now

the essence of knowledge

Tensor Networks for Dimensionality Reduction and Large-Scale Optimization Part 1 Low-Rank Tensor Decompositions

Andrzej Cichocki
RIKEN Brain Science Institute (BSI), Japan and
Skolkovo Institute of Science and Technology (SKOLTECH)
a.cichocki@riken.jp

Namgil Lee
RIKEN BSI, namgil.lee@riken.jp

Ivan Oseledets
Skolkovo Institute of Science and Technology (SKOLTECH) and
Institute of Numerical Mathematics of Russian Academy of Sciences
i.oseledets@skolkovotech.ru

Anh-Huy Phan
RIKEN BSI, phan@brain.riken.jp

Qibin Zhao
RIKEN BSI, qbzhao@brain.riken.jp

Danilo P. Mandic
Department of Electrical and Electronic Engineering
Imperial College London
d.mandic@imperial.ac.uk

Contents

Abstract

Modern applications in engineering and data science are increasingly based on multidimensional data of exceedingly high volume, variety, and structural richness. However, standard machine learning algorithms typically scale exponentially with data volume and complexity of cross-modal couplings - the so called curse of dimensionality - which is prohibitive to the analysis of large-scale, multi-modal and multi-relational datasets. Given that such data are often efficiently represented as multiway arrays or tensors, it is therefore timely and valuable for the multidisciplinary machine learning and data analytic communities to review low-rank tensor decompositions and tensor networks as emerging tools for dimensionality reduction and large scale optimization problems. Our particular emphasis is on elucidating that, by virtue of the underlying low-rank approximations, tensor networks have the ability to alleviate the curse of dimensionality in a number of applied areas. In Part 1 of this monograph we provide innovative solutions to low-rank tensor network decompositions and easy to interpret graphical representations of the mathematical operations on tensor networks. Such a conceptual insight allows for seamless migration of ideas from the flat-view matrices to tensor network operations and vice versa, and provides a platform for further developments, practical applications, and non-Euclidean extensions. It also permits the introduction of various tensor network operations without an explicit notion of mathematical expressions, which may be beneficial for many research communities that do not directly rely on multilinear algebra. Our focus is on the Tucker and tensor train (TT) decompositions and their extensions, and on demonstrating the ability of tensor networks to provide linearly or even super-linearly (e.g., logarithmically) scalable solutions, as illustrated in detail in Part 2 of this monograph.

1

Introduction and Motivation

This monograph aims to present a coherent account of ideas and methodologies related to tensor decompositions (TDs) and tensor networks models (TNs). Tensor decompositions (TDs) decompose complex data tensors of exceedingly high dimensionality into their factor (component) tensors and matrices, while tensor networks (TNs) decompose higher-order tensors into sparsely interconnected small-scale factor matrices and/or low-order core tensors. These low-order core tensors are called "components", "blocks", "factors" or simply "cores". In this way, large-scale data can be approximately represented in highly compressed and distributed formats.

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. Tensor networks can be thought of as special graph structures which break down high-order tensors into a set of sparsely interconnected low-order core tensors, thus allowing for both enhanced interpretation and computational advantages. Such an approach is valuable in many application contexts which require the computation of eigenvalues and the corresponding eigenvectors of extremely high-dimensional linear or nonlinear operators. These operators typically describe the coupling between many degrees of freedom within 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 exist among all

the degrees of freedom but mostly locally, whereby "relevant" information, of relatively low-dimensionality, is embedded into very large-dimensional measurements (Verstraete *et al.*, 2008; Schollwöck, 2013; Orús, 2014; Murg *et al.*, 2015).

Tensor networks offer a theoretical and computational framework for the analysis of 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 or even less entries (Oseledets and Tyrtyshnikov, 2009; Dolgov and Khoromskij, 2013; Kazeev *et al.*, 2013a, 2014; Kressner *et al.*, 2014a; Vervliet *et al.*, 2014; Dolgov and Khoromskij, 2015; Liao *et al.*, 2015; Bolten *et al.*, 2016).

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 several generic optimization problems for huge-scale structured data. Our aim is also to unify the terminology, notation, and algorithms for tensor decompositions and tensor networks which are being developed 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 volume and structural complexity of modern datasets are becoming exceedingly high, to the extent which renders standard analysis methods and algorithms inadequate. Apart from the huge Volume, the other features which characterize big data include Veracity, Variety and Velocity (see Figures 1.1(a) and (b)). Each of the "V features" represents a research challenge in its own right. 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. Some applications give rise to additional "V challenges", such as Visualization, Variability and Value. The Value feature is particularly interesting and refers to the extraction of high quality and consistent information, from which meaningful and interpretable results can be obtained.

Owing to the increasingly affordable recording devices, extreme-scale volumes and variety of data are becoming ubiquitous across the science and engineering disciplines. In the case of multimedia (speech, video), remote sensing 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. 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. Our particular focus is on 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 1.2–1.7); for example, a 3rd-order tensor (with three-modes) looks like a cube (see Figure 1.2). Subtensors are formed when a subset of tensor indices is fixed. Of particular interest are *fibers* which are vectors obtained by fixing every tensor 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 be noted that block matrices can also be represented by tensors, as illustrated in Figure 1.3 for 4th-order tensors.

(a)
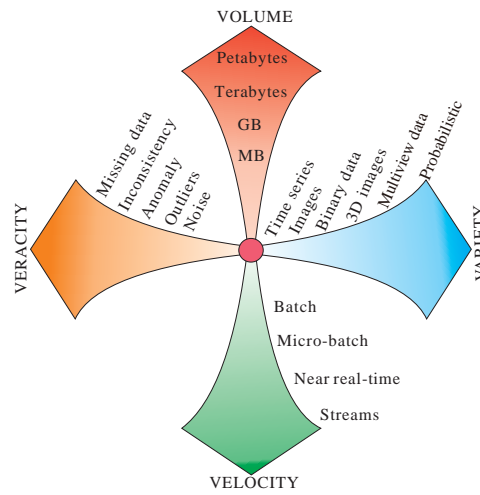
(b)
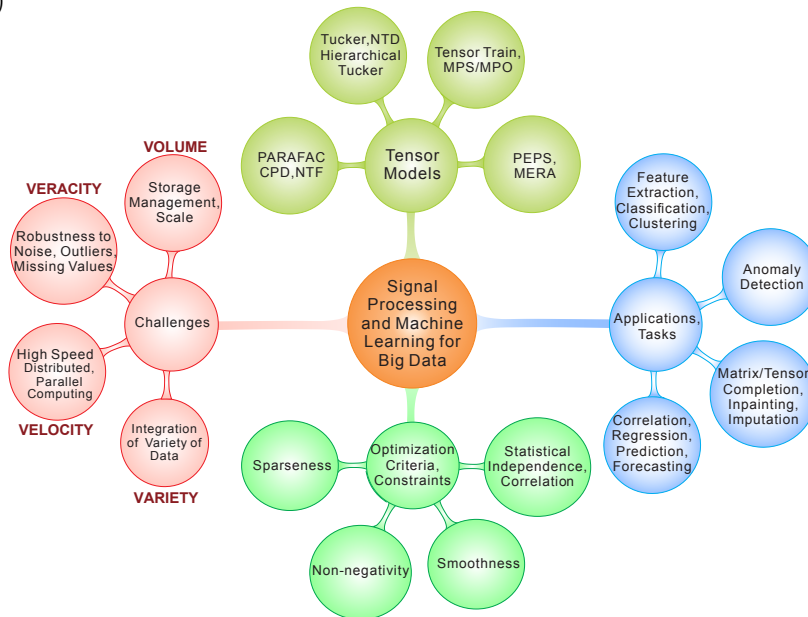
Figure 1.1: A framework for extremely large-scale data analysis. (a) The 4V challenges for big data. (b) A unified framework for the 4V challenges and the potential applications based on tensor decomposition approaches.
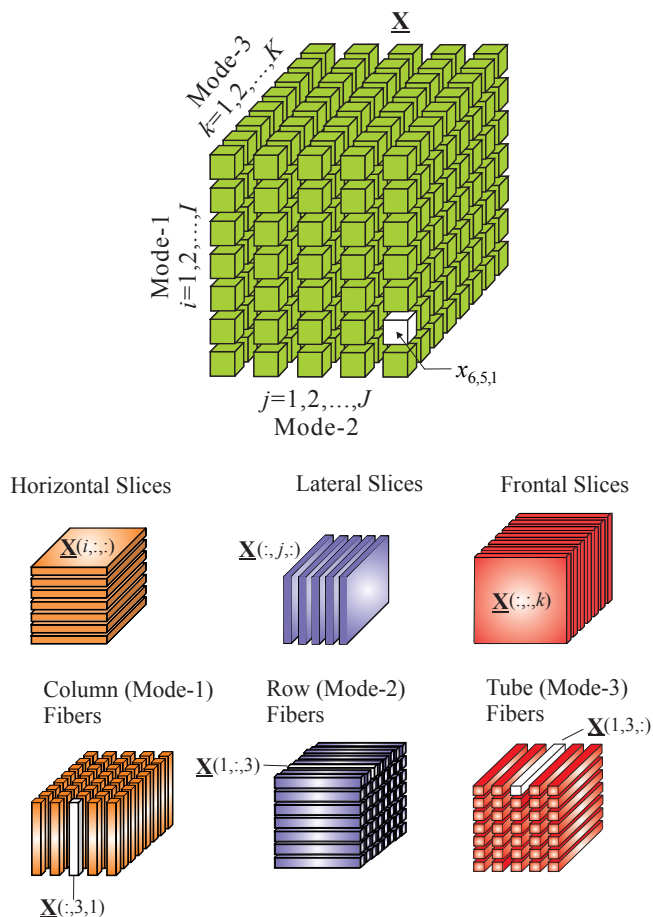
Figure 1.2: A 3rd-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$, with entries $x_{i,j,k} = \underline{\mathbf{X}}(i,j,k)$, and its subtensors: slices (middle) and fibers (bottom). All fibers are treated as column vectors.

We adopt the notation whereby 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 boldface capital letters, e.g., $\mathbf{X} \in \mathbb{R}^{I \times J}$, and vectors (1st-order tensors) by boldface lower case letters, e.g., $\mathbf{x} \in \mathbb{R}^{J}$. For example, the columns of the matrix $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_R] \in \mathbb{R}^{I \times R}$ are
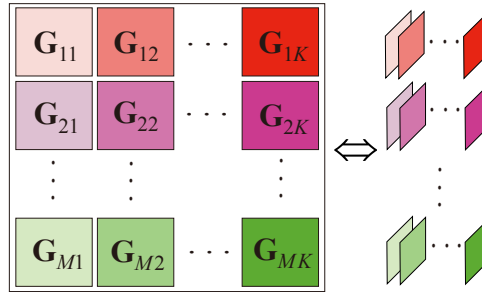
Figure 1.3: A block matrix and its representation as a 4th-order tensor, created by reshaping (or a projection) of blocks in the rows into lateral slices of 3rd-order tensors.
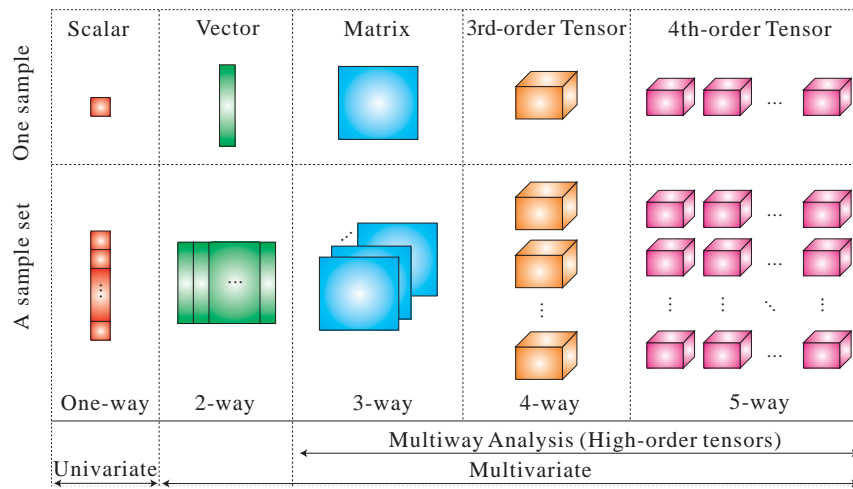
Figure 1.4: Graphical representation of multiway array (tensor) data of increasing structural complexity and "Volume" (see (Olivieri, 2008) for more detail).

the vectors denoted by $\mathbf{a}_r \in \mathbb{R}^I$, while the elements of a matrix (scalars) are denoted by lowercase letters, e.g., $a_{ir} = \mathbf{A}(i, r)$ (see Table 1.1).

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 order of a tensor is the number of its "modes", "ways" or "dimensions", which can include space, time, frequency, trials, classes, and dictionaries. The term "size" stands for the number of values that an index can take in a particular

(a)

(b)

$$\sum_{k=1}^{K} a_{i,j,k}\, b_{k,l,m,p} = c_{i,j,l,m,p}$$
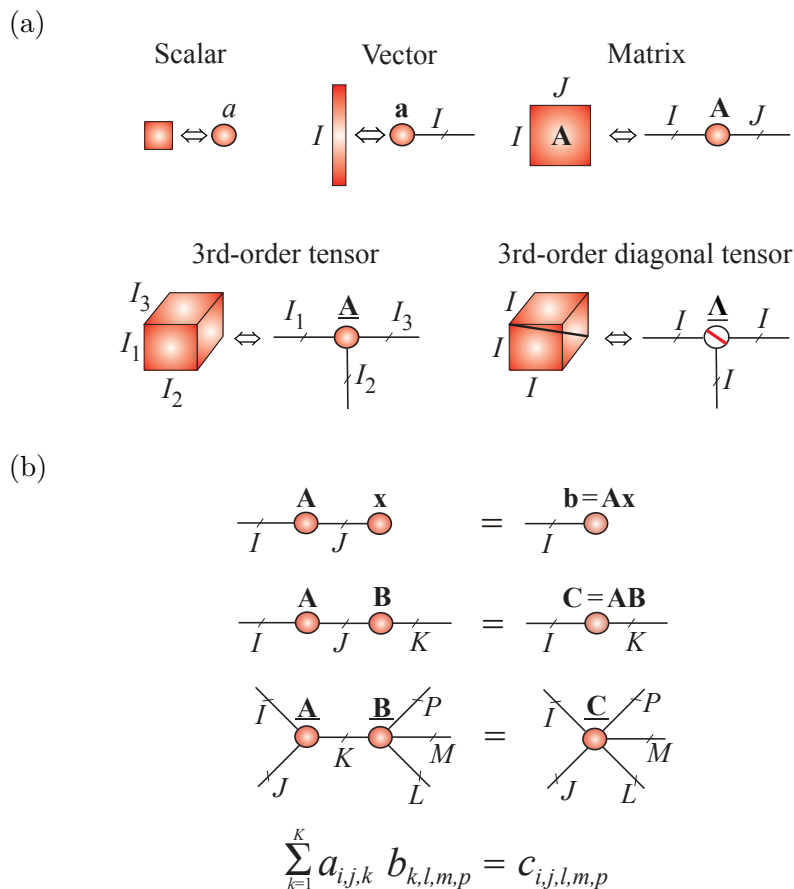
Figure 1.5: Graphical representation of tensor manipulations. (a) Basic building blocks for tensor network diagrams. (b) Tensor network diagrams for matrix-vector multiplication (top), matrix by matrix multiplication (middle) and contraction of two tensors (bottom). The order of reading of indices is anti-clockwise, from the left position.

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 all modes-n ($n = 1, 2, \ldots, N$). Lower-case letters e.g, i, j are used for the subscripts in running indices and capital letters I, J 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 $< n >$ denotes the set of indices $\{1, 2, \ldots, n\}$.

Table 1.1: 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 $I_1 \times I_2 \times \cdots \times I_N$
$x_{i_1,i_2,\ldots,i_N} = \underline{\mathbf{X}}(i_1,i_2,\ldots,i_N)$	$(i_1,i_2,\ldots,i_N)$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
$\underline{\boldsymbol{\Lambda}} \in \mathbb{R}^{R \times R \times \cdots \times R}$	Nth-order diagonal core tensor with nonzero entries λ_r on the main diagonal
$\mathbf{A}^{\mathrm{T}}$, $\mathbf{A}^{-1}$, $\mathbf{A}^{\dagger}$	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 R column vectors $\mathbf{a}_r \in \mathbb{R}^I$, with 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 $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$
$\mathbf{X}_{<n>} \in \mathbb{R}^{I_1 I_2 \cdots I_n \times I_{n+1} \cdots I_N}$	mode-$(1,\ldots,n)$ 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 (matrix) of a tensor $\underline{\mathbf{X}}$ obtained by fixing all indices but two
$\underline{\mathbf{X}}(:, :, :, i_4, \ldots, i_N)$	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
$\otimes_L$, $|\otimes|$	Left Kronecker, strong Kronecker products
$\mathbf{x} = \mathrm{vec}(\underline{\mathbf{X}})$	vectorization of $\underline{\mathbf{X}}$
$\mathrm{tr}(\bullet)$	trace of a square matrix
$\mathrm{diag}(\bullet)$	diagonal matrix

Table 1.2: Terminology used 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	site
variables	open (physical) 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 $\mathbf{x}^{\mathrm{T}} \in \mathbb{R}^{1 \times I}$	bra $\langle\Psi|$
inner product $\langle\mathbf{x}, \mathbf{x}\rangle = \mathbf{x}^{\mathrm{T}}\mathbf{x}$	$\langle\Psi|\Psi\rangle$
Tensor Train (TT)	Matrix Product State (MPS) (with Open Boundary Conditions (OBC))
Tensor Chain (TC)	MPS with Periodic Boundary Conditions (PBC)
Matrix TT	Matrix Product Operators (with OBC)
Hierarchical Tucker (HT)	Tree Tensor Network State (TTNS) with rank-3 tensors

Notations and terminology used for tensors and tensor networks differ across the scientific communities (see Table 1.2); to this end we employ a unifying notation particularly suitable for machine learning and signal processing research, which is summarized in Table 1.1.

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, in this monograph, we grossly simplify the description of tensors and their mathematical operations through diagrammatic representations borrowed from physics and quantum chemistry (see (Orús, 2014) and references therein). 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 1.5(a)). In our adopted notation, each scalar (zero-order tensor), vector (first-order tensor), matrix (2nd-order tensor), 3rd-order tensor or higher-order tensor is represented by a circle (or rectangular), 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 designates a contraction of tensors, which is a summation of products over a common index (see Figure 1.5(b) and Section 2).

Block tensors, where each entry (e.g., of a matrix or a vector) is an individual subtensor, can be represented in a similar graphical form, as illustrated in Figure 1.6. Hierarchical (multilevel block) matrices are also naturally represented by tensors and vice versa, as illustrated in Figure 1.7 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 sophisticated tensor operations. We highlight the significant advantages of such diagrammatic notations in the description of tensor manipulations, and show that most tensor operations can
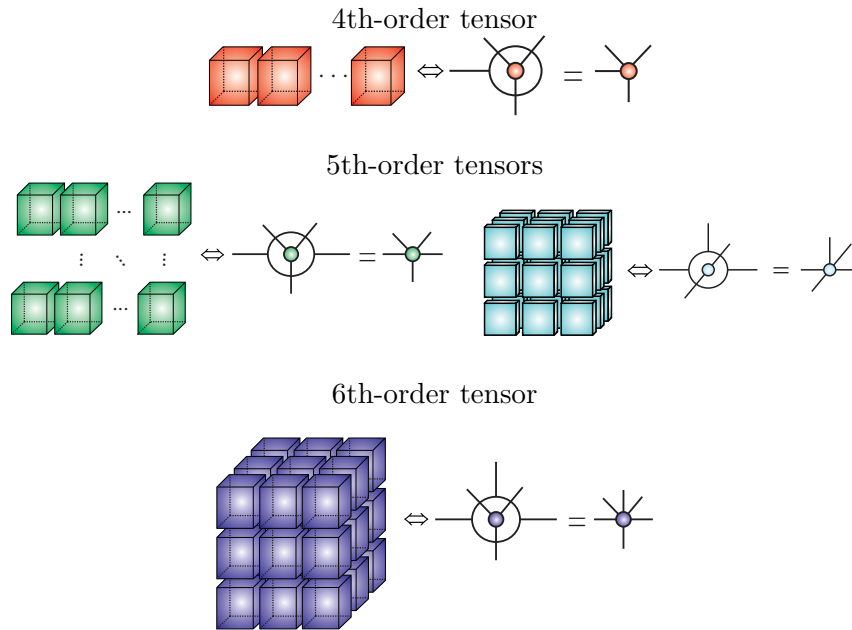
Figure 1.6: Graphical representations and symbols for higher-order block tensors. Each block represents either a 3rd-order tensor or a 2nd-order tensor. The outer circle indicates a global structure of the block tensor (e.g. a vector, a matrix, a 3rd-order block tensor), while the inner circle reflects the structure of each element within the block tensor. For example, in the top diagram a vector of 3rd order tensors is represented by an outer circle with one edge (a vector) which surrounds an inner circle with three edges (a 3rd order tensor), so that the whole structure designates a 4th-order tensor.

be visualized through 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 Bellman (1961) to indicate that the number of samples needed to estimate an arbitrary function with a given level of accuracy grows exponentially with the

(a)

(b)

(c)
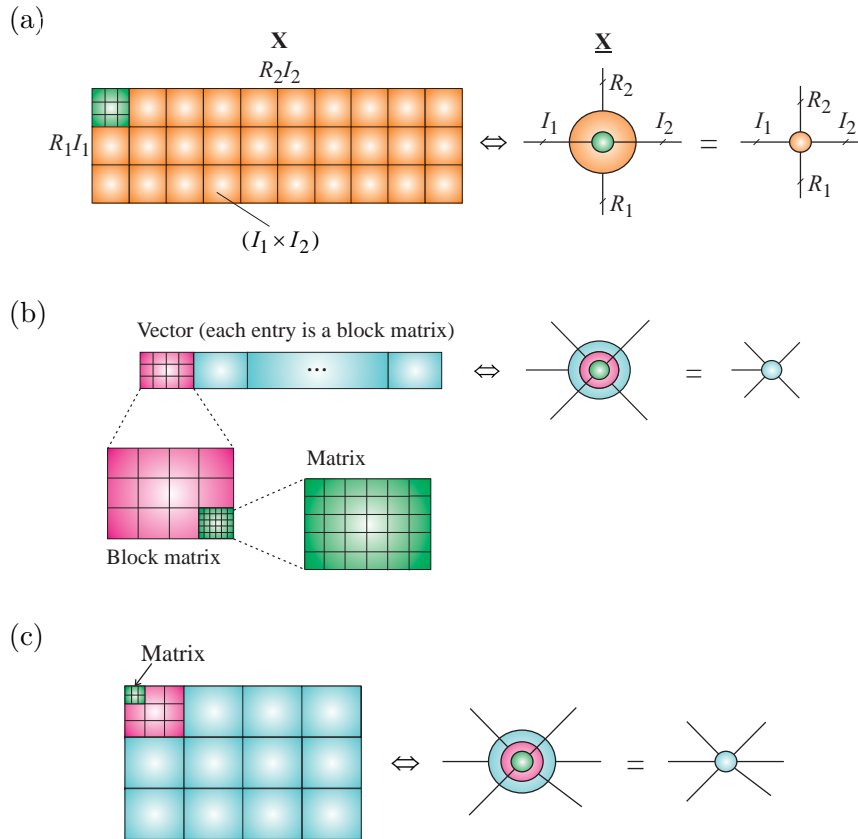
Figure 1.7: Hierarchical matrix structures and their symbolic representation as tensors. (a) A 4th-order tensor representation for a block matrix $\mathbf{X} \in \mathbb{R}^{R_1 I_1 \times R_2 I_2}$ (a matrix of matrices), which comprises block matrices $\mathbf{X}_{r_1, r_2} \in \mathbb{R}^{I_1 \times I_2}$. (b) A 5th-order tensor. (c) A 6th-order tensor.

number of variables, that is, with the dimensionality of the function. In a general context of machine learning and the underlying optimization problems, the "curse of dimensionality" may also refer to an exponentially increasing number of parameters required to describe the data/system or an extremely large number of degrees of freedom. 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 requiring enormous computational and memory resources to process such data. The understanding and handling of the inherent dependencies among the excessive degrees of freedom create both difficult to solve problems and fascinating new opportunities, but comes at a price of reduced accuracy, owing to the necessity to involve various approximations.

