaking the curse of dimensionality, or how to use SVD in many dimensions. *SIAM Journal on Scientific Computing*, 31(5):3744–3759, 2009.

[210] I.V. Oseledets and E.E. Tyrtyshnikov. TT cross–approximation for multidimensional arrays. *Linear Algebra and its Applications*, 432(1):70–88, 2010.

[211] I.V. Oseledets and E.E. Tyrtyshnikov. Algebraic wavelet transform via quantics tensor train decomposition. *SIAM Journal on Scientific Computing*, 33(3):1315–1328, 2011.

[212] E.E. Papalexakis, N. Sidiropoulos, and R. Bro. From K-means to higher-way co-clustering: Multilinear decomposition with sparse latent factors. *IEEE Transactions on Signal Processing*, 61(2):493–506, 2013.

[213] N. Parikh and S.P. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014.

[214] X. Pennec, P. Fillard, and N. Ayache. A Riemannian framework for tensor computing. *International Journal of Computer Vision*, 66(1):41–66, 2006.

[215] D. Perez-Garcia, F. Verstraete, M.M. Wolf, and J.I. Cirac. Matrix product state representations. *Quantum Information & Computation*, 7(5):401–430, July 2007.

[216] R. Pfeifer, G. Evenbly, S. Singh, and G. Vidal. NCON: A tensor network contractor for MATLAB. *arXiv preprint arXiv:1402.0939*, 2014.

[217] A-H. Phan and A. Cichocki. Extended HALS algorithm for nonnegative Tucker decomposition and its applications for multiway analysis and classification. *Neurocomputing*, 74(11):1956–1969, 2011.

[218] A.H. Phan and A. Cichocki. Tensor decompositions for feature extraction and classification of high dimensional datasets. *Nonlinear Theory and its Applications, IEICE*, 1(1):37–68, 2010.

[219] A.H. Phan and A. Cichocki. PARAFAC algorithms for large-scale problems. *Neurocomputing*, 74(11):1970–1984, 2011.

[220] A.H. Phan, A. Cichocki, P. Tichavsky, D. Mandic, and K. Matsuoka. On revealing replicating structures in multiway data: A novel tensor decomposition approach. In *Proceedings of the 10th International Conference LVA/ICA, Tel Aviv, March 12-15*, pages 297–305. Springer, 2012.

[221] A.H. Phan, A. Cichocki, P. Tichavský, R. Zdunek, and S.R. Lehky. From basis components to complex structural patterns. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, pages 3228–3232, 2013.

[222] A.H. Phan, P. Tichavský, and A. Cichocki. TENSORBOX: MATLAB package for tensor decomposition. http://www.bsp.brain.riken.jp/∼phan/tensorbox.php, 2012.

[223] A.H. Phan, P. Tichavský, and A. Cichocki. Low complexity damped Gauss-Newton algorithms for CANDECOMP/PARAFAC. *SIAM Journal on Matrix Analysis and Applications (SIMAX)*, 34(1):126–147, 2013.

[224] Anh Huy Phan, P. Tichavský, and A. Cichocki. Low rank tensor deconvolution. In *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing, ICASSP*, pages 2169–2173, April 2015.

[225] I. Pižorn and F. Verstraete. Variational numerical renormalization group: Bridging the gap between NRG and density matrix renormalization group. *Physical Review Letters*, 108:067202, 2012.

[226] S. Ragnarsson. *Structured Tensor Computations: Blocking Symmetries and Kronecker Factorization*. Phd dissertation, Cornell University, Department of Applied Mathematics, 2012.

[227] M.V. Rakhuba and I.V. Oseledets. Fast multidimensional convolution in low-rank tensor formats via cross–approximation. *SIAM Journal on Scientific Computing*, 37(2):A565–A582, 2015.

[228] P. Richtárik and M. Takáč. Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, 156:433–484, 2016.

[229] J. Salmi, A. Richter, and V. Koivunen. Sequential unfolding SVD for tensors with applications in array signal processing. *IEEE Transactions on Signal Processing*, 57:4719–4733, 2009.

[230] D.V. Savostyanov, S.V. Dolgov, J.M. Werner, and I. Kuprov. Exact NMR simulation of protein-size spin systems using tensor train formalism. *Physical Review B*, 90(8):085139, 2014.

[231] U. Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326(1):96–192, 2011.

[232] U. Schollwöck. Matrix product state algorithms: DMRG, TEBD and relatives. In *Strongly Correlated Systems*, pages 67–98. Springer, 2013.