We show that the curse of dimensionality can be alleviated or even fully dealt with through tensor network representations; these naturally cater for the excessive volume, veracity and variety of data (see Figure 1.1) and are supported by efficient tensor decomposition algorithms which involve relatively simple mathematical operations. Another desirable aspect of tensor networks is their relatively small-scale and low-order *core tensors*, which act as "building blocks" of tensor networks. These core tensors are relatively easy to handle and visualize, and 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.

To address the curse of dimensionality, this work mostly focuses 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 (Klus and Schütte, 2015), however, distributed storage and processing of high-order tensors in their 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 software tools such as the

Tensor Toolbox (Bader and Kolda, 2015) and in the sparse grid approach (Garcke *et al.*, 2001; Bungartz and Griebel, 2004; Hackbusch, 2012).

As already mentioned, the problem of huge dimensionality can be alleviated through various distributed and compressed tensor network formats, achieved by low-rank tensor network approximations. The underpinning idea is that by employing tensor networks 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 or a very few variables (Bebendorf, 2011; Dolgov, 2014; Cho *et al.*, 2016; Trefethen, 2017). In the simplest scenario, the function $f(\mathbf{x})$ can be (approximately) represented in the following separable form

$$f(x_1, x_2, \ldots, x_N) \cong f^{(1)}(x_1) f^{(2)}(x_2) \cdots f^{(N)}(x_N). \qquad (1.1)$$

In practice, when an N-variate function $f(\mathbf{x})$ is discretized into an Nth-order array, or a tensor, the approximation in (1.1) 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 (1.1) 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, the form of separability in (1.1) is rather rare in practice.

The concept of tensor networks rests upon generalized (full or partial) separability of the variables of a high dimensional function. This can be achieved in different tensor formats, including:

- The Canonical Polyadic (CP) format (see Section 3.2), where

$$f(x_1, x_2, \ldots, x_N) \cong \sum_{r=1}^{R} f_r^{(1)}(x_1) \, f_r^{(2)}(x_2) \cdots f_r^{(N)}(x_N), \quad (1.2)$$

  in an exact analogy to (1.1). In a discretized form, the above CP format can be written as an Nth-order tensor

$$\underline{\mathbf{F}} \cong \sum_{r=1}^{R} \mathbf{f}_r^{(1)} \circ \mathbf{f}_r^{(2)} \circ \cdots \circ \mathbf{f}_r^{(N)} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}, \quad (1.3)$$

  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, given by

$$f(x_1, \ldots, x_N) \cong \sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} g_{r_1, \ldots, r_N} \, f_{r_1}^{(1)}(x_1) \cdots f_{r_N}^{(N)}(x_N),$$
$$(1.4)$$

  and its distributed tensor network variants (see Section 3.3),

- The Tensor Train (TT) format (see Section 4.1), in the form

$$f(x_1, x_2, \ldots, x_N) \cong \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_{N-1}=1}^{R_{N-1}} f_{r_1}^{(1)}(x_1) \, f_{r_1 \, r_2}^{(2)}(x_2) \cdots$$
$$\cdots f_{r_{N-2} \, r_{N-1}}^{(N-2)}(x_{N-1}) \, f_{r_{N-1}}^{(N)}(x_N), \quad (1.5)$$

  with the equivalent compact matrix representation

$$f(x_1, x_2, \ldots, x_N) \cong \mathbf{F}^{(1)}(x_1) \, \mathbf{F}^{(2)}(x_2) \cdots \mathbf{F}^{(N)}(x_N), \quad (1.6)$$

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

- The Hierarchical Tucker (HT) format (also known as the Hierarchical Tensor format) can be expressed via a hierarchy of nested

separations in the following way. Consider nested nonempty disjoint subsets u, v, and $t = u \cup v \subset \{1, 2, \ldots, N\}$, then for some $1 \leqslant N_0 < N$, with $u_0 = \{1, \ldots, N_0\}$ and $v_0 = \{N_0 + 1, \ldots, N\}$, the HT format can be expressed as

$$f(x_1, \ldots, x_N) \cong \sum_{r_{u_0}=1}^{R_{u_0}} \sum_{r_{v_0}=1}^{R_{v_0}} g_{r_{u_0}, r_{v_0}}^{(12 \cdots N)} f_{r_{u_0}}^{(u_0)}(\mathbf{x}_{u_0}) f_{r_{v_0}}^{(v_0)}(\mathbf{x}_{v_0}),$$

$$f_{r_t}^{(t)}(\mathbf{x}_t) \cong \sum_{r_u=1}^{R_u} \sum_{r_v=1}^{R_v} g_{r_u, r_v, r_t}^{(t)} f_{r_u}^{(u)}(\mathbf{x}_u) f_{r_v}^{(v)}(\mathbf{x}_v),$$

where $\mathbf{x}_t = \{x_i : i \in t\}$. See Section 2.2.1 for more detail.

Example. In a particular case for $N=4$, the HT format can be expressed by

$$f(x_1, x_2, x_3, x_4) \cong \sum_{r_{12}=1}^{R_{12}} \sum_{r_{34}=1}^{R_{34}} g_{r_{12}, r_{34}}^{(1234)} f_{r_{12}}^{(12)}(x_1, x_2) f_{r_{34}}^{(34)}(x_3, x_4),$$

$$f_{r_{12}}^{(12)}(x_1, x_2) \cong \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} g_{r_1, r_2, r_{12}}^{(12)} f_{r_1}^{(1)}(x_1) f_{r_2}^{(2)}(x_2),$$

$$f_{r_{34}}^{(34)}(x_3, x_4) \cong \sum_{r_3=1}^{R_3} \sum_{r_4=1}^{R_4} g_{r_3, r_4, r_{34}}^{(34)} f_{r_3}^{(3)}(x_3) f_{r_4}^{(4)}(x_4).$$

The Tree Tensor Network States (TTNS) format, which is an extension of the HT format, can be obtained by generalizing the two subsets, u, v, into a larger number of disjoint subsets $u_1, \ldots, u_m$, $m \geqslant 2$. In other words, the TTNS can be obtained by more flexible separations of variables through products of larger numbers of functions at each hierarchical level (see Section 2.2.1 for graphical illustrations and more detail).

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

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 cases this approach may fail to obtain sufficiently good low-rank TN approximations. The concept of generalized separability has already been explored in numerical methods for high-dimensional density function equations (Liao *et al.*, 2015; Trefethen, 2017; Cho *et al.*, 2016) and within a variety of huge-scale optimization problems (see Part 2 of this monograph).

To illustrate how tensor decompositions address excessive volumes of data, if all computations are performed on a CP tensor format in (1.3) and not on the raw Nth-order 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 the tensor order N and size I (see Table 4.4). For example, the discretization of a 5-variate function over 100 sample points on each axis would yield the difficulty to manage $100^5 = 10,000,000,000$ sample points, while a rank-2 CP representation would require only $5 \times 2 \times 100 = 1000$ sample points.

Although the CP format in (1.2) effectively bypasses the curse of dimensionality, the CP approximation may involve numerical problems for very high-order tensors, which in addition to the intrinsic uncloseness of the CP format (i.e., difficulty to arrive at a canonical format), the corresponding algorithms for CP decompositions are often ill-posed (de Silva and Lim, 2008). As a remedy, greedy approaches may be considered which, for enhanced stability, perform consecutive rank-1 corrections (Lim and Comon, 2010). On the other hand, many efficient and stable algorithms exist for the more flexible Tucker format in (1.4), however, this format is not practical for tensor orders $N > 5$ because the number of entries of both the original data tensor and the core tensor (expressed in (1.4) 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, TT tensor network formats in (1.5) 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. The main advantage of the TT format over the CP decomposition is the ability to provide stable

quasi-optimal rank reduction, achieved through, for example, truncated singular value decompositions (tSVD) or adaptive cross-approximation (Oseledets and Tyrtyshnikov, 2010; Bebendorf, 2011; Khoromskij and Veit, 2016). 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[1]. In this monograph, we therefore make extensive use of the TT format for low-rank TN approximations and employ the TT toolbox software for efficient implementations (Oseledets *et al.*, 2012). The TT format will also serve as a basic prototype for high-order tensor representations, while we also consider the Hierarchical Tucker (HT) and the Tree Tensor Network States (TTNS) formats (having more general tree-like structures) whenever advantageous in applications.

Furthermore, we address in depth the concept of tensorization of structured vectors and matrices to convert a wide class of huge-scale optimization problems into much smaller-scale interconnected optimization sub-problems which can be solved by existing optimization methods (see Part 2 of this monograph).

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

- Tensorization of data vectors and matrices into a high-order tensor, followed by a distributed approximate representation of a cost function in a specific low-rank tensor network format.

- Execution of all computations and analysis in tensor network formats (i.e., using only core tensors) that scale linearly, or even sub-linearly (quantized tensor networks), in the tensor order N. This yields both the reduced computational complexity and distributed memory requirements.

---

[1]Although similar approaches have been known in quantum physics for a long time, their rigorous mathematical analysis is still a work in progress (see (Oseledets, 2011; Orús, 2014) and references therein).

1.4   Advantages of Multiway Analysis via Tensor Networks

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. The benefits of multiway (tensor) analysis methods for large-scale datasets then include:

- Ability to perform all mathematical operations in tractable tensor network formats;

- Simultaneous and flexible distributed representations of both the 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, flat view, 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 express mathematical operations on tensors (e.g., tensor contractions and reshaping) in a simple and intuitive way, and without the explicit use of complex mathematical expressions.

In that sense, this monograph both reviews current research in this area and complements optimisation methods, such as the Alternating Direction Method of Multipliers (ADMM) (Boyd *et al.*, 2011).

Tensor decompositions (TDs) have been already adopted in widely diverse disciplines, including psychometrics, chemometrics, biometric, quantum physics/information, quantum chemistry, signal and image processing, machine learning, and brain science (Smilde *et al.*, 2004; Tao *et al.*, 2007; Kroonenberg, 2008; Kolda and Bader, 2009; Hackbusch, 2012; Favier and de Almeida, 2014; Cichocki *et al.*, 2009, 2015b). This is largely due to their advantages in the analysis of data that exhibit not only large volume but also very high variety (see Figure 1.1), as in the case in bio- and neuroinformatics and in computational neuroscience, where various forms of data collection include sparse tabular structures and graphs or hyper-graphs.

Moreover, tensor networks have the ability to efficiently parameterize, through structured compact representations, very general high-dimensional spaces which arise in modern applications (Kressner *et al.*, 2014b; Cichocki, 2014; Zhang *et al.*, 2015; Corona *et al.*, 2015; Litsarev and Oseledets, 2016; Khoromskij and Veit, 2016; Benner *et al.*, 2016). Tensor networks also naturally account for intrinsic multidimensional and distributed patterns present in data, and thus provide the opportunity to develop very sophisticated models for capturing multiple interactions and couplings in data – these are more physically insightful and interpretable than standard pair-wise interactions.

1.5 Scope and Objectives

Review and tutorial papers (Kolda and Bader, 2009; Lu *et al.*, 2011; Grasedyck *et al.*, 2013; Cichocki *et al.*, 2015b; de Almeida *et al.*, 2015; Sidiropoulos *et al.*, 2016; Papalexakis *et al.*, 2016; Bachmayr *et al.*, 2016) and books (Smilde *et al.*, 2004; Kroonenberg, 2008; Cichocki *et al.*, 2009; Hackbusch, 2012) 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, 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), which deals with 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/sub-linear scaling with the size of datasets[2].

Furthermore, we review tensor decompositions and the associated algorithms for very large-scale linear/multilinear dimensionality reduction problems. The related optimization problems often involve structured matrices and vectors with over a billion entries (see (Grasedyck *et al.*, 2013; Dolgov, 2014; Garreis and Ulbrich, 2016) and references therein). In particular, we focus on Symmetric Eigenvalue Decomposition (EVD/PCA) and Generalized Eigenvalue Decomposition (GEVD) (Dolgov *et al.*, 2014; Kressner *et al.*, 2014a; Kressner and Uschmajew, 2016), SVD (Lee and Cichocki, 2015), solutions of overdetermined and undetermined systems of linear algebraic equations (Oseledets and Dolgov, 2012; Dolgov and Savostyanov, 2014), the Moore–Penrose pseudo-inverse of structured matrices (Lee and Cichocki, 2016b), and Lasso problems (Lee and Cichocki, 2016a). 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. For convenience, all these problems are reformulated as constrained optimization problems

---

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

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 a both an alternative and complementary to other emerging methods for huge-scale optimization problems like random coordinate descent (RCD) scheme (Nesterov, 2012; Richtárik and Takáč, 2016), sub-gradient methods (Nesterov, 2014), alternating direction method of multipliers (ADMM) (Boyd *et al.*, 2011), and proximal gradient descent methods (Parikh and Boyd, 2014) (see also (Cevher *et al.*, 2014; Hong *et al.*, 2016) 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 (see Part 2 of this monograph) are considered in detail, and 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.

2

Tensor Operations and Tensor Network Diagrams

Tensor operations benefit from the power of multilinear algebra which is structurally much richer than linear algebra, and even some basic properties, such as the rank, have a more complex meaning. 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 definitions and properties of tensor network operations is provided, including the outer, multi-linear, Kronecker, and Khatri–Rao products. For clarity, graphical illustrations are provided, together with an example rich guidance for tensor network operations and their properties. To avoid any confusion that may arise given the numerous options on tensor reshaping, both mathematical operations and their properties are expressed directly in their native multilinear contexts, supported by graphical visualizations.

2.1 Basic 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. Basic tensor operations are summarized in Table 2.1, and illustrated in Figures 2.1–2.13.

272

Table 2.1: Basic tensor/matrix operations.

$\underline{\mathbf{C}} = \underline{\mathbf{A}} \times_n \mathbf{B}$	Mode-n product of a tensor $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a 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 $$\underline{\mathbf{C}} = \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)}$$
$\underline{\mathbf{C}} = \underline{\mathbf{A}} \,\bar{\times}_n\, \mathbf{b}$	Mode-n product 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 $\mathbf{c}_j = \mathbf{a}_j \otimes_L \mathbf{b}_j \in \mathbb{R}^{IK}$
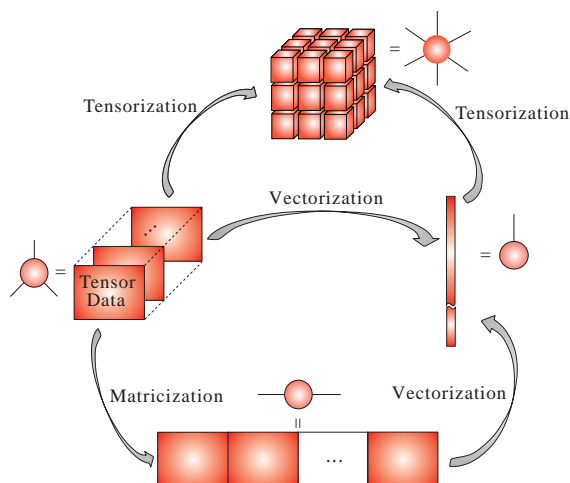
Figure 2.1: Tensor reshaping operations: Matricization, vectorization and tensorization. Matricization refers to converting a tensor into a matrix, vectorization to converting a tensor or a matrix into a vector, while tensorization refers to converting a vector, a matrix or a low-order tensor into a higher-order tensor.

We refer to (Kolda and Bader, 2009; Cichocki *et al.*, 2009; Lee and Cichocki, 2016c) for more detail regarding the basic notations and tensor operations. For convenience, general operations, such as $\text{vec}(\cdot)$ or $\text{diag}(\cdot)$, are defined similarly to the MATLAB syntax.

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 (Dolgov and Savostyanov, 2014):

1. Little-endian convention (reverse lexicographic ordering)

$$\overline{i_1 i_2 \cdots i_N} = i_1 + (i_2 - 1)I_1 + (i_3 - 1)I_1 I_2 + \cdots + (i_N - 1)I_1 \cdots I_{N-1}.$$

2. Big-endian (colexicographic ordering)

$$\overline{i_1 i_2 \cdots i_N} = i_N + (i_{N-1} - 1)I_N + (i_{N-2} - 1)I_N I_{N-1} + \cdots + (i_1 - 1)I_2 \cdots I_N.$$

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[1]. 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 2.2). 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

$$\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times I_1 I_2 \cdots I_{n-1} I_{n+1} \cdots I_N}, \tag{2.1}$$

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

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

Note that the columns of a mode-n matricization, $\mathbf{X}_{(n)}$, of a tensor $\underline{\mathbf{X}}$ are the mode-n fibers of $\underline{\mathbf{X}}$.

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

$$\mathbf{X}_{<n>} \in \mathbb{R}^{I_1 I_2 \cdots I_n \times I_{n+1} \cdots I_N}, \tag{2.2}$$

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

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

---

[1] 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, $\text{vec}(\mathbf{a} \circ \mathbf{b}) = \mathbf{b} \otimes \mathbf{a} = \mathbf{a} \otimes_L \mathbf{b}$.

(a)

(b)

(c)
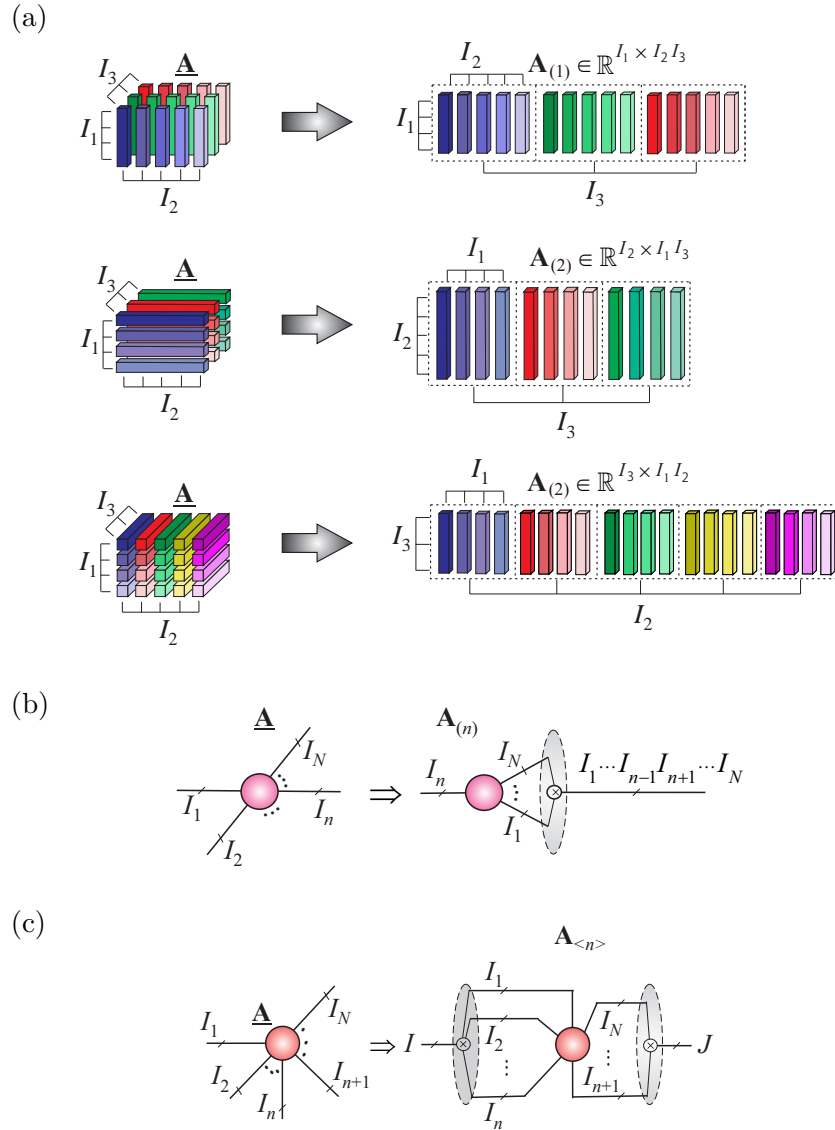
Figure 2.2: Matricization (flattening, unfolding) used in tensor reshaping. (a) Mode-1, mode-2, and mode-3 matricizations of a 3rd-order tensor, from the top to the bottom panel. (b) Tensor network diagram for the mode-n matricization of an Nth-order tensor, $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, into a short and wide matrix, $\mathbf{A}_{(n)} \in \mathbb{R}^{I_n \times I_1 \cdots I_{n-1} I_{n+1} \cdots I_N}$. (c) Mode-$\{1, 2, \ldots, n\}$th (canonical) matricization of an Nth-order tensor, $\underline{\mathbf{A}}$, into a matrix $\mathbf{A}_{<n>} = \mathbf{A}_{(\overline{i_1 \cdots i_n}; \, \overline{i_{n+1} \cdots i_N})} \in \mathbb{R}^{I_1 I_2 \cdots I_n \times I_{n+1} \cdots I_N}$.
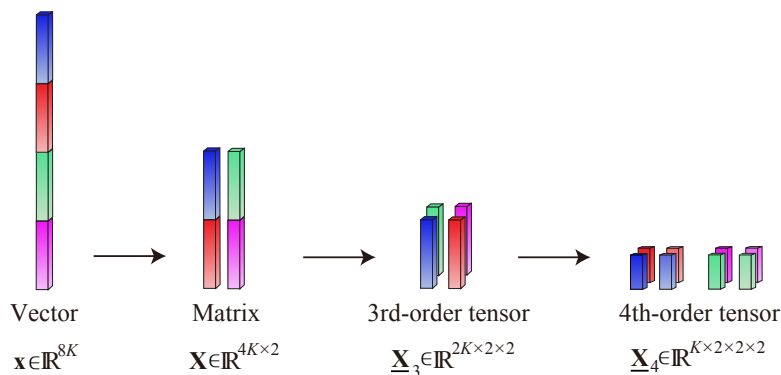
Figure 2.3: Tensorization of a vector into a matrix, 3rd-order tensor and 4th-order tensor.

The matricization operator in the MATLAB notation (reverse lexicographic) is given by

$$\mathbf{X}_{<n>} = \text{reshape}\left(\underline{\mathbf{X}}, I_1 I_2 \cdots I_n, I_{n+1} \cdots I_N\right). \qquad (2.3)$$

As special cases we immediately have (see Figure 2.2)

$$\mathbf{X}_{<1>} = \mathbf{X}_{(1)}, \quad \mathbf{X}_{<N-1>} = \mathbf{X}_{(N)}^{\mathrm{T}}, \quad \mathbf{X}_{<N>} = \text{vec}(\mathbf{X}). \qquad (2.4)$$

The tensorization of a vector or a matrix can be considered as a reverse process to the vectorization or matricization (see Figures 2.1 and 2.3).