[233] Y-Y. Shi, L.-M. Duan, and G. Vidal. Classical simulation of quantum many-body systems with a tree tensor network. *Physical Review A*, 74(2):022320, 2006.

[234] N. Sidiropoulos, R. Bro, and G. Giannakis. Parallel factor analysis in sensor array processing. *IEEE Transactions on Signal Processing*, 48(8):2377–2388, 2000.

[235] N.D. Sidiropoulos. Generalizing Caratheodory's uniqueness of harmonic parameterization to N dimensions. *IEEE Transactions on Information Theory*, 47(4):1687–1690, 2001.

[236] N.D. Sidiropoulos. Low-rank decomposition of multi-way arrays: A signal processing perspective. In *Proceedings of the IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM 2004)*, July 2004.

[237] N.D. Sidiropoulos and R. Bro. On the uniqueness of multilinear decomposition of N-way arrays. *Journal of Chemometrics*, 14(3):229–239, 2000.

[238] C.D. Silva and F.J. Herrmann. Optimization on the Hierarchical Tucker manifold - Applications to tensor completion. *Linear Algebra and its Applications*, 481(15):131–173, 2015.

[239] A. Smilde, R. Bro, and P. Geladi. *Multi-way Analysis: Applications in the Chemical Sciences*. John Wiley & Sons Ltd, New York, 2004.

[240] S.M. Smith, A. Hyvärinen, G. Varoquaux, K.L. Miller, and C.F. Beckmann. Group-PCA for very large fMRI datasets. *NeuroImage*, 101:738–749, 2014.

[241] L. Sorber, I. Domanov, M. Van Barel, and L. De Lathauwer. Exact line and plane search for tensor optimization. *Computational Optimization and Applications*, 63(1):121–142, 2016.

[242] L. Sorber, M. Van Barel, and L. De Lathauwer. Optimization-based algorithms for tensor decompositions: Canonical Polyadic Decomposition, decomposition in rank-$(L_r, L_r, 1)$ terms and a new generalization. *SIAM Journal on Optimization*, 23(2), 2013.

[243] M. Sørensen and L. De Lathauwer. Blind signal separation via tensor decomposition with Vandermonde factor. Part I: Canonical polyadic decomposition. *IEEE Transactions on Signal Processing*, 61(22):5507–5519, 2013.

[244] M. Sørensen, L. De Lathauwer, P. Comon, S. Icart, and L. Deneire. Canonical Polyadic Decomposition with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1190–1213, 2012.

[245] M. Steinlechner. Riemannian optimization for high-dimensional tensor completion. Technical report, Technical report MATHICSE 5.2015, EPF Lausanne, Switzerland, 2015.

[246] M.M. Steinlechner. *Riemannian Optimization for Solving High-Dimensional Problems with Low-Rank Tensor Structure*. PhD thesis, École Polytechnnque Fédérale de Lausanne, 2016.

[247] E.M. Stoudenmire and S.R. White. ITensor Library Release v0.2.5. Technical report, Perimeter Institute for Theoretical Physics, May 2014.

[248] D. Sun, J.and Tao and C. Faloutsos. Beyond streams and graphs: Dynamic tensor analysis. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, pages 374–383. ACM, 2006.

[249] Y. Sun, J. Gao, X. Hong, B. Mishra, and B.Yin. Heterogeneous tensor decomposition for clustering via manifold optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(3):476–489, 2016.

[250] S. K. Suter, M. Makhynia, and R. Pajarola. TAMRESH - tensor approximation multiresolution hierarchy for interactive volume visualization. *Computer Graphics Forum*, 32(3):151–160, 2013.

[251] M. Tan, I. Tsang, and L. Wang. Matching pursuit Lasso Part I and II: Sparse recovery over big dictionary. *IEEE Transactions on Signal Processing*, 63(3):727–753, 2015.

[252] Y. Tang, R. Salakhutdinov, and G. Hinton. Tensor analyzers. In *Proceedings of the 30th International Conference on Machine Learning, (ICML 2013), Atlanta, USA*, 2013.

[253] D. Tao, X. Li, X. Wu, and S. Maybank. General tensor discriminant analysis and Gabor features for gait recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(10):1700–1715, 2007.

[254] R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[255] P. Tichavsky and A. Yeredor. Fast approximate joint diagonalization incorporating weight matrices. *IEEE Transactions on Signal Processing*, 47 (3):878–891, 2009.

[256] M.K. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 567–574, 2009.

[257] C. Tobler. *Low-rank tensor methods for linear systems and eigenvalue problems*. PhD thesis, ETH Zürich, 2012.

[258] L.N. Trefethen. Cubature, approximation, and isotropy in the hypercube. *(submitted)*, 2016.

[259] L. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[260] L.R. Tucker. The extension of factor analysis to three-dimensional matrices. In H. Gulliksen and N. Frederiksen, editors, *Contributions to Mathematical Psychology*, pages 110–127. Holt, Rinehart and Winston, New York, 1964.

[261] A. Uschmajew. Local convergence of the alternating least squares algorithm for canonical tensor approximation. *SIAM Journal on Matrix Analysis and Applications*, 33(2):639–652, 2012.

[262] A. Uschmajew, D. Kressner, and M. Steinlechner. Low-rank tensor methods with subspace correction for symmetric eigenvalue problems. *Oberwolfach Reports*, 10(4):3296–3298, 2013.

[263] A. Uschmajew and B. Vandereycken. The geometry of algorithms using hierarchical tensors. *Linear Algebra and its Applications*, 439:133—166, 2013.

[264] B. Vandereycken. *Riemannian and multilevel optimization for rank-constrained matrix problems with applications to Lyapunov equations*. PhD thesis, Faculty of Engineering, Katholieke Universiteit Leuven, B-3001 Leuven, Belgium, December 2010.

[265] N. Vannieuwenhoven, R. Vandebril, and K. Meerbergen. A new truncation strategy for the higher-order singular value decomposition. *SIAM Journal on Scientific Computing*, 34(2):A1027–A1052, 2012.

[266] M.A.O. Vasilescu and D. Terzopoulos. Multilinear analysis of image ensembles: Tensorfaces. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 2350, pages 447–460, Copenhagen, Denmark, May 2002.

[267] F. Verstraete, V. Murg, and J.I Cirac. Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems. *Advances in Physics*, 57(2):143–224, 2008.

[268] N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer. Tensorlab 3.0, Mar. 2016. Available online.

[269] 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. *IEEE Signal Processing Magazine*, 31(5):71–79, 2014.

[270] N. Vervliet and L. De Lathauwer. A randomized block sampling approach to Canonical Polyadic decomposition of large-scale tensors. *IEEE Transactions on Selected Topics Signal Processing*, 10(2):284–295, 2016.

[271] G. Vidal. Efficient classical simulation of slightly entangled quantum computations. *Physical Review Letters*, 91(14):147902, 2003.

[272] S.A. Vorobyov, Y. Rong, N.D. Sidiropoulos, and A.B. Gershman. Robust iterative fitting of multilinear models. *IEEE Transactions on Signal Processing*, 53(8):2678–2689, 2005.

[273] S. Wahls, V. Koivunen, H.V. Poor, and M. Verhaegen. Learning multidimensional Fourier series with tensor trains. In *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 394–398. IEEE, 2014.

[274] D. Wang, H. Shen, and Y. Truong. Efficient dimension reduction for high-dimensional matrix-valued data. *Neurocomputing*, 190:25–34, 2016.

[275] H. Wang and M. Thoss. Multilayer formulation of the multiconfiguration time-dependent Hartree theory. *Journal of Chemical Physics*, 119(3):1289–1299, 2003.

[276] H. Wang, Q. Wu, L. Shi, Y. Yu, and N. Ahuja. Out-of-core tensor approximation of multi-dimensional matrices of visual data. *ACM Transactions on Graphics*, 24(3):527–535, 2005.

[277] S. Wang and Z. Zhang. Improving CUR matrix decomposition and the Nyström approximation via adaptive sampling. *The Journal of Machine Learning Research*, 14(1):2729–2769, 2013.

[278] Y. Wang, H-Y. Tung, A. J Smola, and A. Anandkumar. Fast and guaranteed tensor decomposition via sketching. In *Advances in Neural Information Processing Systems*, pages 991–999, 2015.

[279] G. Wetzstein, D. Lanman, M. Hirsch, and R.Raskar. Tensor displays: Compressive light field synthesis using multilayer displays with directional backlighting. *ACM Transaction on Graphics*, 31(4):80, 2012.

[280] S.R. White. Density-matrix algorithms for quantum renormalization groups. *Physical Review B*, 48(14):10345, 1993.

[281] D.M. Witten. *A Penalized Matrix Decomposition, and its Applications*. PhD dissertation, Department of Statistics, Stanford University, 2010.

[282] D.M. Witten, R. Tibshirani, and T. Hastie. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 10(3):515–534, 2009.

[283] T. Wu, A.R. Benson, and D.F. Gleich. General tensor spectral co-clustering for higher-order data. *arXiv preprint arXiv:1603.00395*, 2016.