Kronecker, strong Kronecker, and 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

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

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 2.4: Illustration of 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 $\mathbf{B} = [\mathbf{B}_{r_2,r_3}] \in \mathbb{R}^{R_2 I_2 \times R_3 J_2}$, which 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 product, 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

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

can be defined as a block matrix (see Figure 2.4 for a graphical illustration)

$$\mathbf{C} = \mathbf{A} \,|\!\otimes\!|\, \mathbf{B} \in \mathbb{R}^{R_1 IK \times R_3 JL}, \tag{2.5}$$

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 (de Launey and Seberry, 1994; Kazeev *et al.*, 2013a,b). Note that the strong Kronecker product is similar to the standard block matrix multiplication, but performed using Kronecker products of the blocks instead of the standard matrix-matrix products. The above definitions of Kronecker products can be naturally extended to tensors (Phan *et al.*, 2012) (see Table 2.1), as shown below.

The Kronecker product of tensors. The (Left) Kronecker product of two Nth-order 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 $\underline{\mathbf{C}} = \underline{\mathbf{A}} \otimes_L \underline{\mathbf{B}} \in \mathbb{R}^{I_1 J_1 \times \cdots \times I_N J_N}$ of the same order but enlarged in size, with entries $c_{\overline{i_1 j_1}, \ldots, \overline{i_N j_N}} = a_{i_1, \ldots, i_N} \, b_{j_1, \ldots, j_N}$ as
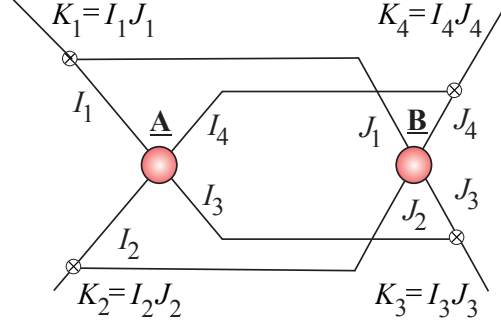
Figure 2.5: The left Kronecker product of two 4th-order tensors, $\underline{\mathbf{A}}$ and $\underline{\mathbf{B}}$, yields a 4th-order tensor, $\underline{\mathbf{C}} = \underline{\mathbf{A}} \otimes_L \underline{\mathbf{B}} \in \mathbb{R}^{I_1 J_1 \times \cdots \times I_4 J_4}$, with entries $c_{k_1,k_2,k_3,k_4} = a_{i_1,\ldots,i_4} \, b_{j_1,\ldots,j_4}$, where $k_n = \overline{i_n j_n}$ ($n = 1, 2, 3, 4$). Note that the order of tensor $\underline{\mathbf{C}}$ is the same as the order of $\underline{\mathbf{A}}$ and $\underline{\mathbf{B}}$, but the size in every mode within $\underline{\mathbf{C}}$ is a product of the respective sizes of $\underline{\mathbf{A}}$ and $\underline{\mathbf{B}}$.

illustrated in Figure 2.5.

The mode-n Khatri–Rao product of tensors. The Mode-n Khatri–Rao product of two Nth-order tensors, $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_n \times \cdots \times I_N}$ and $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_n \times \cdots \times J_N}$, for which $I_n = J_n$, 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 $\underline{\mathbf{C}}(:,\ldots:,i_n,:,\ldots,:) = \underline{\mathbf{A}}(:,\ldots:,i_n,:,\ldots,:) \otimes \underline{\mathbf{B}}(:,\ldots:,i_n,:,\ldots,:)$.

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

$$\mathbf{A} \odot \mathbf{B} = [\mathbf{a}_1 \otimes \mathbf{b}_1, \mathbf{a}_2 \otimes \mathbf{b}_2, \ldots, \mathbf{a}_R \otimes \mathbf{b}_R] \in \mathbb{R}^{IJ \times R}, \qquad (2.6)$$

$$\mathbf{A} \odot_L \mathbf{B} = [\mathbf{a}_1 \otimes_L \mathbf{b}_1, \mathbf{a}_2 \otimes_L \mathbf{b}_2, \ldots, \mathbf{a}_R \otimes_L \mathbf{b}_R] \in \mathbb{R}^{IJ \times R}. \qquad (2.7)$$

---

[2]For simplicity, the mode 2 subindex is usually neglected, i.e., $\mathbf{A} \odot_2 \mathbf{B} = \mathbf{A} \odot \mathbf{B}$.

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

$$\mathbf{A} \odot_1 \mathbf{B} = \begin{bmatrix} \mathbf{A}(1,:) \otimes \mathbf{B}(1,:) \\ \vdots \\ \mathbf{A}(I,:) \otimes \mathbf{B}(I,:) \end{bmatrix} \in \mathbb{R}^{I \times RQ}. \tag{2.8}$$
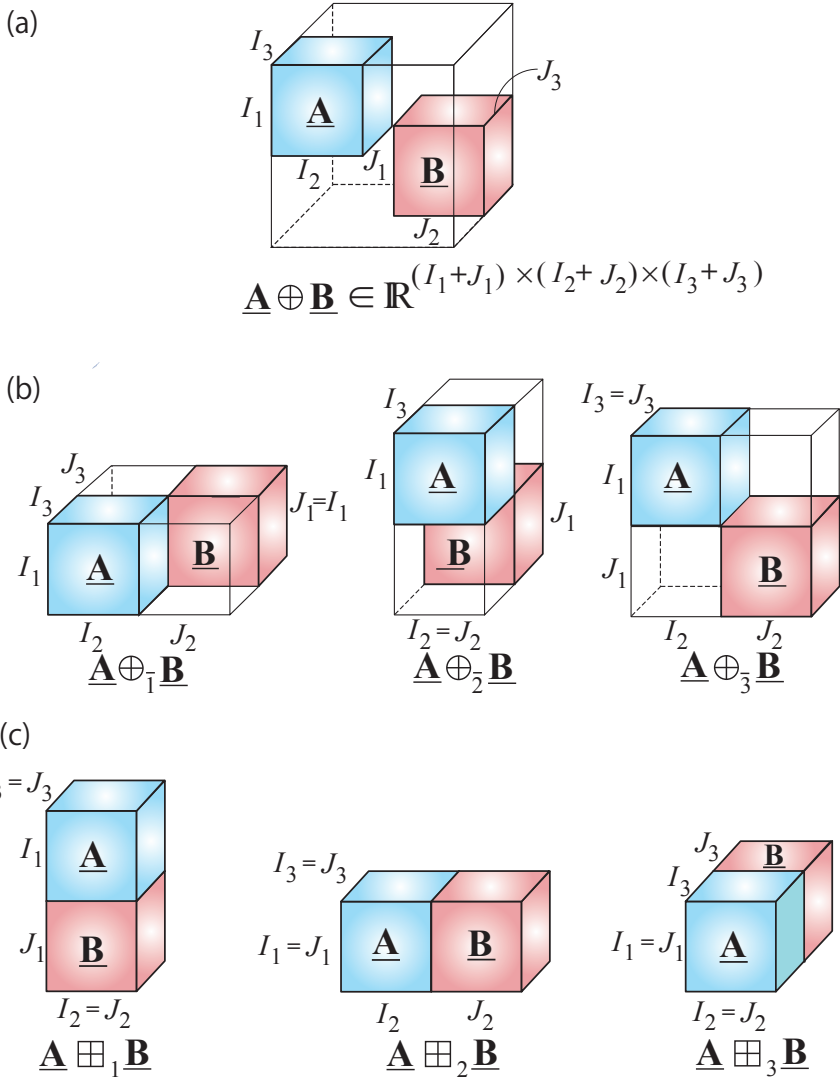
Direct sum of tensors. A direct sum of Nth-order tensors $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times \cdots \times J_N}$ yields a tensor $\underline{\mathbf{C}} = \underline{\mathbf{A}} \oplus \underline{\mathbf{B}} \in \mathbb{R}^{(I_1+J_1) \times \cdots \times (I_N+J_N)}$, 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 Figure 2.6(a)).

Partial (mode-n) direct sum of tensors. A 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 $\underline{\mathbf{C}} = \underline{\mathbf{A}} \oplus_{\overline{n}} \underline{\mathbf{B}} \in \mathbb{R}^{(I_1+J_1) \times \cdots \times (I_{n-1}+J_{n-1}) \times I_n \times (I_{n+1}+J_{n+1}) \times \cdots \times (I_N+J_N)}$, where $\underline{\mathbf{C}}(:,\ldots,:,i_n,:,\ldots,:) = \underline{\mathbf{A}}(:,\ldots,:,i_n,:,\ldots,:) \oplus \underline{\mathbf{B}}(:,\ldots,:,i_n,:,\ldots,:)$, as illustrated in Figure 2.6(b).

Concatenation of Nth-order tensors. A concatenation along mode-n of tensors $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times \cdots \times J_N}$, for which $I_m = J_m$, $\forall m \neq n$ yields a tensor $\underline{\mathbf{C}} = \underline{\mathbf{A}} \boxplus_n \underline{\mathbf{B}} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times (I_n+J_n) \times I_{n+1} \times \cdots \times (I_N)}$, with subtensors $\underline{\mathbf{C}}(i_1,\ldots,i_{n-1},:,i_{n+1},\ldots,i_N) = \underline{\mathbf{A}}(i_1,\ldots,i_{n-1},:,i_{n+1},\ldots,i_N) \oplus \underline{\mathbf{B}}(i_1,\ldots,i_{n-1},:,i_{n+1},\ldots,i_N)$, as illustrated in Figure 2.6(c). For a 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, consider two 3rd-order tensors $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$. Their 3D Convolution 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 $\underline{\mathbf{C}}(k_1,k_2,k_3) = \sum_{j_1} \sum_{j_2} \sum_{j_3} \underline{\mathbf{B}}(j_1,j_2,j_3) \underline{\mathbf{A}}(k_1 - j_1, k_2 - j_2, k_3 - j_3)$ as illustrated in Figure 2.7 and Figure 2.8.

Partial (mode-n) Convolution. For simplicity, consider two 3rd-order tensors $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$. Their mode-2 (partial)

(a)

$$\underline{\mathbf{A}} \oplus \underline{\mathbf{B}} \in \mathbb{R}^{(I_1+J_1) \times (I_2+J_2) \times (I_3+J_3)}$$

(b)

$$\underline{\mathbf{A}} \oplus_{\bar{1}} \underline{\mathbf{B}} \qquad \underline{\mathbf{A}} \oplus_{\bar{2}} \underline{\mathbf{B}} \qquad \underline{\mathbf{A}} \oplus_{\bar{3}} \underline{\mathbf{B}}$$

(c)

$$\underline{\mathbf{A}} \boxplus_{1} \underline{\mathbf{B}} \qquad \underline{\mathbf{A}} \boxplus_{2} \underline{\mathbf{B}} \qquad \underline{\mathbf{A}} \boxplus_{3} \underline{\mathbf{B}}$$

Figure 2.6: Illustration of the direct sum, partial direct sum and concatenation operators of two 3rd-order tensors. (a) Direct sum. (b) Partial (mode-1, mode-2, and mode-3) direct sum. (c) Concatenations along mode-1,2,3.

A

1	2	3	4
0	3	2	1
5	0	1	4
3	1	0	2

$*$

B

0	-1	0
-1	4	-1
0	-1	0

$=$

C

0	-1	-2	-3	-4	0
-1	2	1	4	12	-4
0	-9	8	0	-6	-1
-5	17	-10	-2	12	-4
-3	6	1	-4	4	-2
0	-3	-1	0	-2	0

$1 \cdot 4 + 2 \cdot (-1) = 2$

$2 \cdot (-1) + 3 \cdot 4 + 2 \cdot (-1) = 8$

$3 \cdot (-1) + 3 \cdot (-1) + 2 \cdot 4 + 1 \cdot (-1) + 1 \cdot (-1) = 0$
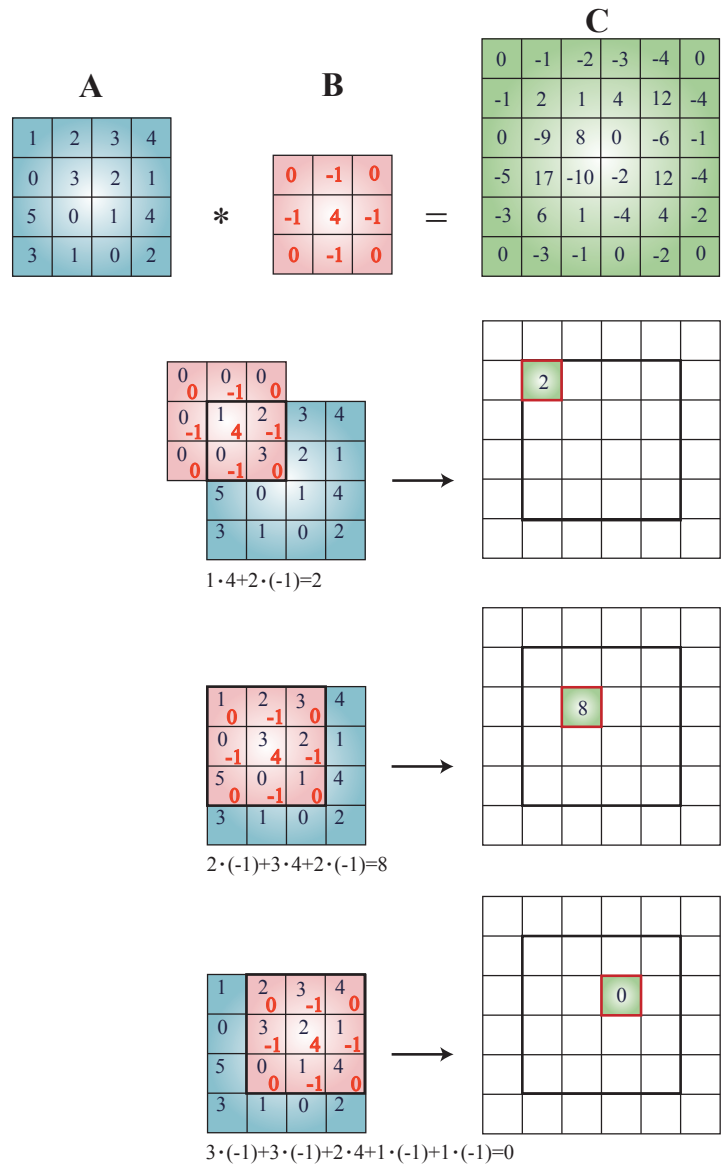
Figure 2.7: Illustration of the 2D convolution operator, performed through a sliding window operation along both the horizontal and vertical index.

A **B** (I_3+J_3-1) **C**

I_3

I_1 J_3 (I_1+J_1-1) 3

$*$ J_1 $=$

I_2 J_2

Σ Reduction (summation)

2	3	3
6	2	4
4	2	5

0	-1	0
-1	5	-1
0	-1	0

0	-3	0
-6	10	-4
0	-2	0

4	0	3
2	3	5
2	1	2

-2	-1	0
-1	1	1
0	1	2

-8	0	0
-2	3	5
0	1	4

$\circledast$ $=$

0	3	2
2	3	1
1	0	5

0	-1	0
-1	4	-1
0	-1	0

0	-3	0
-2	12	-1
0	-1	0

Hadamard product
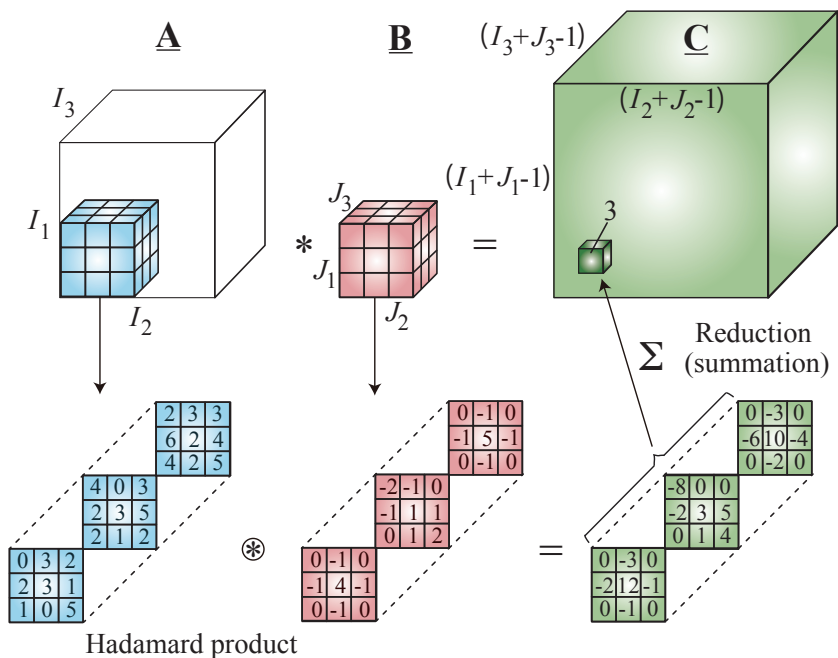
Figure 2.8: Illustration of the 3D convolution operator, performed through a sliding window operation along all three indices.

convolution yields a tensor $\underline{\mathbf{C}} = \underline{\mathbf{A}} \boxdot_2 \underline{\mathbf{B}} \in \mathbb{R}^{I_1 J_1 \times (I_2+J_2-1) \times I_3 J_3}$, the subtensors (vectors) of which are $\underline{\mathbf{C}}(k_1,:,k_3) = \underline{\mathbf{A}}(i_1,:,i_3) * \underline{\mathbf{B}}(j_1,: ,j_3) \in \mathbb{R}^{I_2+J_2-1}$, where $k_1 = \overline{i_1 j_1}$, and $k_3 = \overline{i_3 j_3}$.

Outer 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 $c_{i_1,\dots,i_N,j_1,\dots,j_M} = a_{i_1,\dots,i_N} b_{j_1,\dots,j_M}$.

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, $\mathbf{X} = \mathbf{a} \circ \mathbf{b} = \mathbf{a}\mathbf{b}^{\mathrm{T}} \in \mathbb{R}^{I \times J}$. 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, $\underline{\mathbf{X}} = \mathbf{b}^{(1)} \circ \mathbf{b}^{(2)} \circ \cdots \circ \mathbf{b}^{(N)}$ of nonzero vectors, $\mathbf{b}^{(n)} \in \mathbb{R}^{I_n}$, with the tensor entries given by $x_{i_1,i_2,\ldots,i_N} = b_{i_1}^{(1)} b_{i_2}^{(2)} \cdots b_{i_N}^{(N)}$.

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

$$\underline{\mathbf{X}} = \sum_{r=1}^{R} \mathbf{b}_r^{(1)} \circ \mathbf{b}_r^{(2)} \circ \cdots \circ \mathbf{b}_r^{(N)} = \sum_{r=1}^{R} \left(\underset{n=1}{\overset{N}{\circ}} \mathbf{b}_r^{(n)} \right), \quad \mathbf{b}_r^{(n)} \in \mathbb{R}^{I_n}, \quad (2.9)$$

which is exactly the form of the Kruskal tensor, illustrated in Figure 2.9, also known under the names of CANDECOMP/PARAFAC, Canonical Polyadic Decomposition (CPD), or simply the CP decomposition in (1.2). We will use the acronyms CP and CPD.

Tensor rank. The tensor rank, also called the CP rank, is a natural extension of the matrix rank and is defined as a minimum number, R, of rank-1 terms in an exact CP decomposition of the form in (2.9).

Although the CP decomposition has already found many practical applications, its limiting theoretical property is that the best rank-R approximation of a given data tensor may not exist (see de Silva and Lim (2008) for more detail). However, a rank-R tensor can be approximated arbitrarily well 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 (Bini, 1985), which is defined as the minimum number of rank-1 tensors which provides the approximation of a given tensor 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 its modes, and the same value of entries for every permutation of its indices. For example, for vectors $\mathbf{b}^{(n)} = \mathbf{b} \in \mathbb{R}^I$, $\forall n$, 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 can be expressed as a
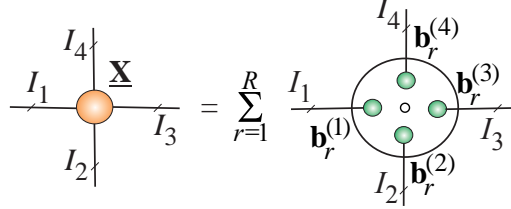
Figure 2.9: The CP decomposition for a 4th-order tensor $\underline{\mathbf{X}}$ of rank R. Observe that the rank-1 subtensors are formed through the outer products of the vectors $\mathbf{b}_r^{(1)}, \ldots, \mathbf{b}_r^{(4)}$, $r = 1, \ldots, R$.

linear combination of such symmetric rank-1 tensors through the so-called symmetric CP decomposition, given by

$$\underline{\mathbf{X}} = \sum_{r=1}^{R} \lambda_r \mathbf{b}_r \circ \mathbf{b}_r \circ \cdots \circ \mathbf{b}_r, \qquad \mathbf{b}_r \in \mathbb{R}^I, \qquad (2.10)$$

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 for its exact representation.

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

$$\underline{\mathbf{C}} = \underline{\mathbf{A}} \times_n \mathbf{B} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N}, \qquad (2.11)$$

with entries

$$c_{i_1, i_2, \ldots, i_{n-1}, j, i_{n+1}, \ldots, i_N} = \sum_{i_n=1}^{I_n} a_{i_1, i_2, \ldots, i_N} \, b_{j, i_n}. \qquad (2.12)$$

From (2.12) and Figure 2.10, 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 (Li *et al.*, 2015; Ballard *et al.*, 2015a,b).

Full multilinear (Tucker) product. A full multilinear product, also called the Tucker product, of an Nth-order tensor, $\underline{\mathbf{G}} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$,

(a)

$$\underline{\mathbf{C}} = \underline{\mathbf{A}} \times_1 \mathbf{B} \qquad\qquad \mathbf{C}_{(1)} = \mathbf{B}\,\mathbf{A}_{(1)}$$

(b)

$$\underline{\mathbf{C}} = \underline{\mathbf{A}} \times_n \mathbf{B} \qquad\qquad \mathbf{C}_{(n)} = \mathbf{B}\,\mathbf{A}_{(n)}$$

Figure 2.10: 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) Graphical representation of a 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, $\mathbf{B} \in \mathbb{R}^{J \times I_n}$.

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 in all the modes and can be compactly written as (see Figure 2.11(b))

$$\begin{aligned}
\underline{\mathbf{C}} &= \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)} \qquad\qquad (2.13)\\
&= [\![\underline{\mathbf{G}}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N)}]\!] \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}.
\end{aligned}$$

Observe that this format corresponds to the Tucker decomposition (Tucker, 1964, 1966; Kolda and Bader, 2009) (see Section 3.3).

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) yields a tensor

$$\underline{\mathbf{C}} = \underline{\mathbf{A}} \, \bar{\times}_n \mathbf{b} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N}, \qquad (2.14)$$
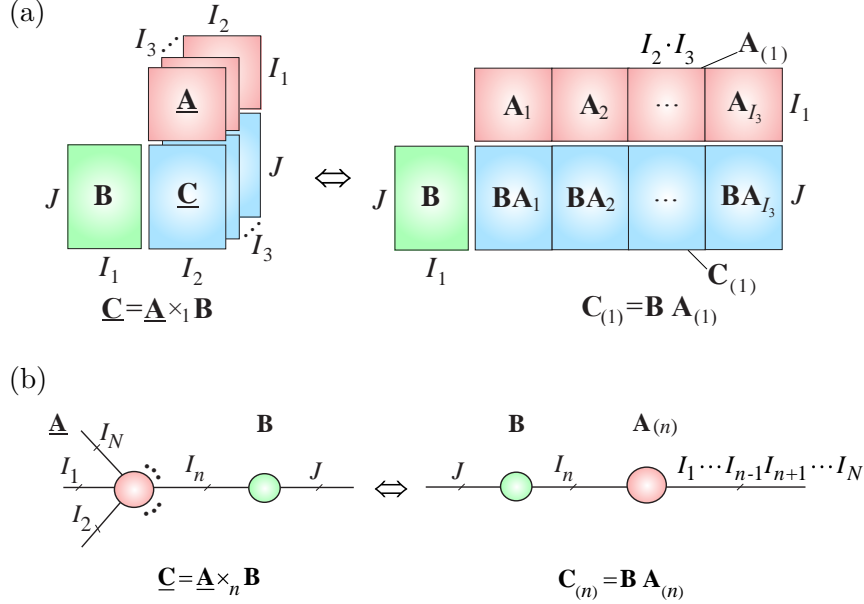
with entries

$$c_{i_1,\ldots,i_{n-1},i_{n+1},\ldots,i_N} = \sum_{i_n=1}^{I_n} a_{i_1,\ldots,i_{n-1},i_n,i_{n+1},\ldots,i_N} \, b_{i_n}. \qquad (2.15)$$

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, with the mode n removed (see Figure 2.11).

Multilinear products of tensors by matrices or vectors play a key role in deterministic methods for the reshaping of tensors and dimensionality reduction, as well as in probabilistic methods for randomization/sketching procedures and in random projections of tensors into matrices or vectors. In other words, 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. (Sun *et al.*, 2006; Drineas and Mahoney, 2007; Lu *et al.*, 2011; Li and Monga, 2012; Pham and Pagh, 2013; Wang *et al.*, 2015; Kuleshov *et al.*, 2015; Sorber *et al.*, 2016) (see Section 3.5 for more detail).

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 multiplication, inner product, and outer product.

In a way similar to the mode-n multilinear product[3], the mode-$\binom{m}{n}$ product (tensor contraction) of two tensors, $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_M}$, with common modes, $I_n = J_m$, yields an $(N+M-2)$-order tensor, $\underline{\mathbf{C}} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N \times J_1 \times \cdots \times J_{m-1} \times J_{m+1} \times \cdots \times J_M}$, in the form (see Figure 2.12(a))
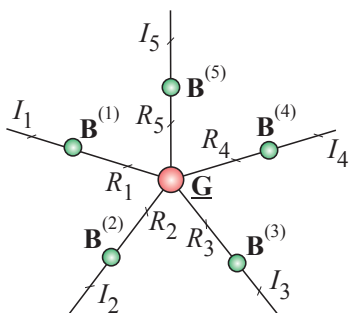
$$\underline{\mathbf{C}} = \underline{\mathbf{A}} \times_n^m \underline{\mathbf{B}}, \qquad (2.16)$$

---

[3]In the literature, sometimes the symbol $\times_n$ is replaced by $\bullet_n$.
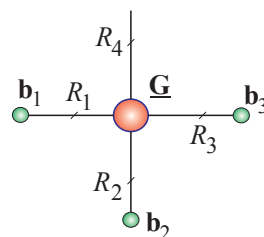
(a)
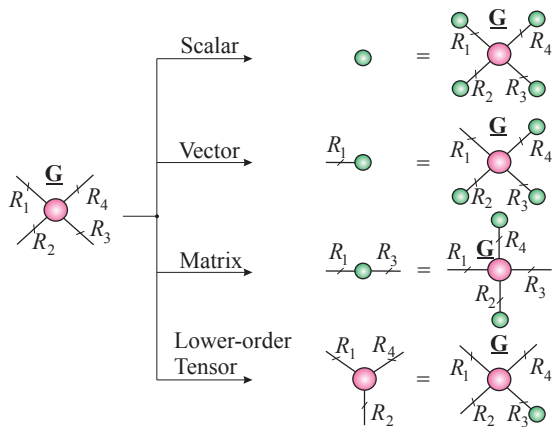

(b)   (c)

Figure 2.11: Multilinear tensor products in a compact tensor network notation. (a) Transforming and/or compressing a 4th-order tensor, $\underline{\mathbf{G}} \in \mathbb{R}^{R_1 \times R_2 \times R_3 \times R_4}$, 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 by one. For example, a multilinear product of a 4th-order 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 4th-order tensor, $\underline{\mathbf{G}} \in \mathbb{R}^{R_1 \times R_2 \times R_3 \times R_4}$, and three vectors, $\mathbf{b}_n \in \mathbb{R}^{R_n}$ $(n = 1, 2, 3)$, yields the vector $\mathbf{c} = \underline{\mathbf{G}} \bar{\times}_1 \mathbf{b}_1 \bar{\times}_2 \mathbf{b}_2 \bar{\times}_3 \mathbf{b}_3 \in \mathbb{R}^{R_4}$.

(a)

(b)

(c)

(d)
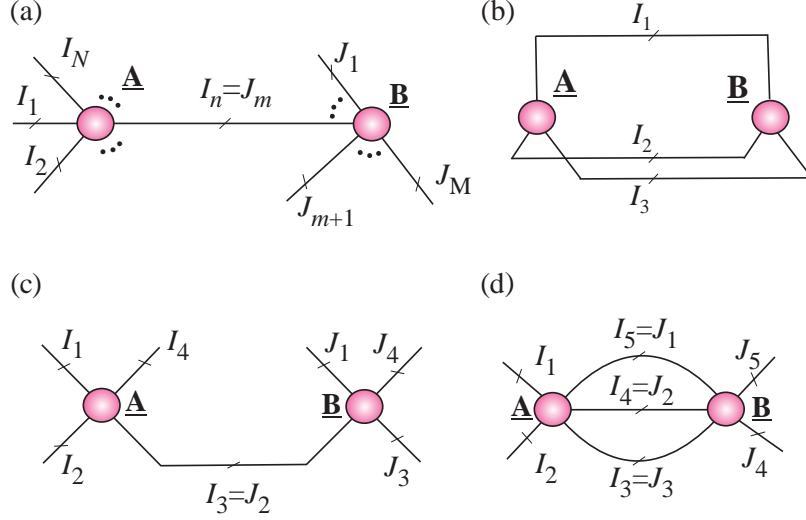
Figure 2.12: Examples of contractions of two tensors. (a) Multilinear product of two tensors is denoted by $\underline{\mathbf{A}} \times_n^m \underline{\mathbf{B}}$. (b) Inner product of two 3rd-order tensors yields a scalar $c = \langle \underline{\mathbf{A}}, \underline{\mathbf{B}} \rangle = \underline{\mathbf{A}} \times_{1,2,3}^{1,2,3} \underline{\mathbf{B}} = \underline{\mathbf{A}} \,\bar{\times}\, \underline{\mathbf{B}} = \sum_{i_1,i_2,i_3} a_{i_1,i_2,i_3}\, b_{i_1,i_2,i_3}$. (c) Tensor contraction of two 4th-order tensors, along mode-3 in $\underline{\mathbf{A}}$ and mode-2 in $\underline{\mathbf{B}}$, 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 along the modes $3,4,5$ in $\underline{\mathbf{A}}$ and $1,2,3$ in $\underline{\mathbf{B}}$ yields a 4th-order tensor, $\underline{\mathbf{C}} = \underline{\mathbf{A}} \times_{5,4,3}^{1,2,3} \underline{\mathbf{B}} \in \mathbb{R}^{I_1 \times I_2 \times J_4 \times J_5}$.

for which the entries are computed as

$$c_{i_1, \ldots, i_{n-1}, i_{n+1}, \ldots, i_N, j_1, \ldots, j_{m-1}, j_{m+1}, \ldots, j_M} =$$

$$= \sum_{i_n=1}^{I_n} a_{i_1, \ldots, i_{n-1}, i_n, i_{n+1}, \ldots, i_N}\; b_{j_1, \ldots, j_{m-1}, i_n, j_{m+1}, \ldots, j_M}. \quad (2.17)$$

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 2.12. For convenience of presentation, the super- or sub-index, e.g., m, n, will be omitted in a few special cases. 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

$$\underline{\mathbf{C}} = \underline{\mathbf{A}} \times_N^1 \underline{\mathbf{B}} = \underline{\mathbf{A}} \times^1 \underline{\mathbf{B}} = \underline{\mathbf{A}} \bullet \underline{\mathbf{B}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N-1} \times J_2 \times \cdots \times J_M}, \quad (2.18)$$

for which the entries

$$c_{i_1,i_2,\dots,i_{N-1},j_2,j_3,\dots,j_M} = \sum_{i_N=1}^{I_N} a_{i_1,i_2,\dots,i_N}\, b_{i_N,j_2,\dots,j_M}.$$

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

Note that tensor contractions are, in general not associative or commutative, since when contracting more than two tensors, the order has to be precisely specified (defined), for example, $\underline{\mathbf{A}} \times_a^b (\underline{\mathbf{B}} \times_c^d \underline{\mathbf{C}})$ for $b < c$.

It is also important to note that a matrix-by-vector product, $\mathbf{y} = \mathbf{Ax} \in \mathbb{R}^{I_1 \cdots I_N}$, 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 $\underline{\mathbf{X}}$ (see Section 4.5).

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. In practice, due to the high computational complexity of tensor contractions, especially for tensor networks with loops, this operation is often performed approximately (Lubasch *et al.*, 2014; Di Napoli *et al.*, 2014; Pfeifer *et al.*, 2014; Kao *et al.*, 2015).

Tensor trace. Consider a tensor with partial self-contraction modes, where the outer (or open) indices represent physical modes of the tensor, while the inner indices indicate its contraction modes. The Tensor Trace operator performs the summation of all inner indices of the tensor (Gu *et al.*, 2009). For example, a tensor $\underline{\mathbf{A}}$ of size $R \times I \times R$ has two inner indices, modes 1 and 3 of size R, and one open mode of size I. Its tensor trace yields a vector of length I, given by

$$\mathbf{a} = \mathrm{Tr}(\underline{\mathbf{A}}) = \sum_r \underline{\mathbf{A}}(r, :, r)\,,$$

the elements of which are the traces of its lateral slices $\mathbf{A}_i \in \mathbb{R}^{R \times R}$ $(i = 1, 2, \dots, I)$, that is, (see bottom of Figure 2.13)

$$\mathbf{a} = [\mathrm{tr}(\mathbf{A}_1), \dots, \mathrm{tr}(\mathbf{A}_i), \dots, \mathrm{tr}(\mathbf{A}_I)]^{\mathrm{T}}. \tag{2.19}$$
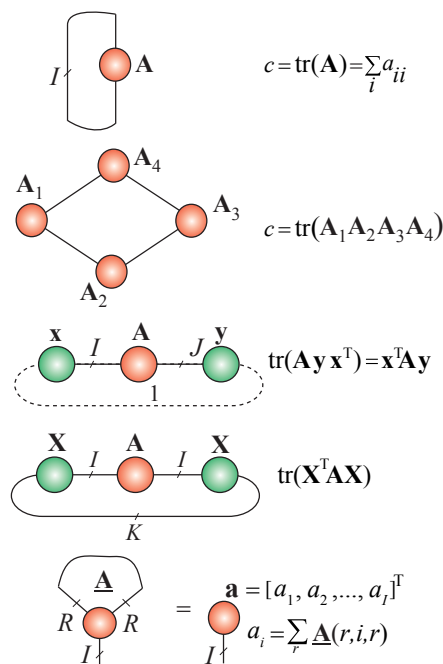
Figure 2.13: Tensor network notation for the traces of matrices (panels 1-4 from the top), and a (partial) tensor trace (tensor self-contraction) of a 3rd-order tensor (bottom panel). Note that graphical representations of the trace of matrices intuitively explain the permutation property of trace operator, e.g., $\text{tr}(\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3\mathbf{A}_4) = \text{tr}(\mathbf{A}_3\mathbf{A}_4\mathbf{A}_1\mathbf{A}_2)$.

A tensor can have more than one pair of inner indices, e.g., the tensor $\underline{\mathbf{A}}$ of size $R \times I \times S \times S \times I \times R$ has two pairs of inner indices, modes 1 and 6, modes 3 and 4, and two open modes (2 and 5). The tensor trace of $\underline{\mathbf{A}}$ therefore returns a matrix of size $I \times I$ defined as

$$\text{Tr}(\underline{\mathbf{A}}) = \sum_r \sum_s \underline{\mathbf{A}}(r, :, s, s, :, r).$$

A variant of Tensor Trace (Lee and Cichocki, 2016c) for the case of the partial tensor self-contraction considers a tensor $\underline{\mathbf{A}} \in \mathbb{R}^{R \times I_1 \times I_2 \times \cdots \times I_N \times R}$ and yields a reduced-order tensor $\underline{\tilde{\mathbf{A}}} = \text{Tr}(\underline{\mathbf{A}}) \in$
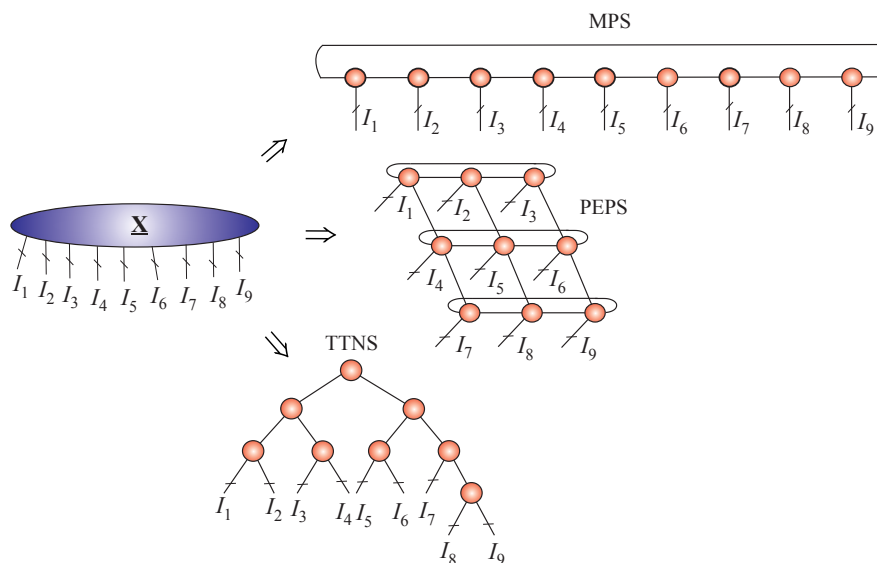
Figure   2.14: Illustration of the decomposition of a 9th-order tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_9}$, into different forms of tensor networks (TNs). In general, the objective is to decompose a very high-order tensor into sparsely (weakly) connected low-order and small size tensors, typically 3rd-order and 4th-order tensors called cores. Top: The Tensor Chain (TC) model, which is equivalent to the Matrix Product State (MPS) with periodic boundary conditions (PBC). Middle: The Projected Entangled-Pair States (PEPS), also with PBC. Bottom: The Tree Tensor Network State (TTNS).

$\mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, with entries

$$\widetilde{\underline{\mathbf{A}}}(i_1, i_2, \ldots, i_N) = \sum_{r=1}^{R} \underline{\mathbf{A}}(r, i_1, i_2, \ldots, i_N, r), \qquad (2.20)$$

Conversions of tensors to scalars, vectors, matrices or tensors with reshaped modes and/or reduced orders are illustrated in Figures 2.11– 2.13.

2.2 Graphical Representation of Fundamental Tensor Networks

Tensor networks (TNs) represent a higher-order tensor as a set of sparsely interconnected lower-order tensors (see Figure 2.14), and

in this way provide computational and storage benefits. The lines (branches, edges) connecting core tensors correspond to the contracted modes while their weights (or numbers of branches) represent the rank of a tensor network[4], whereas the lines which do not connect core tensors correspond to the "external" physical variables (modes, indices) within the data tensor. In other words, the number of free (dangling) edges (with weights larger than one) determines the order of a data tensor under consideration, while set of weights of internal branches represents the TN rank.

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

Hierarchical Tucker (HT) decompositions (also called hierarchical tensor representation) have been introduced in (Hackbusch and Kühn, 2009) and also independently in (Grasedyck, 2010), see also (Hackbusch, 2012; Lubich *et al.*, 2013; Uschmajew and Vandereycken, 2013; Kressner and Tobler, 2014; Bachmayr *et al.*, 2016) and references therein[5]. Generally, the HT decomposition requires splitting the set of modes of a tensor in a hierarchical way, which results in a binary tree containing a subset of modes at each branch (called a dimension tree); examples of binary trees are given in Figures 2.15, 2.16 and 2.17. 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 cycles (loops), i.e., no edges connecting a node with itself. The splitting operation of the set of modes of the original data tensor by binary tree edges is performed through a suitable matricization.

    Choice of dimension tree. 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

---

[4]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.

[5]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) (Wang and Thoss, 2003). Furthermore, the PARATREE model, developed independently for signal processing applications (Salmi *et al.*, 2009), is quite similar to the HT model (Grasedyck, 2010).
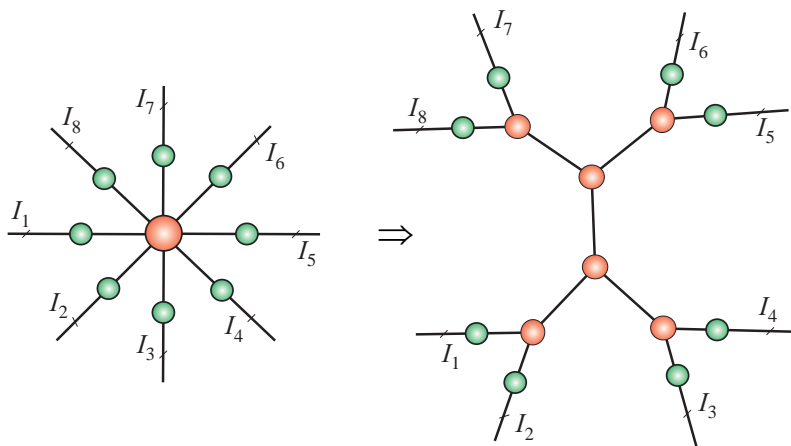
Figure 2.15: The standard Tucker decomposition of an 8th-order tensor into a core tensor (red circle) and eight factor matrices (green circles), and its transformation into an equivalent Hierarchical Tucker (HT) model using interconnected smaller size 3rd-order core tensors and the same factor matrices.

are "separated" from other groups of modes, so that a sequential HT decomposition can be performed via a (truncated) SVD applied to a suitably matricized tensor. 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 Section 2.2.2 and Section 4 (Oseledets, 2011; Oseledets and Tyrtyshnikov, 2009).

Using mathematical formalism, a dimension tree is a binary tree T_N, $N > 1$, which satisfies that

   (i) all nodes $t \in T_N$ are non-empty subsets of $\{1, 2, \ldots, N\}$,

  (ii) the set $t_{root} = \{1, 2, \ldots, N\}$ is the root node of T_N, and

 (iii) each non-leaf node has two children $u, v \in T_N$ such that t is a disjoint union $t = u \cup v$.

The HT model is illustrated through the following Example.

Example. Suppose that the dimension tree T_7 is given, which gives
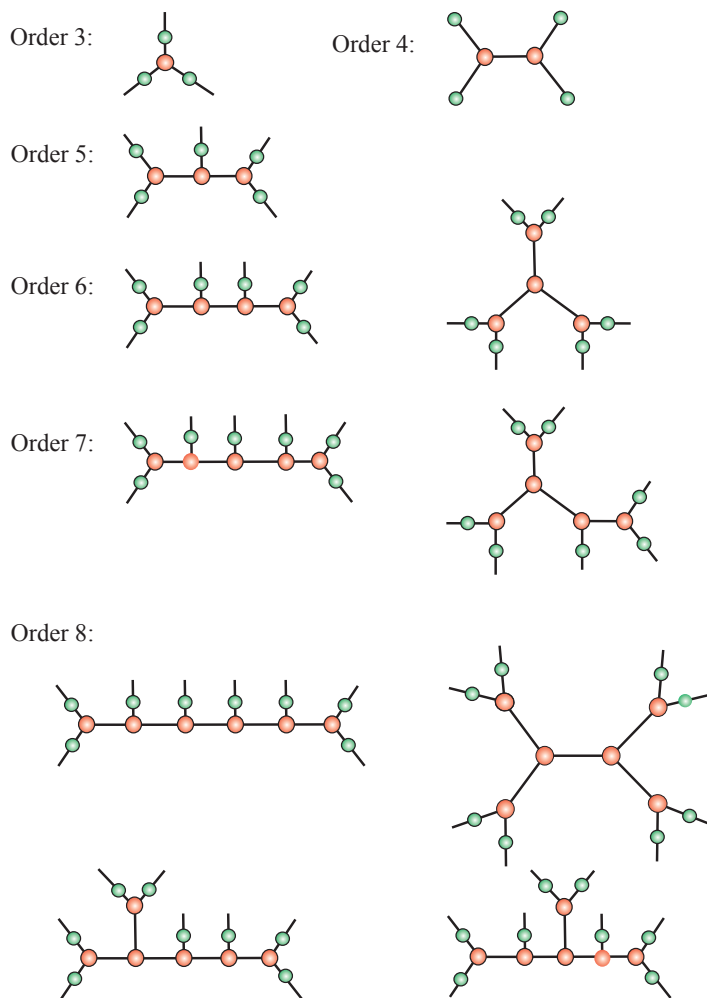
Figure 2.16: Examples of HT/TT models (formats) for distributed Tucker decompositions with 3rd-order cores, for different orders of data tensors. Green circles denote factor matrices (which can be absorbed by core tensors), while red circles indicate cores. Observe that the representations are not unique.

the HT decomposition illustrated in Figure 2.17. The HT decomposition of a tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_7}$ with given set of integers $\{R_t\}_{t \in T_7}$ can
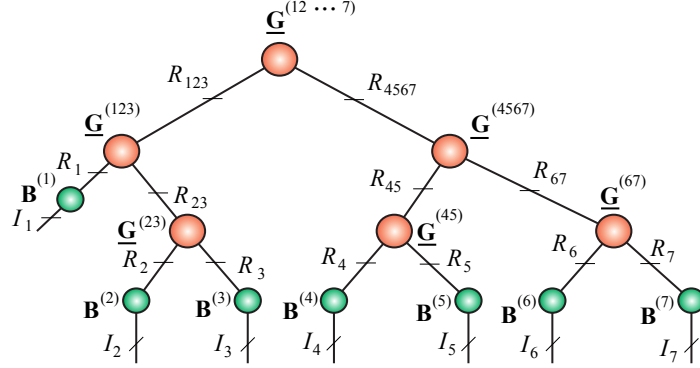
Figure 2.17: Example illustrating the HT decomposition for a 7th-order data tensor.

be expressed in the tensor and vector/matrix forms as follows. Let intermediate tensors $\underline{\mathbf{X}}^{(t)}$ with $t = \{n_1, \ldots, n_k\} \subset \{1, \ldots, 7\}$ have the size $I_{n_1} \times I_{n_2} \times \cdots \times I_{n_k} \times R_t$. Let $\underline{\mathbf{X}}_{r_t}^{(t)} \equiv \underline{\mathbf{X}}^{(t)}(:, \ldots, :, r_t)$ denote the subtensor of $\underline{\mathbf{X}}^{(t)}$ and $\mathbf{X}^{(t)} \equiv \mathbf{X}_{<k>}^{(t)} \in \mathbb{R}^{I_{n_1} I_{n_2} \cdots I_{n_k} \times R_t}$ denote the corresponding unfolded matrix. Let $\underline{\mathbf{G}}^{(t)} \in \mathbb{R}^{R_u \times R_v \times R_t}$ be core tensors where u and v denote respectively the left and right children of t.