[284] Z. Xu, F.Yan, and A. Qi. Infinite Tucker decomposition: Nonparametric Bayesian models for multiway data analysis. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, ICML '12, pages 1023–1030. Omnipress, July 2012.

[285] Y. Yang and T. Hospedales. Deep Multi-task Representation Learning: A Tensor Factorisation Approach. *ArXiv e-prints*, 2016.

[286] T. Yokota, Q. Zhao, and A. Cichocki. Smooth PARAFAC decomposition for tensor completion. *ArXiv e-prints*, 2015.

[287] Z. Zhang, X. Yang, I.V. Oseledets, G.E. Karniadakis, and L. Daniel. Enabling high-dimensional hierarchical uncertainty quantification by ANOVA and tensor-train decomposition. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(1):63–76, 2015.

[288] H.H. Zhao, Z.Y. Xie, Q.N. Chen, Z.C. Wei, J.W. Cai, and T. Xiang. Renormalization of tensor-network states. *Physical Review B*, 81(17):174411, 2010.

[289] Q. Zhao, C. Caiafa, D.P. Mandic, Z.C. Chao, Y. Nagasaka, N. Fujii, L. Zhang, and A. Cichocki. Higher order partial least squares (HOPLS): A generalized multilinear regression method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1660–1673, 2013.

[290] Q. Zhao, C.F. Caiafa, D.P. Mandic, L. Zhang, T. Ball, A. Schulze-Bonhage, and A. Cichocki. Multilinear subspace regression: An orthogonal tensor decomposition approach. In *Advances in Neural Information Processing Systems 24(NIPS)*, pages 1269–1277. MIT, 2011.

[291] Q. Zhao, L. Zhang, and A. Cichocki. Bayesian CP factorization of incomplete tensors with automatic rank determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1751–1763, 2015.

[292] Q. Zhao, G. Zhou, T. Adali, L. Zhang, and A. Cichocki. Kernelization of tensor-based models for multiway data analysis: Processing of multidimensional structured data. *IEEE Signal Processing Magazine*, 30(4):137–148, 2013.

[293] Q. Zhao, G. Zhou, L. Zhang, A. Cichocki, and S. Amari. Bayesian robust tensor factorization for incomplete multiway data. *IEEE Transactions on Neural Networks and Learning Systems*, PP(99):1–1, 2016.

[294] S. Zhe, Y. Qi, Y. Park, Z. Xu, I. Molloy, and S.h Chari. DinTucker: Scaling up Gaussian process models on large Multidimensional arrays. In *Thirtieth AAAI Conference on Artificial Intelligence (in print)*, 2016.

[295] S. Zhe, P. Wang, K.-c. Lee, Z. Xu, J. Yang, Y. Park, and Y. Qi. Distributed flexible nonlinear tensor factorization. *ArXiv e-prints*, 2016.

[296] G Zhou and A Cichocki. Canonical Polyadic Decomposition based on a single mode blind source separation. *IEEE Signal Processing Letters*, 19(8):523–526, 2012.

[297] G. Zhou and A. Cichocki. Fast and unique Tucker decompositions via multiway blind source separation. *Bulletin of Polish Academy of Science*, 60(3):389–407, 2012.

[298] G. Zhou and A. Cichocki. TDALAB: Tensor Decomposition Laboratory. http://bsp.brain.riken.jp/TDALAB/, 2013.

[299] G Zhou, A Cichocki, and S Xie. Fast nonnegative matrix/tensor factorization based on low-rank approximation. *IEEE Transactions on Signal Processing*, 60(6):2928–2940, June 2012.

[300] G. Zhou, A. Cichocki, Y. Zhang, and D.P. Mandic. Group component analysis for multiblock data: Common and individual feature extraction. *IEEE Transactions on Neural Networks and Learning Systems*, (in print), 2016.

[301] G. Zhou, A. Cichocki, Q. Zhao, and S. Xie. Efficient nonnegative Tucker decompositions: Algorithms and uniqueness. *IEEE Transactions on Image Processing*, 24(12):4990–5003, 2015.

[302] G. Zhou, Q. Zhao, Y. Zhang, T. Adali, S. Xie, and A. Cichocki. Linked component analysis from matrices to high-order tensors: Applications to biomedical data. *Proceedings of the IEEE*, 104(2):310–331, 2016.

[303] J. Zhou, A. Bhattacharya, A.H. Herring, and D.B. Dunson. Bayesian factorizations of big sparse tensors. *Journal of the American Statistical Association*, 110(512):1562–1576, 2015.