The HT model shown in Figure 2.17 can be then described mathematically in the vector form as

$$\text{vec}(\underline{\mathbf{X}}) \cong \left(\mathbf{X}^{(123)} \otimes_L \mathbf{X}^{(4567)}\right) \text{vec}\left(\mathbf{G}^{(12\cdots7)}\right),$$

$$\mathbf{X}^{(123)} \cong \left(\mathbf{B}^{(1)} \otimes_L \mathbf{X}^{(23)}\right) \mathbf{G}_{<2>}^{(123)}, \quad \mathbf{X}^{(4567)} \cong \left(\mathbf{X}^{(45)} \otimes_L \mathbf{X}^{(67)}\right) \mathbf{G}_{<2>}^{(4567)},$$

$$\mathbf{X}^{(23)} \cong \left(\mathbf{B}^{(2)} \otimes_L \mathbf{B}^{(3)}\right) \mathbf{G}_{<2>}^{(23)}, \qquad \mathbf{X}^{(45)} \cong \left(\mathbf{B}^{(4)} \otimes_L \mathbf{B}^{(5)}\right) \mathbf{G}_{<2>}^{(45)},$$

$$\mathbf{X}^{(67)} \cong \left(\mathbf{B}^{(6)} \otimes_L \mathbf{B}^{(7)}\right) \mathbf{G}_{<2>}^{(67)}.$$

An equivalent, more explicit form, using tensor notations becomes

$$\underline{\mathbf{X}} \cong \sum_{r_{123}=1}^{R_{123}} \sum_{r_{4567}=1}^{R_{4567}} g_{r_{123},r_{4567}}^{(12\cdots7)} \, \underline{\mathbf{X}}_{r_{123}}^{(123)} \circ \underline{\mathbf{X}}_{r_{4567}}^{(4567)},$$

$$\underline{\mathbf{X}}_{r_{123}}^{(123)} \cong \sum_{r_1=1}^{R_1} \sum_{r_{23}=1}^{R_{23}} g_{r_1,r_{23},r_{123}}^{(123)} \mathbf{b}_{r_1}^{(1)} \circ \mathbf{X}_{r_{23}}^{(23)},$$

$$\underline{\mathbf{X}}_{r_{4567}}^{(4567)} \cong \sum_{r_{45}=1}^{R_{45}} \sum_{r_{67}=1}^{R_{67}} g_{r_{45},r_{67},r_{4567}}^{(4567)} \mathbf{X}_{r_{45}}^{(45)} \circ \mathbf{X}_{r_{67}}^{(67)},$$

$$\mathbf{X}_{r_{23}}^{(23)} \cong \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} g_{r_2,r_3,r_{23}}^{(23)} \, \mathbf{b}_{r_2}^{(2)} \circ \mathbf{b}_{r_3}^{(3)},$$

$$\mathbf{X}_{r_{45}}^{(45)} \cong \sum_{r_4=1}^{R_4} \sum_{r_5=1}^{R_5} g_{r_4,r_5,r_{45}}^{(45)} \, \mathbf{b}_{r_4}^{(4)} \circ \mathbf{b}_{r_5}^{(5)},$$

$$\mathbf{X}_{r_{67}}^{(67)} \cong \sum_{r_6=1}^{R_6} \sum_{r_7=1}^{R_7} g_{r_6,r_7,r_{67}}^{(67)} \, \mathbf{b}_{r_6}^{(6)} \circ \mathbf{b}_{r_7}^{(7)}.$$

The TT/HT decompositions lead naturally to a distributed Tucker decomposition, 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 2.15. Figure 2.16 illustrates exemplary HT/TT structures for data tensors of various orders (Tobler, 2012; Kressner and Tobler, 2014). Note that for a 3rd-order tensor, there is only one HT tensor network representation, while for a 5th-order we have 5, and for a 10th-order tensor there are 11 possible HT architectures.

A simple approach to reduce the size of a large-scale core tensor in the standard Tucker decomposition (typically, for $N > 5$) would be to apply the concept of distributed tensor networks (DTNs). The
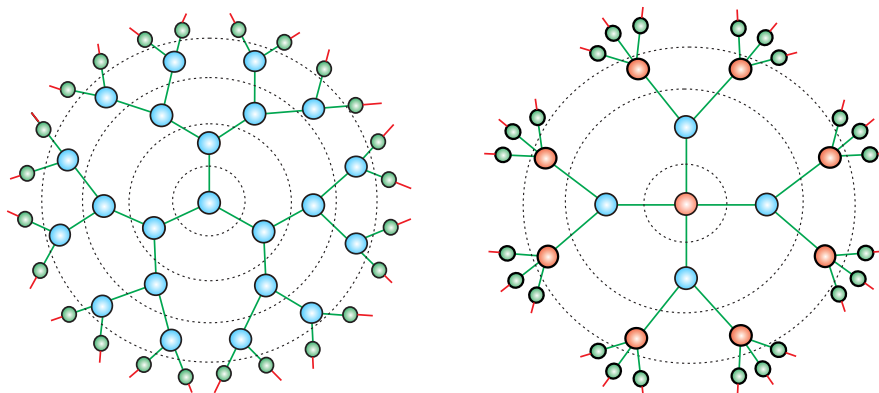
Figure 2.18: The Tree Tensor Network State (TTNS) with 3rd-order and 4th-order cores for the representation of 24th-order data tensors. The TTNS can be considered both as a generalization of HT/TT format and as a distributed model for the Tucker-*N* decomposition (see Section 3.3).

DTNs assume two kinds of cores (blocks): (i) the internal cores (nodes) which are connected only to other cores and have no free edges and (ii) external cores which do have free edges representing physical modes (indices) of a given data tensor (see also Section 2.2.4). Such distributed representations of tensors are not unique.

The tree tensor network state (TTNS) model, whereby all nodes are of 3rd-order or higher, can be considered as a generalization of the TT/HT decompositions, as illustrated by two examples in Figure 2.18 (Nakatani and Chan, 2013). A more detailed mathematical description of the TTNS is given in Section 3.3.

2.2.2 Tensor Train (TT) Network

The Tensor Train (TT) format can be interpreted as a special case of the HT format, where all nodes (TT-cores) of the underlying tensor network are connected in cascade (or train), i.e., they are aligned while factor matrices corresponding to the leaf modes are assumed to be identities and thus need not be stored. The TT format was first proposed in numerical analysis and scientific computing in (Oseledets, 2011; Oseledets and Tyrtyshnikov, 2009). Figure 2.19 presents the concept of
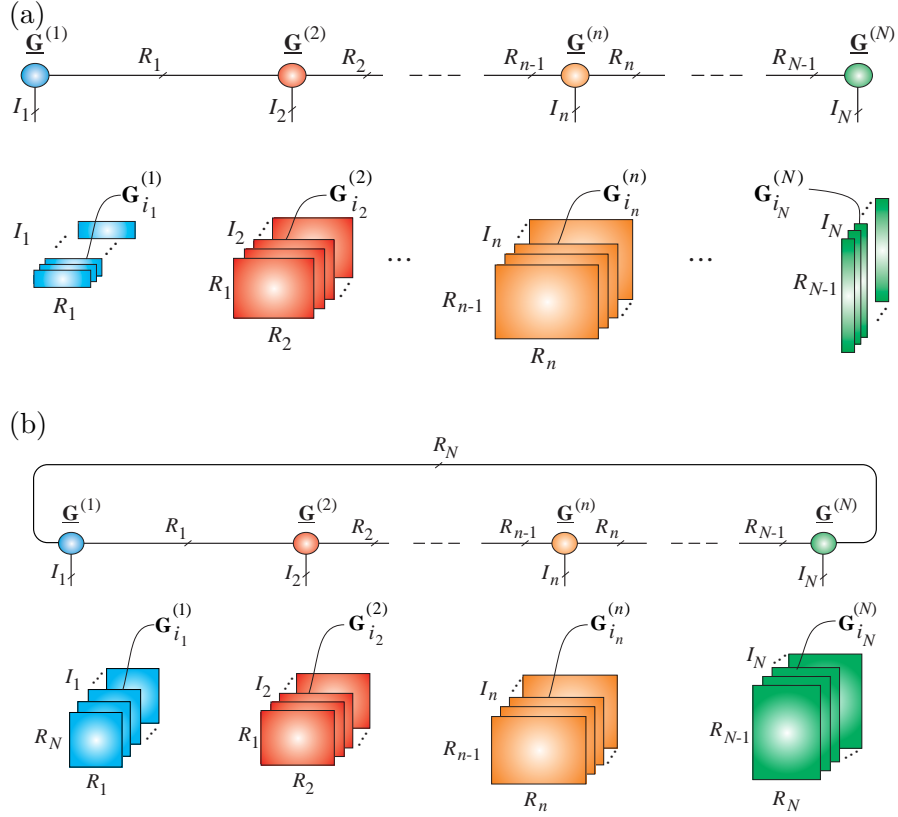
Figure 2.19: Concepts of the tensor train (TT) and tensor chain (TC) decompositions (MPS with OBC and PBC, respectively) for an Nth-order data tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. (a) Tensor Train (TT) can be mathematically described as $x_{i_1,i_2,\ldots,i_N} = \mathbf{G}^{(1)}_{i_1} \mathbf{G}^{(2)}_{i_2} \cdots \mathbf{G}^{(N)}_{i_N}$, where (bottom panel) 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 $R_0 = R_N = 1$. (b) For the Tensor Chain (TC), the entries of a tensor are expressed as $x_{i_1,i_2,\ldots,i_N} = \mathrm{tr}\left(\mathbf{G}^{(1)}_{i_1} \mathbf{G}^{(2)}_{i_2} \cdots \mathbf{G}^{(N)}_{i_N}\right) = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_N=1}^{R_N} g^{(1)}_{r_N, i_1, r_1} g^{(2)}_{r_1, i_2, r_2} \cdots g^{(N)}_{r_{N-1}, i_N, r_N}$, where (bottom panel) the lateral slices of the TC-cores are defined as $\mathbf{G}^{(n)}_{i_n} = \underline{\mathbf{G}}^{(n)}(:, i_n, :) \in \mathbb{R}^{R_{n-1} \times R_n}$ 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 $R_0 = R_N > 1$. Notice that TC/MPS is effectively a TT with a single loop connecting the first and the last core, so that all TC-cores are of 3rd-order.

TT decomposition for an Nth-order tensor, the entries of which can be computed as a cascaded (multilayer) multiplication of appropriate matrices (slices of TT-cores). The weights of internal edges (denoted by $\{R_1, R_2, \ldots, R_{N-1}\}$) represent the TT-rank. In this way, the so aligned sequence of core tensors represents a "tensor train" where the role of "buffers" is played by TT-core connections. 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 2.20 (for more detail see Section 4).

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 (Affleck *et al.*, 1987). It was subsequently extended by many researchers[6] (see (White, 1993; Vidal, 2003; Perez-Garcia *et al.*, 2007; Verstraete *et al.*, 2008; Schollwöck, 2013; Huckle *et al.*, 2013; Orús, 2014) and references therein).

Advantages of TT formats. An important advantage of the TT/MPS format over the HT format is its simpler practical implementation, as no binary tree needs to be determined (see Section 4). Another attractive property of the TT-decomposition is its simplicity when performing 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 the TT format is given in Section 4.5. Moreover, only

---

[6]In fact, the TT was rediscovered several times under different names: MPS, valence bond states, and density matrix renormalization group (DMRG) (White, 1993). The DMRG usually refers not only to a tensor network format but also the efficient computational algorithms (see also (Schollwöck, 2011; Hubig *et al.*, 2015) and references therein). Also, in quantum physics the ALS algorithm is called the one-site DMRG, while the Modified ALS (MALS) is known as the two-site DMRG (for more detail, see Part 2).
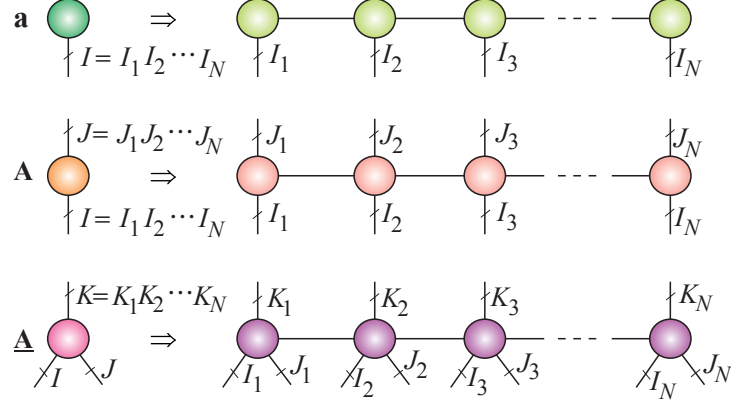
Figure 2.20: Forms of tensor train decompositions for a vector, $\mathbf{a} \in \mathbb{R}^I$, matrix, $\mathbf{A} \in \mathbb{R}^{I \times J}$, and 3rd-order tensor, $\underline{\mathbf{A}} \in \mathbb{R}^{I \times J \times K}$ (by applying a suitable tensorization).

TT-cores need to be stored and processed, which makes the number of parameters to scale linearly in the tensor order, N, of a data tensor and all mathematical operations are then performed only on the low-order and relatively small size core tensors.

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

$$\text{rank}_{\text{TT}}(\underline{\mathbf{X}}) = \mathbf{r}_{TT} = \{R_1, \ldots, R_{N-1}\}, \quad R_n = \text{rank}(\mathbf{X}_{<n>}), \quad (2.21)$$

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 $\underline{\mathbf{X}}$. 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 for a given raw data tensor.

The number of data samples to be stored scales linearly in the tensor order, N, and the size, I, and quadratically in the maximum TT rank bound, R, that is

$$\sum_{n=1}^{N} R_{n-1} R_n I_n \sim \mathcal{O}(NR^2I), \quad R := \max_{n}\{R_n\}, \quad I := \max_{n}\{I_n\}. \,(2.22)$$

This is why it is crucially important to have low-rank TT approximations[7]. A drawback of the TT format is that the ranks of a tensor train

---

[7]In the worst case scenario the TT ranks can grow up to $I^{(N/2)}$ for an Nth-order tensor.

MPS　　　　　　　　　　　　MPO

PEPS　　　　　　　　　　　PEPO
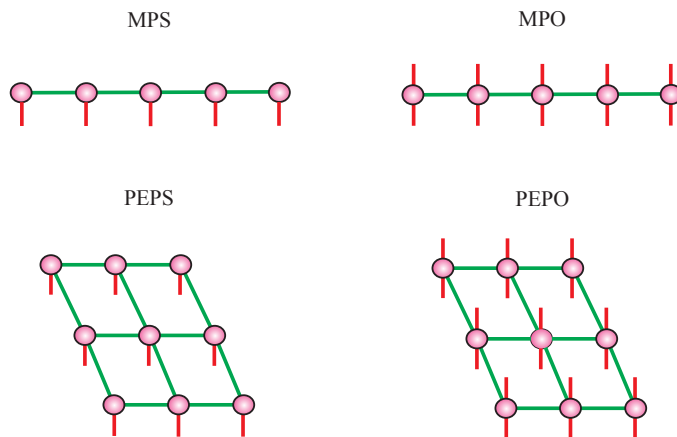
Figure 2.21: Class of 1D and 2D tensor train networks with open boundary conditions (OBC): the Matrix Product State (MPS) or (vector) Tensor Train (TT), the Matrix Product Operator (MPO) or Matrix TT, the Projected Entangled-Pair States (PEPS) or Tensor Product State (TPS), and the Projected Entangled-Pair Operators (PEPO).

decomposition depend on the ordering (permutation) of the modes, which gives different size of cores for different ordering. To solve this challenging permutation problem, we can estimate mutual information between individual TT cores pairwise (see (Barcza *et al.*, 2011; Ehlers *et al.*, 2015)). The procedure can be arranged in the following three steps: (i) Perform a rough (approximate) TT decomposition with relative low TT-rank and calculate mutual information between all pairs of cores, (ii) order TT cores in such way that the mutual information matrix is close to a diagonal matrix, and finally, (iii) perform TT decomposition again using the so optimised order of TT cores (see also Part 2).

2.2.3　Tensor Networks with Cycles: 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 and computation 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. The ranks can be often kept under control through hierarchical two-dimensional TT models called the PEPS (Projected Entangled Pair States[8]) and PEPO (Projected Entangled Pair Operators) tensor networks, which contain cycles, as shown in Figure 2.21. 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 order (Verstraete *et al.*, 2008; Evenbly and Vidal, 2009; Schuch *et al.*, 2010).

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 2.22) (Giovannetti *et al.*, 2008; Orús, 2014; Matsueda, 2016). 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 cycles.

Compared with the PEPS and PEPO formats, the main advantage of the 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. Moreover, TNs with cycles, especially the MERA tensor network allow us to model more complex functions and interactions between variables.

---

[8]An "entangled pair 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 (Orús, 2014; Handschuh, 2015).

(a)   (b)
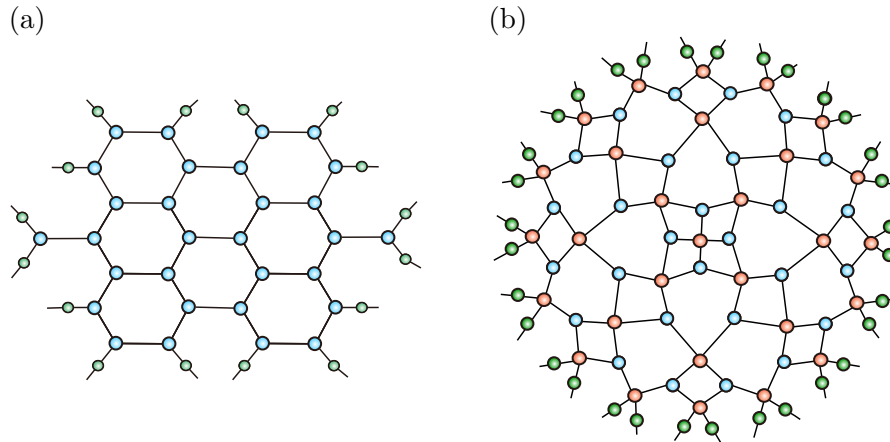
Figure 2.22: Examples of TN architectures with loops. (a) Honey-Comb Lattice (HCL) for a 16th-order tensor. (b) MERA for a 32th-order tensor.

2.2.4 Concatenated (Distributed) Representation of TT Networks

Complexity of algorithms for computation (contraction) on tensor networks typically scales 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. A step towards reducing storage and computational requirements would be therefore to reduce the size (volume) of core tensors by increasing their number through distributed tensor networks (DTNs), as illustrated in Figure 2.22. The underpinning idea is that each core tensor in an original TN is replaced by another TN (see Figure 2.23 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 (Hübener *et al.*, 2010). 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 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 reduction of the total number of network parameters through distributed computing. However,
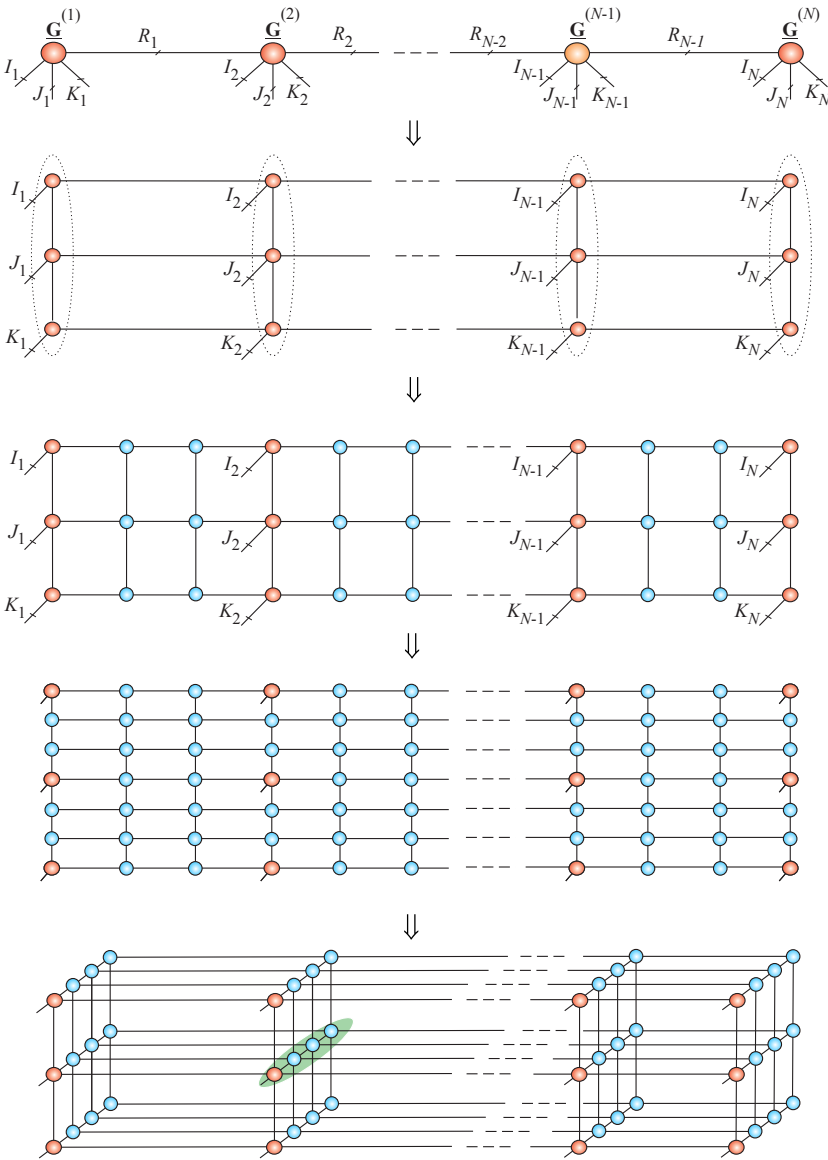
Figure 2.23: Graphical representation of a large-scale data tensor via its TT model (top panel), the PEPS model of the TT (third panel), and its transformation to a distributed 2D (second from bottom panel) and 3D (bottom panel) tensor train networks.

Table 2.2: 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 analogies.

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

compared to initial tree structures, the contraction of the resulting distributed tensor network becomes much more difficult because of the loops in the architecture.

2.2.5 Links between TNs and Machine Learning Models

Table 2.2 summarizes the conceptual connections of tensor networks with graphical and neural network models in machine learning and statistics (Morton, 2012; Critch, 2013; Critch and Morton, 2014; Kazeev *et al.*, 2014; Novikov and Rodomanov, 2014; Yang and Hospedales, 2016; Cohen *et al.*, 2016; Cohen and Shashua, 2016; Evenbly and White, 2016). More research is needed to establish deeper and more precise relationships.

2.2.6 Changing the Structure of Tensor Networks

An advantage of the graphical (graph) representation of tensor networks is that the graphs allow us to perform complex mathematical operations on core tensors in an intuitive and easy to understand way,

without the need to resort to complicated mathematical expressions. 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 (Zhao *et al.*, 2010; Hübener *et al.*, 2010; Handschuh, 2015). In particular:

- A change in topology to a HT/TT tree structure provides reduced computational complexity, through sequential contractions of core tensors and enhanced stability of the corresponding algorithms;

- Topology of TNs with cycles can be modified so as to completely eliminate the cycles or to reduce their number;

- Even for vastly diverse original data tensors, topology modifications may produce identical or similar TN structures which make it easier to compare and jointly analyze block of interconnected data tensors. This provides opportunity to perform joint group (linked) analysis of tensors by decomposing them to TNs.

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.

Tensor Network transformations. In order to modify tensor network 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 into new core tensors, as illustrated in Figures 2.24.

The example in Figure 2.24(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

(a)

(b)
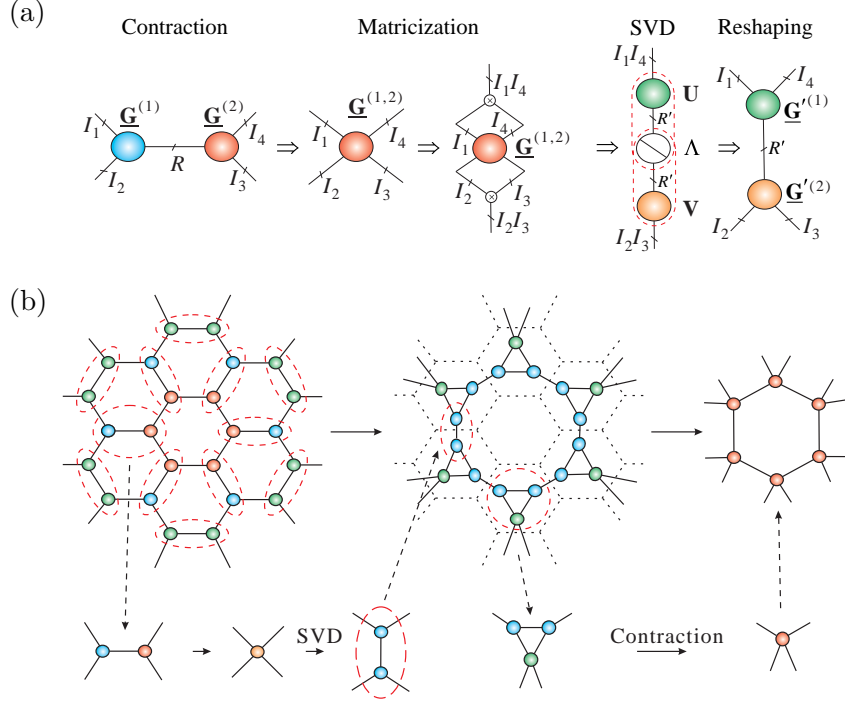
Figure 2.24: Illustration of basic transformations on a tensor network. (a) Contraction, matricization, matrix factorization (SVD) and reshaping of matrices back into tensors. (b) Transformation of a Honey-Comb lattice into a Tensor Chain (TC) via tensor contractions and the SVD.

performed to give the tensor

$$\underline{\mathbf{G}}^{(1,2)} = \underline{\mathbf{G}}^{(1)} \times^1 \underline{\mathbf{G}}^{(2)} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4}, \tag{2.23}$$

with entries $g_{i_1,i_2,i_3,i_4}^{(1,2)} = \sum_{r=1}^{R} g_{i_1,i_2,r}^{(1)} \, g_{r,i_3,i_4}^{(2)}$. 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

$$\mathbf{G}_{\overline{i_1 i_4}, \overline{i_2 i_3}}^{(1,2)} \cong \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}} \in \mathbb{R}^{I_1 I_4 \times I_2 I_3}. \tag{2.24}$$

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 $\underline{\mathbf{G}}^{'(2)} \in \mathbb{R}^{R' \times I_2 \times I_3}$.
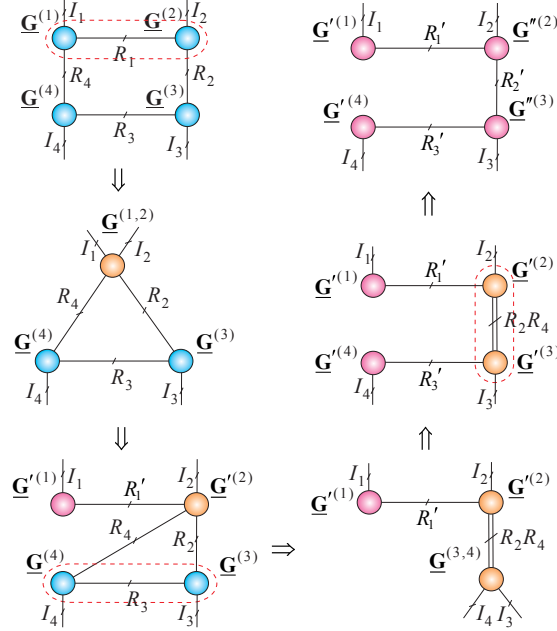
Figure 2.25: Transformation of the closed-loop Tensor Chain (TC) into the open-loop Tensor Train (TT). This is achieved by suitable contractions, reshaping and decompositions of core tensors.

The above tensor transformation procedure is quite general, and is applied in Figure 2.24(b) to transform a Honey-Comb lattice into a tensor chain (TC), while Figure 2.25 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

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

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

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

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 2.25.

2.3 Generalized Tensor Network Formats

The fundamental TNs considered so far assume that the links between the cores are expressed by tensor contractions. In general, links between the core tensors (or tensor sub-networks) 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. For example, the use of the outer product leads to Block Term Decomposition (BTD) (De Lathauwer, 2008; De Lathauwer and Nion, 2008; De Lathauwer, 2011; Sorber *et al.*, 2013) and use the Kronecker products yields to the Kronecker Tensor Decomposition (KTD) (Ragnarsson, 2012; Phan *et al.*, 2012, 2013b). Block term decompositions (BTD) are closely related to constrained Tucker formats (with a sparse block Tucker core) and the Hierarchical Outer Product Tensor Approximation (HOPTA), which be employed for very high-order data tensors (Cichocki, 2014).

Figure 2.26 illustrates such a BTD model for a 6th-order tensor, where the links between the components are expressed via outer products, while Figure 2.27 shows a more flexible Hierarchical Outer Product Tensor Approximation (HOPTA) model suitable for very high-order tensors.

Observe that the fundamental operator in the HOPTA generalized tensor networks is outer (tensor) product, which for two tensors $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times \cdots \times J_M}$, of arbitrary orders N and M, is defined as an $(N + M)$th-order 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 $c_{i_1,\ldots,i_N, j_1,\ldots,j_M} = a_{i_1,\ldots,i_N} \, b_{j_1,\ldots,j_M}$. This standard outer product of two tensors can be generalized to a nonlinear outer product as follows

$$\left(\underline{\mathbf{A}} \circ_f \underline{\mathbf{B}} \right)_{i_1,\ldots,i_N, j_1,\ldots,J_M} = f \left(a_{i_1,\ldots,i_N}, b_{j_1,\ldots,j_M} \right), \qquad (2.25)$$

where $f(\cdot, \cdot)$ is a suitably designed nonlinear function with associative and commutative properties. In a similar way, we can define other nonlinear tensor products, for example, Hadamard, Kronecker or Khatri–Rao products and employ them in generalized nonlinear tensor
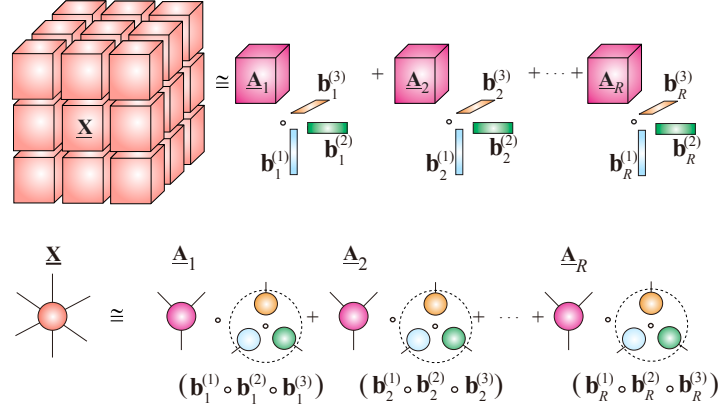
Figure 2.26: Block term decomposition (BTD) of a 6th-order block tensor, to yield $\underline{\mathbf{X}} = \sum_{r=1}^{R} \underline{\mathbf{A}}_r \circ \left(\mathbf{b}_r^{(1)} \circ \mathbf{b}_r^{(2)} \circ \mathbf{b}_r^{(3)} \right)$ (top panel), for more detail see (De Lathauwer, 2008; Sorber *et al.*, 2013). BTD in the tensor network notation (bottom panel). Therefore, the 6th-order tensor $\underline{\mathbf{X}}$ is approximately represented as a sum of R terms, each of which is an outer product of a 3rd-order tensor, $\underline{\mathbf{A}}_r$, and another a 3rd-order, rank-1 tensor, $\mathbf{b}_r^{(1)} \circ \mathbf{b}_r^{(2)} \circ \mathbf{b}_r^{(3)}$ (in dashed circle), which itself is an outer product of three vectors.

networks. The advantage of the HOPTA model over other TN models is its flexibility 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

$$\text{BTD} : \underline{\mathbf{X}} \cong \sum_{r=1}^{R} (\underline{\mathbf{A}}_r \circ \underline{\mathbf{B}}_r), \tag{2.26}$$

$$\text{KTD} : \underline{\tilde{\mathbf{X}}} \cong \sum_{r=1}^{R} (\underline{\mathbf{A}}_r \otimes \underline{\mathbf{B}}_r), \tag{2.27}$$

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

Note that the $2N$th-order subtensors, $\underline{\mathbf{A}}_r \circ \underline{\mathbf{B}}_r$ and $\underline{\mathbf{A}}_r \otimes \underline{\mathbf{B}}_r$, have the same elements, just arranged differently. For example, if $\underline{\mathbf{X}} = \underline{\mathbf{A}} \circ \underline{\mathbf{B}}$
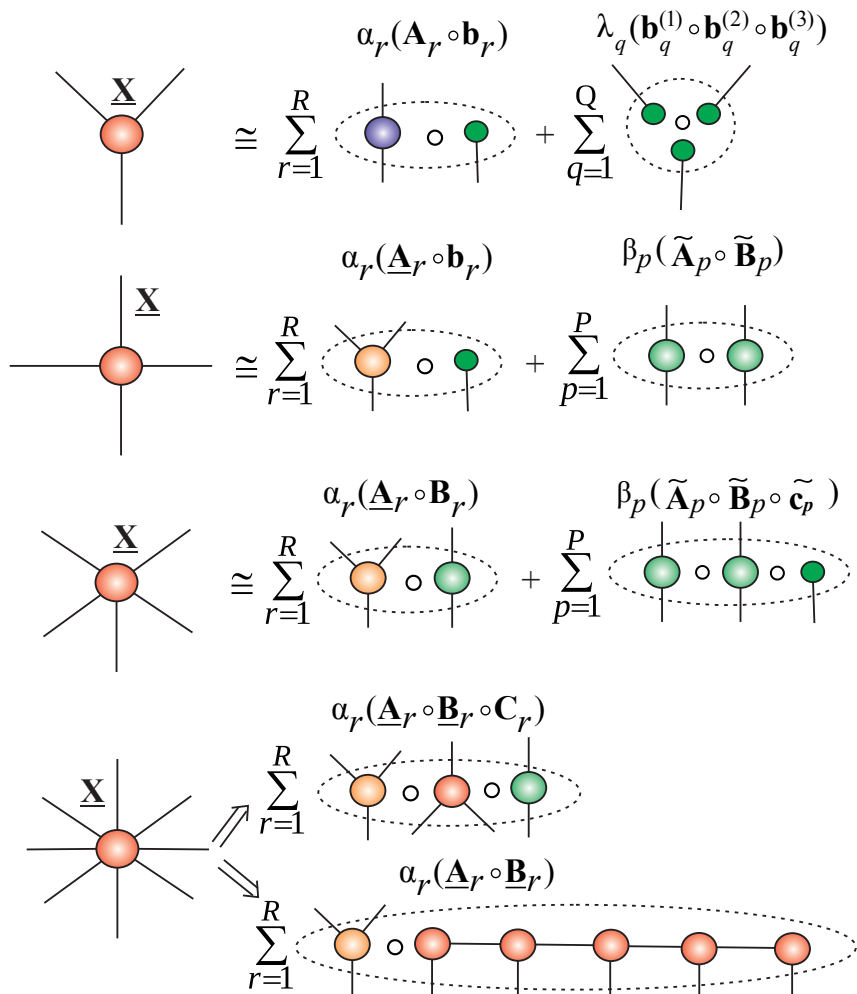
Figure 2.27: Conceptual model of the HOPTA generalized tensor network, illustrated for data tensors of different orders. For simplicity, we use the standard outer (tensor) products, but conceptually nonlinear outer products (see Eq. (2.25) and other tensor product operators (Kronecker, Hadamard) can also be employed. Each component (core tensor), $\underline{\mathbf{A}}_r$, $\underline{\mathbf{B}}_r$ and/or $\underline{\mathbf{C}}_r$, can be further hierarchically decomposed using suitable outer products, so that the HOPTA models can be applied to very high-order tensors.

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. 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 (2.27) represents 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 (2.27) expresses the rank-$(L_r \circ 1)$ BTD;

- 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. Therefore, the BTD model in (2.27) can also be used for recursive TT-decompositions.

The generalized tensor network approach caters for a large variety of tensor decomposition models, which may find applications in scientific computing, signal processing or deep learning (see, eg., (De Lathauwer, 2011; Cichocki, 2013b, 2014; Phan *et al.*, 2015b; Cohen and Shashua, 2016)).

In this monograph, we will mostly focus on the more established Tucker and TT decompositions (and some of their extensions), due to their conceptual simplicity, availability of stable and efficient algorithms for their computation 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.

3

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 to observed mixed signals in order to extract components with specific properties and/or estimate the mixing matrix (Cichocki and Amari, 2003; Cichocki *et al.*, 2009; Comon and Jutten, 2010; De la Torre, 2012; Hyvärinen, 2013). 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, by the diversities, we refer to different characteristics, features or morphology of latent variables which allow us to extract the desired components or features, for example, sparse or statistically independent components.

3.1 Constrained Low-Rank Matrix Factorizations

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

$$\mathbf{X} = \mathbf{A}\mathbf{\Lambda}\mathbf{B}^{\mathrm{T}} + \mathbf{E} = \sum_{r=1}^{R} \lambda_r \, \mathbf{a}_r \circ \mathbf{b}_r + \mathbf{E} = \sum_{r=1}^{R} \lambda_r \, \mathbf{a}_r \, \mathbf{b}_r^{\mathrm{T}} + \mathbf{E}, \qquad (3.1)$$

where $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_R)$ 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 (3.1), $\mathbf{X} \in \mathbb{R}^{I \times J}$ 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 depending on application, $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_R] \in \mathbb{R}^{J \times R}$, is the matrix of unknown components, factors, latent variables, or hidden sources, represented by vectors $\mathbf{b}_r \in \mathbb{R}^J$ (see Figure 3.2).

It should be noted that 2-way CA has an inherent symmetry. Indeed, Eq. (3.1) could also be written as $\mathbf{X}^{\mathrm{T}} \approx \mathbf{B}\mathbf{A}^{\mathrm{T}}$, 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) (Bach and Jordan, 2003; Cichocki *et al.*, 2009; Bruckstein *et al.*, 2009; Comon and Jutten, 2010; Kauppi *et al.*, 2015; Yokota *et al.*, 2016). 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.

Singular value decomposition (SVD) of the data matrix $\mathbf{X} \in \mathbb{R}^{I \times J}$ is a special, very important, case of the factorization in Eq. (3.1), and is given by

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}} = \sum_{r=1}^{R} \sigma_r \, \mathbf{u}_r \circ \mathbf{v}_r = \sum_{r=1}^{R} \sigma_r \, \mathbf{u}_r \mathbf{v}_r^{\mathrm{T}}, \qquad (3.2)$$

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[1]. 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

$$\mathbf{X}_k \approx \mathbf{A}_k \mathbf{B}_k^{\mathrm{T}}, \qquad (k = 1, 2, \dots, K), \tag{3.3}$$

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 (3.3) can be related to models of group PCA/ICA through suitable pre-processing, dimensionality reduction and post-processing procedures (Esposito *et al.*, 2005; Groves *et al.*, 2011; Cichocki, 2013a; Smith *et al.*, 2014; Zhou *et al.*, 2016b). 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) 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 (Bach and Jordan, 2003; Cichocki and Amari, 2003; Comon and Jutten, 2010; Zhou *et al.*, 2012; Hyvärinen, 2013).

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. We proceed to show how constrained matrix factorizations or 2-way CA models can

---

[1](Mirsky, 1960) has generalized this optimality to arbitrary unitarily invariant norms.

be extended to multilinear models using tensor decompositions, such as the Canonical Polyadic (CP) and the Tucker decompositions, as illustrated in Figures 3.1, 3.2 and 3.3.

3.2 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 (Hitchcock, 1927; Harshman, 1970; Carroll and Chang, 1970)

$$
\begin{aligned}
\underline{\mathbf{X}} &\cong \sum_{r=1}^{R} \lambda_r \, \mathbf{b}_r^{(1)} \circ \mathbf{b}_r^{(2)} \circ \cdots \circ \mathbf{b}_r^{(N)} \\
&= \underline{\mathbf{\Lambda}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)} \\
&= [\![\underline{\mathbf{\Lambda}}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N)}]\!],
\end{aligned}
\tag{3.4}
$$

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 matrices (see Figure 3.1 and Figure 3.2).

Via the Khatri–Rao products (see Table 2.1), the CP decomposition can be equivalently expressed in a matrix/vector form as

$$
\begin{aligned}
\mathbf{X}_{(n)} &\cong \mathbf{B}^{(n)} \mathbf{\Lambda} (\mathbf{B}^{(N)} \odot \cdots \odot \mathbf{B}^{(n+1)} \odot \mathbf{B}^{(n-1)} \odot \cdots \odot \mathbf{B}^{(1)})^{\mathrm{T}} \\
&= \mathbf{B}^{(n)} \mathbf{\Lambda} (\mathbf{B}^{(1)} \odot_L \cdots \odot_L \mathbf{B}^{(n-1)} \odot_L \mathbf{B}^{(n+1)} \odot_L \cdots \odot_L \mathbf{B}^{(N)})^{\mathrm{T}}
\end{aligned}
\tag{3.5}
$$

and

$$
\begin{aligned}
\mathrm{vec}(\underline{\mathbf{X}}) &\cong [\mathbf{B}^{(N)} \odot \mathbf{B}^{(N-1)} \odot \cdots \odot \mathbf{B}^{(1)}] \, \boldsymbol{\lambda} \\
&\cong [\mathbf{B}^{(1)} \odot_L \mathbf{B}^{(2)} \odot_L \cdots \odot_L \mathbf{B}^{(N)}] \, \boldsymbol{\lambda},
\end{aligned}
\tag{3.6}
$$

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

The rank of a tensor $\underline{\mathbf{X}}$ is defined as the smallest R for which the CP decomposition in (3.4) holds exactly.

Algorithms to compute CP decomposition. In real world applications, the signals of interest are corrupted by noise, so that the CP

(a) Standard block diagram for CP decomposition of a 3rd-order tensor
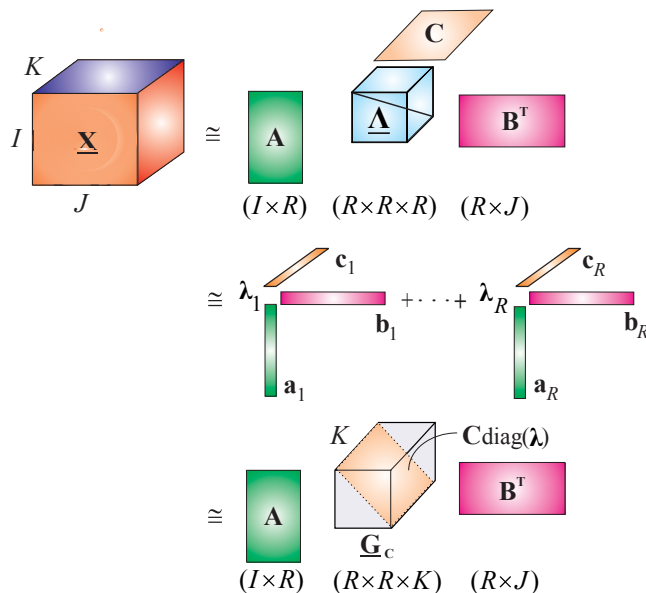
(b) CP decomposition for a 4th-order tensor in the tensor network notation
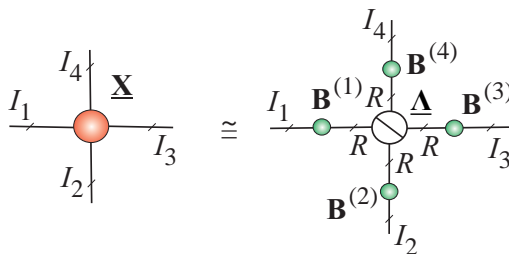
Figure 3.1: Representations of the CP decomposition. The objective of the CP decomposition is to estimate the factor matrices $\mathbf{B}^{(n)} \in \mathbb{R}^{I_n \times R}$ and scaling coefficients $\{\lambda_1, \lambda_1, \ldots, \lambda_R\}$. (a) The CP decomposition of a 3rd-order tensor in the 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 $\underline{\mathbf{G}}_c = \underline{\mathbf{\Lambda}} \times_3 \mathbf{C}$. (b) The CP decomposition for a 4th-order tensor in the form $\underline{\mathbf{X}} \cong \underline{\mathbf{\Lambda}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \times_3 \mathbf{B}^{(3)} \times_4 \mathbf{B}^{(4)} = \sum_{r=1}^{R} \lambda_r \, \mathbf{b}_r^{(1)} \circ \mathbf{b}_r^{(2)} \circ \mathbf{b}_r^{(3)} \circ \mathbf{b}_r^{(4)}$.
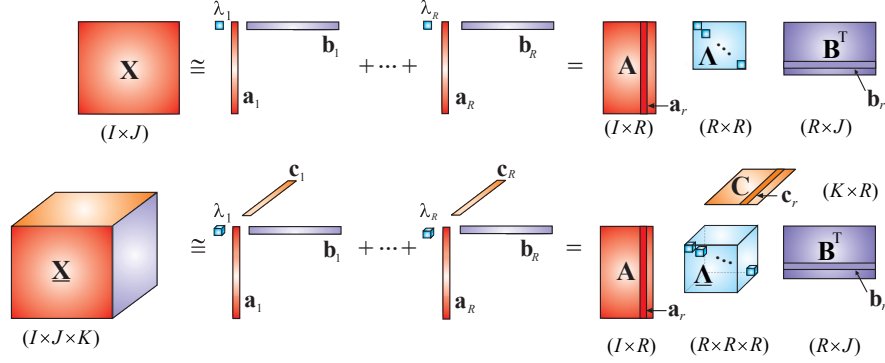
Figure 3.2: Analogy between a low-rank matrix factorization, $\mathbf{X} \cong \mathbf{A}\mathbf{\Lambda}\mathbf{B}^{\mathrm{T}} = \sum_{r=1}^{R} \lambda_r \, \mathbf{a}_r \circ \mathbf{b}_r$ (top), and a simple low-rank tensor factorization (CP decomposition), $\underline{\mathbf{X}} \cong \underline{\mathbf{\Lambda}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} = \sum_{r=1}^{R} \lambda_r \, \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$ (bottom).

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

$$J_2(\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N)}) = \|\underline{\mathbf{X}} - [\![\mathbf{\Lambda}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N)}]\!]\|_F^2, \quad (3.7)$$

or Least Absolute Error (LAE) criteria (Vorobyov *et al.*, 2005)

$$J_1(\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N)}) = \|\underline{\mathbf{X}} - [\![\mathbf{\Lambda}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N)}]\!]\|_1. \quad (3.8)$$

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 (Harshman, 1970; Kolda and Bader, 2009).

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 in Eq. (3.5), the component matrices, $\mathbf{B}^{(n)}$, can be updated sequentially as

$$\mathbf{B}^{(n)} \leftarrow \mathbf{X}_{(n)} \left(\bigodot_{k \neq n} \mathbf{B}^{(k)} \right) \left(\circledast_{k \neq n} (\mathbf{B}^{(k)\,\mathrm{T}} \mathbf{B}^{(k)}) \right)^{\dagger}. \quad (3.9)$$

The main challenge (or bottleneck) in implementing ALS and Gradient Decent (GD) techniques for CP decomposition lies therefore in

---

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:     $\mathbf{A} \leftarrow \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})(\mathbf{C}^{\mathrm{T}}\mathbf{C} \circledast \mathbf{B}^{\mathrm{T}}\mathbf{B})^{\dagger}$

4:     Normalize column vectors of $\mathbf{A}$ to unit length (by computing the norm of each column vector and dividing each element of a vector by its norm)

5:     $\mathbf{B} \leftarrow \mathbf{X}_{(2)}(\mathbf{C} \odot \mathbf{A})(\mathbf{C}^{\mathrm{T}}\mathbf{C} \circledast \mathbf{A}^{\mathrm{T}}\mathbf{A})^{\dagger}$

6:     Normalize column vectors of $\mathbf{B}$ to unit length

7:     $\mathbf{C} \leftarrow \mathbf{X}_{(3)}(\mathbf{B} \odot \mathbf{A})(\mathbf{B}^{\mathrm{T}}\mathbf{B} \circledast \mathbf{C}^{\mathrm{T}}\mathbf{C})^{\dagger}$

8:     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 $\boldsymbol{\lambda}$.

---

multiplying a matricized tensor and Khatri–Rao product (of factor matrices) (Phan *et al.*, 2013a; Choi and Vishwanathan, 2014) and in the computation of the pseudo-inverse of $(R \times R)$ matrices (for the basic ALS see Algorithm 1).

The ALS approach is attractive for its simplicity, and often provides satisfactory performance for well defined problems with high SNRs and well separated and non-collinear components. For ill-conditioned problems, advanced algorithms are required which 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 (Phan *et al.*, 2013c; Sorber *et al.*, 2013; Phan *et al.*, 2015a). Implementation of parallel ALS algorithm over distributed memory for very large-scale tensors was proposed in (Choi and Vishwanathan, 2014; Karlsson *et al.*, 2016).

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, algorithms for tensor decompositions have generally 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 need to either: (i) reshape the tensor at hand into a set of matrices using traditional matricizations, (ii) employ reduced randomized unfolding matrices, or (iii) perform suitable random multiple projections of a data tensor onto lower-dimensional subspaces. 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 (Kuleshov *et al.*, 2015).

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, $\mathbf{X}_{\bar{3}} = \underline{\mathbf{X}} \ \bar{\times}_3 \ \boldsymbol{\omega}_3 \in \mathbb{R}^{I_1 \times I_2}$, $\mathbf{X}_{\bar{2}} = \underline{\mathbf{X}} \ \bar{\times}_2 \ \boldsymbol{\omega}_2 \in \mathbb{R}^{I_1 \times I_3}$ 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 a tensor of any order to matrices and consequently transform the CP decomposition problem to constrained matrix factorizations problem, which can be solved via simultaneous (joint) matrix diagonalization (De Lathauwer, 2006; Chabriel *et al.*, 2014). It was shown that 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[2] that plagued earlier tensor-to-matrix reductions. Although a uniform random sampling may experience problems for tensors with spiky elements, it often outperforms the standard CP-ALS decomposition algorithms.

Alternative algorithms for the CP decomposition of huge-scale tensors include tensor sketching – a random mapping technique, which exploits kernel methods and regression (Pham and Pagh, 2013; Wang *et al.*, 2015), and the class of distributed algorithms such as the

---

[2]In linear algebra, the eigen-gap of a linear operator is the difference between two successive eigenvalues, where the eigenvalues are sorted in an ascending order.

DFacTo (Choi and Vishwanathan, 2014) and the GigaTensor which is based on Hadoop / MapReduce paradigm (Kang *et al.*, 2012).

Constraints. Under rather mild conditions, the CP decomposition is generally unique by itself (Kruskal, 1977; Sidiropoulos and Bro, 2000). It does not require additional constraints on the factor matrices to achieve uniqueness, which makes it a powerful and useful tool for tensor factroization. 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 into the algorithms to compute CPD and at the same time 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 (Sidiropoulos, 2004; Dhillon, 2009; Sørensen *et al.*, 2012; Kim *et al.*, 2014; Zhou and Cichocki, 2012b; Liavas and Sidiropoulos, 2015).

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 (Acar and Yener, 2009; Kolda and Bader, 2009; Mørup, 2011; Anandkumar *et al.*, 2014; Wang *et al.*, 2015; Tresp *et al.*, 2015; Sidiropoulos *et al.*, 2016). It is also routinely used in exploratory data analysis, where the rank-1 terms capture essential properties of dynamically complex datasets, while in wireless communication systems, signals transmitted by different users correspond to rank-1 terms in the case of line-of-sight propagation and therefore admit 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 (Sidiropoulos *et al.*, 2000; Sidiropoulos, 2001; Sørensen and De Lathauwer, 2013).

3.3 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:

$$
\begin{aligned}
\underline{\mathbf{X}} &\cong \sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} g_{r_1 r_2 \cdots r_N} \left(\mathbf{b}_{r_1}^{(1)} \circ \mathbf{b}_{r_2}^{(2)} \circ \cdots \circ \mathbf{b}_{r_N}^{(N)} \right) \\
&= \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)} \\
&= [\![\underline{\mathbf{G}}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N)}]\!],
\end{aligned}
\tag{3.10}
$$

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 3.3). The core tensor (typically, $R_n << I_n$) models a potentially complex pattern of mutual interaction between the vectors in different modes. The model in (3.10) is often referred to as the Tucker-N model.

The CP and Tucker decompositions have long history. For recent surveys and more detailed information we refer to (Kolda and Bader, 2009; Grasedyck *et al.*, 2013; Comon, 2014; Cichocki *et al.*, 2015b; Sidiropoulos *et al.*, 2016).

Using the properties of the Kronecker tensor product, the Tucker-N decomposition in (3.10) can be expressed in an equivalent matrix and vector form as

$$
\begin{aligned}
\mathbf{X}_{(n)} &\cong \mathbf{B}^{(n)} \mathbf{G}_{(n)} (\mathbf{B}^{(1)} \otimes_L \cdots \otimes_L \mathbf{B}^{(n-1)} \otimes_L \mathbf{B}^{(n+1)} \otimes_L \cdots \otimes_L \mathbf{B}^{(N)})^{\mathrm{T}} \\
&= \mathbf{B}^{(n)} \mathbf{G}_{(n)} (\mathbf{B}^{(N)} \otimes \cdots \otimes \mathbf{B}^{(n+1)} \otimes \mathbf{B}^{(n-1)} \otimes \cdots \otimes \mathbf{B}^{(1)})^{\mathrm{T}},
\end{aligned}
\tag{3.11}
$$

$$
\begin{aligned}
\mathbf{X}_{<n>} &\cong (\mathbf{B}^{(1)} \otimes_L \cdots \otimes_L \mathbf{B}^{(n)}) \, \mathbf{G}_{<n>} (\mathbf{B}^{(n+1)} \otimes_L \cdots \otimes_L \mathbf{B}^{(N)})^{\mathrm{T}} \\
&= (\mathbf{B}^{(n)} \otimes \cdots \otimes \mathbf{B}^{(1)}) \, \mathbf{G}_{<n>} (\mathbf{B}^{(N)} \otimes \mathbf{B}^{(N-1)} \otimes \cdots \otimes \mathbf{B}^{(n+1)})^{\mathrm{T}},
\end{aligned}
\tag{3.12}
$$

$$
\begin{aligned}
\mathrm{vec}(\underline{\mathbf{X}}) &\cong [\mathbf{B}^{(1)} \otimes_L \mathbf{B}^{(2)} \otimes_L \cdots \otimes_L \mathbf{B}^{(N)}] \, \mathrm{vec}(\underline{\mathbf{G}}) \\
&= [\mathbf{B}^{(N)} \otimes \mathbf{B}^{(N-1)} \otimes \cdots \otimes \mathbf{B}^{(1)}] \, \mathrm{vec}(\underline{\mathbf{G}}),
\end{aligned}
\tag{3.13}
$$

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

Table 3.1 and Table 3.2 summarize fundamental mathematical representations of CP and Tucker decompositions for 3rd-order and Nth-order tensors.

The Tucker decomposition is said to be in an *independent Tucker format* if all the factor matrices, $\mathbf{B}^{(n)}$, are full column rank, while a Tucker format is termed an *orthonormal format*, if in addition, all the factor matrices, $\mathbf{B}^{(n)} = \mathbf{U}^{(n)}$, are orthogonal. The standard Tucker model often has orthogonal factor matrices.

Multilinear rank. The multilinear rank of an Nth-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ corresponds to the N-tuple $(R_1, R_2, \ldots, R_N)$ consisting of the dimensions of the different subspaces. If the Tucker decomposition (3.10) holds exactly it is mathematically defined as

$$\mathrm{rank}_{ML}(\underline{\mathbf{X}}) = \{\mathrm{rank}(\mathbf{X}_{(1)}), \mathrm{rank}(\mathbf{X}_{(2)}), \ldots, \mathrm{rank}(\mathbf{X}_{(N)})\}, \quad (3.14)$$

with $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times I_1 \cdots I_{n-1} I_{n+1} \cdots I_N}$ for $n = 1, 2, \ldots, N$. Rank of the Tucker decompositions can be determined using information criteria (Yokota *et al.*, 2017), or through the number of dominant eigenvalues when an approximation accuracy of the decomposition or a noise level is given (see Algorithm 8).

The independent Tucker format has the following important properties if the equality in (3.10) holds exactly (see, e.g., (Jiang *et al.*, 2016) and references therein):

1. The 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 exactly the same, i.e.,

$$\mathrm{rank}_{CP}(\underline{\mathbf{X}}) = \mathrm{rank}_{CP}(\underline{\mathbf{G}}). \quad (3.15)$$

2. 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

$$R_n \leqslant \prod_{p \neq n}^{N} R_p \quad \forall n. \quad (3.16)$$

(a) Standard block diagrams of Tucker (top) and Tucker-CP (bottom) decompositions for a 3rd-order tensor
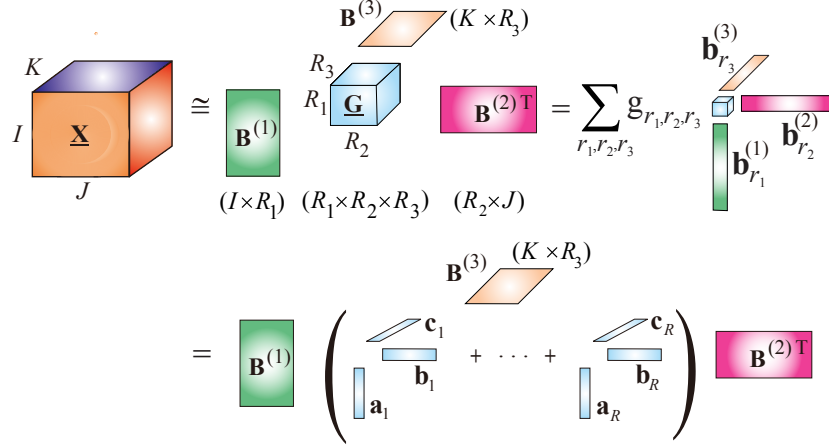
(b) The TN diagram for the Tucker and Tucker/CP decompositions of a 4th-order tensor
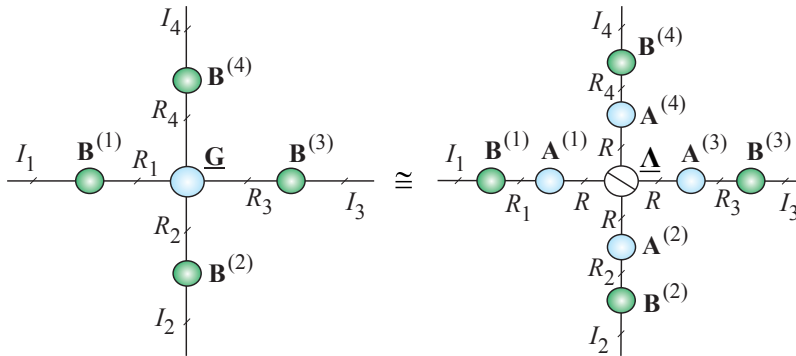
Figure 3.3: Illustration of the Tucker and Tucker-CP decompositions, where the objective is to compute the factor matrices, $\mathbf{B}^{(n)}$, and the core tensor, $\underline{\mathbf{G}}$. (a) Tucker decomposition of a 3rd-order tensor, $\underline{\mathbf{X}} \cong \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \times_3 \mathbf{B}^{(3)}$. In some applications, the core tensor can be further approximately factorized using the CP decomposition as $\underline{\mathbf{G}} \cong \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$ (bottom diagram), or alternatively using TT/HT decompositions. (b) Graphical representation of the Tucker-CP decomposition for a 4th-order tensor, $\underline{\mathbf{X}} \cong \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \times_3 \mathbf{B}^{(3)} \times_4 \mathbf{B}^{(4)} = [\![\underline{\mathbf{G}}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \mathbf{B}^{(3)}, \mathbf{B}^{(4)}]\!] \cong (\underline{\mathbf{\Lambda}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \mathbf{A}^{(3)} \times_4 \mathbf{A}^{(4)}) \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \times_3 \mathbf{B}^{(3)} \times_4 \mathbf{B}^{(4)} = [\![\underline{\mathbf{\Lambda}}; \mathbf{B}^{(1)}\mathbf{A}^{(1)}, \mathbf{B}^{(2)}\mathbf{A}^{(2)}, \mathbf{B}^{(3)}\mathbf{A}^{(3)}, \mathbf{B}^{(4)}\mathbf{A}^{(4)}]\!]$.

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

$$R_1 \leqslant \mathrm{rank}_{CP}(\underline{\mathbf{X}}) \leqslant R_2 R_3 \cdots R_N. \qquad (3.17)$$

3. If a data tensor is symmetric and admits 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 $\mathrm{rank}_{CP}(\underline{\mathbf{X}}) = \mathrm{rank}_{CP}(\underline{\mathbf{G}})$.

4. For the orthonormal Tucker format, that is, $\underline{\mathbf{X}} = [\![\underline{\mathbf{G}}; \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(N)}]\!] \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, with $\mathbf{U}^{(n)\mathrm{T}} \mathbf{U}^{(n)} = \mathbf{I}$, $\forall n$, the Frobenius norms and the Schatten p-norms[3] of the data tensor, $\underline{\mathbf{X}}$, and its core tensor, $\underline{\mathbf{G}}$, are equal, i.e.,

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

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 3.1). 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 (Zhou and Cichocki, 2012a).

The Tucker-N model, in which $(N-K)$ factor matrices are identity matrices is called the Tucker-(K, N) model. In the simplest scenario,

---

[3]The Schatten p-norm of an Nth-order tensor $\underline{\mathbf{X}}$ is defined as the average of the Schatten norms of mode-n unfoldings, 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 the rth 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 $\log \det(\mathbf{X}\mathbf{X}^{\mathrm{T}} + \varepsilon \mathbf{I})$, $\varepsilon > 0$.

Table 3.1: Different forms of CP and Tucker representations of a 3rd-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$, where $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_R]^{\mathrm{T}}$, and $\boldsymbol{\Lambda} = \mathrm{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_R\}$.

CP Decomposition	Tucker Decomposition

Scalar representation

$$x_{ijk} = \sum_{r=1}^{R} \lambda_r \, a_{ir} \, b_{jr} \, c_{kr} \qquad x_{ijk} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} g_{r_1 r_2 r_3} \, a_{i r_1} \, b_{j r_2} \, c_{k r_3}$$

Tensor representation, outer products

$$\underline{\mathbf{X}} = \sum_{r=1}^{R} \lambda_r \, \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \qquad \underline{\mathbf{X}} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} g_{r_1 r_2 r_3} \, \mathbf{a}_{r_1} \circ \mathbf{b}_{r_2} \circ \mathbf{c}_{r_3}$$

Tensor representation, multilinear products

$$\begin{aligned} \underline{\mathbf{X}} &= \underline{\boldsymbol{\Lambda}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} \\ &= [\![\underline{\boldsymbol{\Lambda}}; \ \mathbf{A}, \ \mathbf{B}, \ \mathbf{C}]\!] \end{aligned} \qquad \begin{aligned} \underline{\mathbf{X}} &= \underline{\mathbf{G}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} \\ &= [\![\underline{\mathbf{G}}; \ \mathbf{A}, \ \mathbf{B}, \ \mathbf{C}]\!] \end{aligned}$$

Matrix representations

$$\begin{aligned} \mathbf{X}_{(1)} &= \mathbf{A} \, \boldsymbol{\Lambda} \, (\mathbf{B} \odot_L \mathbf{C})^{\mathrm{T}} \\ \mathbf{X}_{(2)} &= \mathbf{B} \, \boldsymbol{\Lambda} \, (\mathbf{A} \odot_L \mathbf{C})^{\mathrm{T}} \\ \mathbf{X}_{(3)} &= \mathbf{C} \, \boldsymbol{\Lambda} \, (\mathbf{A} \odot_L \mathbf{B})^{\mathrm{T}} \end{aligned} \qquad \begin{aligned} \mathbf{X}_{(1)} &= \mathbf{A} \, \mathbf{G}_{(1)} \, (\mathbf{B} \otimes_L \mathbf{C})^{\mathrm{T}} \\ \mathbf{X}_{(2)} &= \mathbf{B} \, \mathbf{G}_{(2)} \, (\mathbf{A} \otimes_L \mathbf{C})^{\mathrm{T}} \\ \mathbf{X}_{(3)} &= \mathbf{C} \, \mathbf{G}_{(3)} \, (\mathbf{A} \otimes_L \mathbf{B})^{\mathrm{T}} \end{aligned}$$

Vector representation

$$\mathrm{vec}(\underline{\mathbf{X}}) = (\mathbf{A} \odot_L \mathbf{B} \odot_L \mathbf{C}) \boldsymbol{\lambda} \qquad \mathrm{vec}(\underline{\mathbf{X}}) = (\mathbf{A} \otimes_L \mathbf{B} \otimes_L \mathbf{C}) \, \mathrm{vec}(\underline{\mathbf{G}})$$

Matrix slices $\mathbf{X}_k = \mathbf{X}(:,:,k)$

$$\mathbf{X}_k = \mathbf{A} \, \mathrm{diag}(\lambda_1 \, c_{k,1}, \ldots, \lambda_R \, c_{k,R}) \mathbf{B}^{\mathrm{T}} \qquad \mathbf{X}_k = \mathbf{A} \left(\sum_{r_3=1}^{R_3} c_{kr_3} \mathbf{G}(:,:,r_3) \right) \mathbf{B}^{\mathrm{T}}$$

Table 3.2: Different forms of CP and Tucker representations of an Nth-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$.

CP	Tucker

Scalar product

$$x_{i_1,\ldots,i_N} = \sum_{r=1}^{R} \lambda_r \, b_{i_1,r}^{(1)} \cdots b_{i_N,r}^{(N)} \qquad x_{i_1,\ldots,i_N} = \sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} g_{r_1,\ldots,r_N} \, b_{i_1,r_1}^{(1)} \cdots b_{i_N,r_N}^{(N)}$$

Outer product

$$\underline{\mathbf{X}} = \sum_{r=1}^{R} \lambda_r \, \mathbf{b}_r^{(1)} \circ \cdots \circ \mathbf{b}_r^{(N)} \qquad\qquad \underline{\mathbf{X}} = \sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} g_{r_1,\ldots,r_N} \, \mathbf{b}_{r_1}^{(1)} \circ \cdots \circ \mathbf{b}_{r_N}^{(N)}$$

Multilinear product

$$\underline{\mathbf{X}} = \underline{\mathbf{\Lambda}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)} \qquad \underline{\mathbf{X}} = \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)}$$

$$\underline{\mathbf{X}} = \left[\!\left[\underline{\mathbf{\Lambda}}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N)} \right]\!\right] \qquad \underline{\mathbf{X}} = \left[\!\left[\underline{\mathbf{G}}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N)} \right]\!\right]$$

Vectorization

$$\mathrm{vec}(\underline{\mathbf{X}}) = \left(\overset{1}{\underset{n=N}{\bigodot}} \mathbf{B}^{(n)} \right) \boldsymbol{\lambda} \qquad\qquad \mathrm{vec}(\underline{\mathbf{X}}) = \left(\overset{1}{\underset{n=N}{\bigotimes}} \mathbf{B}^{(n)} \right) \mathrm{vec}(\underline{\mathbf{G}})$$

Matricization

$$\mathbf{X}_{(n)} = \mathbf{B}^{(n)} \mathbf{\Lambda} \left(\overset{1}{\underset{m=N,\, m \neq n}{\bigodot}} \mathbf{B}^{(m)} \right)^{\mathrm{T}} \qquad \mathbf{X}_{(n)} = \mathbf{B}^{(n)} \mathbf{G}_{(n)} \left(\overset{1}{\underset{m=N,\, m \neq n}{\bigotimes}} \mathbf{B}^{(m)} \right)^{\mathrm{T}}$$

$$\mathbf{X}_{<n>} = (\overset{1}{\underset{m=n}{\bigodot}} \mathbf{B}^{(m)}) \mathbf{\Lambda} (\overset{n+1}{\underset{m=N}{\bigodot}} \mathbf{B}^{(m)})^{\mathrm{T}}, \qquad \mathbf{X}_{<n>} = (\overset{1}{\underset{m=n}{\bigotimes}} \mathbf{B}^{(m)}) \mathbf{G}_{<n>} (\overset{n+1}{\underset{m=N}{\bigotimes}} \mathbf{B}^{(m)})^{\mathrm{T}}$$

Slice representation

$$\underline{\mathbf{X}}(:,:,k_3) = \mathbf{B}^{(1)} \tilde{\mathbf{D}}_{k_3} \mathbf{B}^{(2)\,\mathrm{T}} \qquad\qquad \underline{\mathbf{X}}(:,:,k_3) = \mathbf{B}^{(1)} \tilde{\mathbf{G}}_{k_3} \mathbf{B}^{(2)\,\mathrm{T}}, \ k_3 = \overline{i_3 i_4 \cdots i_N}$$

$$\tilde{\mathbf{D}}_{k_3} = \mathrm{diag}(\tilde{d}_{11}, \ldots, \tilde{d}_{RR}) \in \mathbb{R}^{R \times R} \ \text{ with entries } \ \tilde{d}_{rr} = \lambda_r b_{i_3,r}^{(3)} \cdots b_{i_N,r}^{(N)}$$

$$\tilde{\mathbf{G}}_{k_3} = \sum_{r_3} \cdots \sum_{r_N} b_{i_3,r_3}^{(3)} \cdots b_{i_N,r_N}^{(N)} \mathbf{G}_{:,:,r_3,\ldots,r_N} \ \text{ is the sum of frontal slices.}$$

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[4]

$$\underline{\mathbf{X}} \cong \underline{\mathbf{G}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{I} = \underline{\mathbf{G}} \times_1 \mathbf{A} \times_2 \mathbf{B}, \quad (3.18)$$

or in an equivalent matrix form

$$\mathbf{X}_k = \mathbf{A}\mathbf{G}_k\mathbf{B}^{\mathrm{T}}, \qquad (k = 1, 2, \ldots, K), \quad (3.19)$$

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 $\mathbf{A} \in \mathbb{R}^{I \times R_1}$, $\mathbf{B} \in \mathbb{R}^{J \times R_2}$.

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

$$\underline{\mathbf{X}} \cong [\![\underline{\mathbf{G}}; \underline{\mathbf{B}}^{(1)}, \underline{\mathbf{B}}^{(2)}, \ldots, \underline{\mathbf{B}}^{(N)}]\!], \quad (3.20)$$

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 $\mathbf{i}_n = (i_{n,1}i_{n,2} \ldots i_{n,K_n})$ (Lee and Cichocki, 2016c).

Furthermore, the nested (hierarchical) form of such a generalized Tucker decomposition leads to the Tree Tensor Networks State (TTNS) model (Nakatani and Chan, 2013) (see Figure 2.15 and Figure 2.18), with possibly a varying order of cores, which can be formulated as

$$\begin{aligned} \underline{\mathbf{X}} &= [\![\underline{\mathbf{G}}_1; \; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N_1)}]\!] \\ \underline{\mathbf{G}}_1 &= [\![\underline{\mathbf{G}}_2; \; \underline{\mathbf{A}}^{(1,2)}, \underline{\mathbf{A}}^{(2,2)}, \ldots, \underline{\mathbf{A}}^{(N_2,2)}]\!]. \\ &\cdots \\ \underline{\mathbf{G}}_P &= [\![\underline{\mathbf{G}}_{P+1}; \; \underline{\mathbf{A}}^{(1,P+1)}, \underline{\mathbf{A}}^{(2,P+1)}, \ldots, \underline{\mathbf{A}}^{(N_{P+1},P+1)}]\!], \quad (3.21) \end{aligned}$$

---

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

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

Note that some factor tensors, $\underline{\mathbf{A}}^{(n,1)}$ and/or $\underline{\mathbf{A}}^{(n_p,p)}$, can be identity tensors which yield an irregular structure, possibly with a varying 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. It is usually assumed that the tree structure of TTNS is given or assumed *a priori*, and recent efforts aim to find an optimal tree structure from a subset of tensor entries and without any *a priori* knowledge of the tree structure. This is achieved using so-called rank-adaptive cross-approximation techniques which approximate a tensor by hierarchical tensor formats (Ballani and Grasedyck, 2014; Ballani *et al.*, 2014).

Operations in the Tucker format. If large-scale data tensors admit an exact or approximate representation in their Tucker formats, then most mathematical operations can be performed more efficiently using the so obtained much smaller core tensors and factor matrices. Consider the Nth-order tensors $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ in the Tucker format, given by

$$\underline{\mathbf{X}} = [\![\underline{\mathbf{G}}_X; \mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(N)}]\!] \quad \text{and} \quad \underline{\mathbf{Y}} = [\![\underline{\mathbf{G}}_Y; \mathbf{Y}^{(1)}, \ldots, \mathbf{Y}^{(N)}]\!], \quad (3.22)$$

for which the respective multilinear ranks are $\{R_1, R_2, \ldots, R_N\}$ and $\{Q_1, Q_2, \ldots, Q_N\}$, then the following mathematical operations can be performed directly in the Tucker format[5], which admits a significant reduction in computational costs (Phan *et al.*, 2013b, 2015b; Lee and Cichocki, 2016c):

- **The addition** of two Tucker tensors of the same order and sizes
$$\underline{\mathbf{X}} + \underline{\mathbf{Y}} = [\![\underline{\mathbf{G}}_X \oplus \underline{\mathbf{G}}_Y; \ [\mathbf{X}^{(1)}, \mathbf{Y}^{(1)}], \ldots, [\mathbf{X}^{(N)}, \mathbf{Y}^{(N)}]]\!], \quad (3.23)$$
  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 $\mathbf{Y}^{(n)} \in \mathbb{R}^{I_n \times Q_n}$, $\forall n$.

---

[5]Similar operations can be performed in the CP format, assuming that the core tensors are diagonal.

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

$$\underline{\mathbf{X}} \otimes \underline{\mathbf{Y}} = [\![\underline{\mathbf{G}}_X \otimes \underline{\mathbf{G}}_Y;\ \mathbf{X}^{(1)} \otimes \mathbf{Y}^{(1)}, \ldots, \mathbf{X}^{(N)} \otimes \mathbf{Y}^{(N)}]\!]. \quad (3.24)$$

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

$$\underline{\mathbf{X}} \circledast \underline{\mathbf{Y}} = [\![\underline{\mathbf{G}}_X \otimes \underline{\mathbf{G}}_Y;\ \mathbf{X}^{(1)} \odot_1 \mathbf{Y}^{(1)}, \ldots, \mathbf{X}^{(N)} \odot_1 \mathbf{Y}^{(N)}]\!], \quad (3.25)$$

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

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

$$\begin{aligned}
\langle \underline{\mathbf{X}}, \underline{\mathbf{Y}} \rangle &= \mathrm{vec}(\underline{\mathbf{X}})^{\mathrm{T}}\, \mathrm{vec}(\underline{\mathbf{Y}}) && (3.26) \\
&= \mathrm{vec}(\underline{\mathbf{G}}_X)^{\mathrm{T}} \left(\bigotimes_{n=1}^{N} \mathbf{X}^{(n)\,\mathrm{T}} \right) \left(\bigotimes_{n=1}^{N} \mathbf{Y}^{(n)} \right) \mathrm{vec}(\underline{\mathbf{G}}_Y) \\
&= \mathrm{vec}(\underline{\mathbf{G}}_X)^{\mathrm{T}} \left(\bigotimes_{n=1}^{N} \mathbf{X}^{(n)\,\mathrm{T}}\, \mathbf{Y}^{(n)} \right) \mathrm{vec}(\underline{\mathbf{G}}_Y) \\
&= \langle [\![\underline{\mathbf{G}}_X; (\mathbf{X}^{(1)\,\mathrm{T}}\, \mathbf{Y}^{(1)}), \ldots, (\mathbf{X}^{(N)\,\mathrm{T}}\, \mathbf{Y}^{(N)})]\!], \underline{\mathbf{G}}_Y \rangle.
\end{aligned}$$

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

$$\begin{aligned}
\|\underline{\mathbf{X}}\|_F &= \langle \underline{\mathbf{X}}, \underline{\mathbf{X}} \rangle \\
&= \mathrm{vec}\left([\![\underline{\mathbf{G}}_X; (\mathbf{X}^{(1)\,\mathrm{T}}\, \mathbf{X}^{(1)}), \ldots, (\mathbf{X}^{(N)\,\mathrm{T}}\, \mathbf{X}^{(N)})]\!] \right)^{\mathrm{T}} \mathrm{vec}(\underline{\mathbf{G}}_X) \\
&= \mathrm{vec}(\underline{\mathbf{G}}_X)^{\mathrm{T}}\, \mathrm{vec}(\underline{\mathbf{G}}_X) = \|\underline{\mathbf{G}}_X\|_F. && (3.27)
\end{aligned}$$

- **The N-D discrete convolution** of tensors $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $\underline{\mathbf{Y}} \in \mathbb{R}^{J_1 \times \cdots \times J_N}$ in their Tucker formats can be expressed as

$$\begin{aligned}
\underline{\mathbf{Z}} = \underline{\mathbf{X}} * \underline{\mathbf{Y}} &= [\![\underline{\mathbf{G}}_Z; \mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(N)}]\!] && (3.28) \\
&\in \mathbb{R}^{(I_1+J_1-1) \times \cdots \times (I_N+J_N-1)}.
\end{aligned}$$

If $\{R_1, R_2, \ldots, R_N\}$ is the multilinear rank of $\underline{\mathbf{X}}$ and $\{Q_1, Q_2, \ldots, Q_N\}$ the multilinear rank $\underline{\mathbf{Y}}$, then 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 the factor matrices

$$\mathbf{Z}^{(n)} = \mathbf{X}^{(n)} \boxdot_1 \mathbf{Y}^{(n)} \in \mathbb{R}^{(I_n + J_n - 1) \times R_n Q_n}, \qquad (3.29)$$

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 $s_n = \overline{r_n q_n} = 1, 2, \ldots, R_n Q_n$.

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

$$\mathcal{F}(\underline{\mathbf{X}}) = [\![\underline{\mathbf{G}}_X; \mathcal{F}(\mathbf{X}^{(1)}), \ldots, \mathcal{F}(\mathbf{X}^{(N)})]\!]. \qquad (3.30)$$

Note that if the data tensor admits low multilinear rank approximation, then performing the FFT on factor matrices of relatively small size $\mathbf{X}^{(n)} \in \mathbb{R}^{I_n \times R_n}$, instead of a large-scale data tensor, decreases considerably computational complexity. This approach is referred to as the super fast Fourier transform in Tucker format.

3.4   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 (De Lathauwer *et al.*, 2000a,b), 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 3.4).

   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 and its lateral slices

$$\langle \mathbf{S}_{:,k,:} \mathbf{S}_{:,l,:} \rangle = 0, \quad \text{for} \quad k \neq l, \qquad (3.31)$$

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 and its lateral slices

$$\|\mathbf{S}_{:,k,:}\|_F \geqslant \|\mathbf{S}_{:,l,:}\|_F, \quad k \geqslant l. \tag{3.32}$$

These norms play a role similar to singular values in standard matrix SVD.

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, $\mathbf{X}_{(n)} \cong \mathbf{U}^{(n)}\mathbf{S}_n\mathbf{V}^{(n)\mathrm{T}} \in \mathbb{R}^{I_n \times I_1 \cdots I_{n-1} I_{n+1} \cdots I_N}$. 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

$$\underline{\mathbf{S}} = \underline{\mathbf{X}} \times_1 \mathbf{U}^{(1)\,\mathrm{T}} \times_2 \mathbf{U}^{(2)\,\mathrm{T}} \cdots \times_N \mathbf{U}^{(N)\,\mathrm{T}}, \tag{3.33}$$

so that

$$\underline{\mathbf{X}} = \underline{\mathbf{S}} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \cdots \times_N \mathbf{U}^{(N)}. \tag{3.34}$$

Due to the orthogonality of the core tensor $\underline{\mathbf{S}}$, its slices are also 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 3.4 and Algorithm 2 outline the truncated HOSVD, for which any optimized matrix SVD procedure can be applied.

For large-scale tensors, the unfolding matrices, $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times I_{\bar{n}}}$ ($I_{\bar{n}} = I_1 \cdots I_n I_{n+1} \cdots I_N$) may become prohibitively large (with $I_{\bar{n}} \gg I_n$), easily exceeding the memory of standard computers. Using a direct and simple divide-and-conquer approach, the truncated SVD of an unfolding matrix, $\mathbf{X}_{(n)} = \mathbf{U}^{(n)}\mathbf{S}_n\mathbf{V}^{(n)\mathrm{T}}$, can be partitioned into Q slices, as $\mathbf{X}_{(n)} = [\mathbf{X}_{1,n}, \mathbf{X}_{2,n}, \ldots, \mathbf{X}_{Q,n}] = \mathbf{U}^{(n)}\mathbf{S}_n[\mathbf{V}_{1,n}^{\mathrm{T}}, \mathbf{V}_{2,n}^{\mathrm{T}}, \ldots, \mathbf{V}_{Q,n}^{\mathrm{T}}]$. Next, the orthogonal matrices $\mathbf{U}^{(n)}$ and the diagonal matrices $\mathbf{S}_n$ can be obtained from the eigenvalue decompositions $\mathbf{X}_{(n)}\mathbf{X}_{(n)}^{\mathrm{T}} = \mathbf{U}^{(n)}\mathbf{S}_n^2\mathbf{U}^{(n)\mathrm{T}} = \sum_q \mathbf{X}_{q,n}\mathbf{X}_{q,n}^{\mathrm{T}} \in \mathbb{R}^{I_n \times I_n}$, allowing for the terms $\mathbf{V}_{q,n} = \mathbf{X}_{q,n}^{\mathrm{T}}\mathbf{U}^{(n)}\mathbf{S}_n^{-1}$ to be computed separately. This enables us to optimize the size of the qth

(a)

(b)

(c)
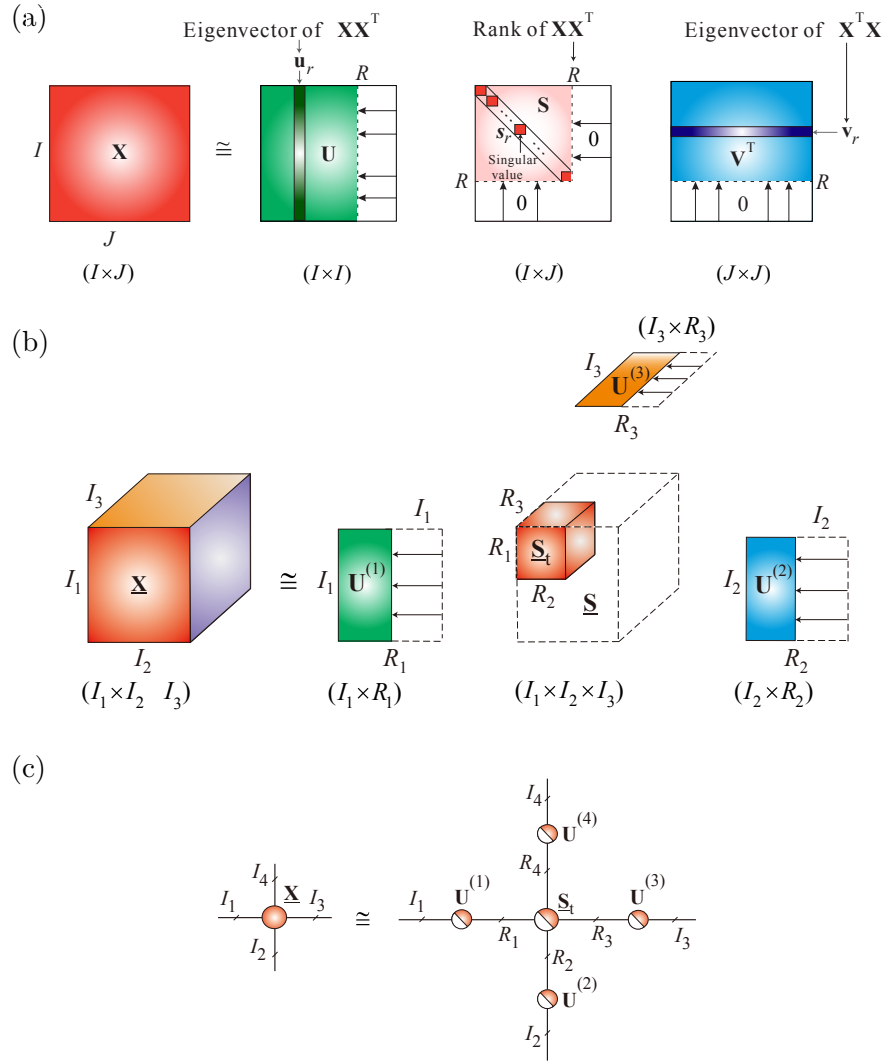
Figure 3.4: Graphical illustration of the truncated SVD and HOSVD. (a) The exact and truncated standard matrix SVD, $\mathbf{X} \cong \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}}$. (b) The truncated (approximative) HOSVD for a 3rd-order tensor calculated as $\underline{\mathbf{X}} \cong \underline{\mathbf{S}}_t \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)}$. (c) Tensor network notation for the HOSVD of a 4th-order tensor $\underline{\mathbf{X}} \cong \underline{\mathbf{S}}_t \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)} \times_4 \mathbf{U}^{(4)}$. All the factor matrices, $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times R_n}$, and the core tensor, $\underline{\mathbf{S}}_t = \underline{\mathbf{G}} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$, are orthogonal.

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 $I_{\bar{n}} = \prod_{k \neq n} I_k$.

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 $\mathbf{X}$ to a relatively small matrix by random sketching, i.e. through multiplication with a random sampling matrix $\mathbf{\Omega}$ (see Algorithm 3). Note that we explicitly allow the rank of the data matrix $\mathbf{X}$ to be overestimated (that is, $\tilde{R} = R + P$, where R is a true but unknown rank and P is the over-sampling parameter) because it is easier to obtain more accurate approximation of this form. Performance of randomized SVD can be further improved by integrating multiple random sketches, that is, by multiplying a data matrix $\mathbf{X}$ by a set of random matrices $\mathbf{\Omega}_p$ for $p = 1, 2, \ldots, P$ and integrating leading low-dimensional subspaces by applying a Monte Carlo integration method (Chen *et al.*, 2016).

Using special random sampling matrices, for instance, a subsampled 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 not accurate enough for matrices for which the singular values decay slowly (Halko *et al.*, 2011).

The truncated HOSVD can be optimized and implemented in several alternative ways. For example, if $R_n \ll I_n$, the truncated tensor $\underline{\mathbf{Z}} \leftarrow \underline{\mathbf{X}} \times_1 \mathbf{U}^{(1)\mathrm{T}}$ yields a smaller unfolding matrix $\mathbf{Z}_{(2)} \in \mathbb{R}^{I_2 \times R_1 I_3 \cdots I_N}$, so that the multiplication $\mathbf{Z}_{(2)} \mathbf{Z}_{(2)}^{\mathrm{T}}$ can be faster in the next iterations (Vannieuwenhoven *et al.*, 2012; Austin *et al.*, 2015).

Furthermore, since the unfolding matrices $\mathbf{Y}_{(n)}^{\mathrm{T}}$ 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 (Constantine *et al.*, 2014; Constantine and Gleich, 2011; Benson *et al.*, 2014).

Algorithm 2: **Sequentially Truncated HOSVD (Van-nieuwenhoven *et al.*, 2012)**

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 $\underline{\hat{\mathbf{X}}} = [\![\underline{\mathbf{S}}; \mathbf{U}^{(1)}, \ldots, \underline{\mathbf{U}}^{(N)}]\!]$, such that $\|\underline{\mathbf{X}} - \underline{\hat{\mathbf{X}}}\|_F \leqslant \varepsilon$
 1: $\underline{\mathbf{S}} \leftarrow \underline{\mathbf{X}}$
 2: **for** $n = 1$ to N **do**
 3:    $[\mathbf{U}^{(n)}, \mathbf{S}, \mathbf{V}] = \texttt{truncated\_svd}(\mathbf{S}_{(n)}, \frac{\varepsilon}{\sqrt{N}})$
 4:    $\mathbf{S} \leftarrow \mathbf{VS}$
 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 $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times R_n}$.

Algorithm 3: **Randomized SVD (rSVD) for large-scale and low-rank matrices with single sketch (Halko *et al.*, 2011)**

Input: A matrix $\mathbf{X} \in \mathbb{R}^{I \times J}$, desired or estimated rank R, and oversampling parameter P or overestimated rank $\tilde{R} = R + P$, exponent of the power method q ($q = 0$ or $q = 1$)
Output: An approximate rank-$\tilde{R}$ SVD, $\mathbf{X} \cong \mathbf{USV}^{\mathrm{T}}$, i.e., orthogonal matrices $\mathbf{U} \in \mathbb{R}^{I \times \tilde{R}}$, $\mathbf{V} \in \mathbb{R}^{J \times \tilde{R}}$ and diagonal matrix $\mathbf{S} \in \mathbb{R}^{\tilde{R} \times \tilde{R}}$ with singular values
 1: Draw a random Gaussian matrix $\mathbf{\Omega} \in \mathbb{R}^{J \times \tilde{R}}$,
 2: Form the sample matrix $\mathbf{Y} = (\mathbf{XX}^{\mathrm{T}})^q \mathbf{X\Omega} \in \mathbb{R}^{I \times \tilde{R}}$
 3: Compute a QR decomposition $\mathbf{Y} = \mathbf{QR}$
 4: Form the matrix $\mathbf{A} = \mathbf{Q}^{\mathrm{T}}\mathbf{X} \in \mathbb{R}^{\tilde{R} \times J}$
 5: Compute the SVD of the small matrix $\mathbf{A}$ as $\mathbf{A} = \hat{\mathbf{U}}\mathbf{SV}^{\mathrm{T}}$
 6: Form the matrix $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

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 (De Lathauwer *et al.*, 2000a)

$$\|\underline{\mathbf{X}} - [\![\underline{\mathbf{S}}; \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(N)}]\!]\| \leqslant \sqrt{N}\|\underline{\mathbf{X}} - \underline{\mathbf{X}}_{\mathrm{Best}}\|, \qquad (3.35)$$

Algorithm 4: **Higher Order Orthogonal Iteration (HOOI)**
(**De Lathauwer *et al.*, 2000b**; **Austin *et al.*, 2015**)

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

---

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

When it comes to the problem of 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. For the HOOI algorithms, see Algorithm 4 and Algorithm 5. For more sophisticated algorithms for Tucker decompositions with orthogonality and nonnegativity constraints, suitable for large-scale data tensors, see (Phan and Cichocki, 2011; Zhou *et al.*, 2012; Constantine *et al.*, 2014; Jeon *et al.*, 2016).

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 (3.33). Such computation is performed sequentially by fast matrix-by-matrix multiplications[6], as illustrated in Figure 3.5(a) and (b).

We have shown that 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,

---

[6]Efficient and parallel (state of the art) algorithms for multiplications of such very large-scale matrices are proposed in (Li *et al.*, 2015; Ballard *et al.*, 2015a).

Table 3.3: Basic multiway component analysis (MWCA)/Low-Rank Tensor Approximations (LRTA) and related multiway dimensionality reduction models. The symbol $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ denotes a noisy data tensor, while $\mathbf{Y} = \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)}$ 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 $\underline{\mathbf{Y}} = \sum_{r=1}^{R} \lambda_r (\mathbf{b}_r^{(1)} \circ \mathbf{b}_r^{(2)} \circ \cdots \circ \mathbf{b}_r^{(N)})$.

Cost Function	Constraints
Multilinear (sparse) PCA (MPCA)	$\mathbf{u}_r^{(n)\,\mathrm{T}} \mathbf{u}_r^{(n)} = 1, \ \forall(n, r)$
$\max_{\mathbf{u}_r^{(n)}} \ \underline{\mathbf{X}} \bar{\times}_1 \mathbf{u}_r^{(1)} \bar{\times}_2 \mathbf{u}_r^{(2)} \cdots \bar{\times}_N \mathbf{u}_r^{(N)} + \gamma \sum_{n=1}^{N} \|\mathbf{u}_r^{(n)}\|_1$	$\mathbf{u}_r^{(n)\,\mathrm{T}} \mathbf{u}_q^{(n)} = 0 \ \text{ for } \ r \neq q$
HOSVD/HOOI	
$\min_{\mathbf{U}^{(n)}} \ \|\underline{\mathbf{X}} - \underline{\mathbf{G}} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \cdots \times_N \mathbf{U}^{(N)}\|_F^2$	$\mathbf{U}^{(n)\,\mathrm{T}} \mathbf{U}^{(n)} = \mathbf{I}_{R_n}, \ \forall n$
Multilinear ICA	Vectors of $\mathbf{B}^{(n)}$ statistically
$\min_{\mathbf{B}^{(n)}} \ \|\underline{\mathbf{X}} - \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)}\|_F^2$	as independent as possible
Nonnegative CP/Tucker decomposition	
(NTF/NTD) (Cichocki *et al.*, 2009)	Entries of $\underline{\mathbf{G}}$ and $\mathbf{B}^{(n)}, \ \forall n$
$\min_{\mathbf{B}^{(n)}} \ \|\underline{\mathbf{X}} - \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \cdots \times_N \mathbf{B}^{(N)}\|_F^2$	are nonnegative
$+\gamma \sum_{n=1}^{N} \sum_{r_n=1}^{R_n} \|\mathbf{b}_{r_n}^{(n)}\|_1$	
Sparse CP/Tucker decomposition	Sparsity constraints
$\min_{\mathbf{B}^{(n)}} \ \|\underline{\mathbf{X}} - \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \cdots \times_N \mathbf{B}^{(N)}\|_F^2$	imposed on $\mathbf{B}^{(n)}$
$+\gamma \sum_{n=1}^{N} \sum_{r_n=1}^{R_n} \|\mathbf{b}_{r_n}^{(n)}\|_1$	
Smooth CP/Tucker decomposition	Smoothness imposed
(SmCP/SmTD) (Yokota *et al.*, 2016)	on vectors $\mathbf{b}_r^{(n)}$
$\min_{\mathbf{B}^{(n)}} \ \|\underline{\mathbf{X}} - \underline{\mathbf{\Lambda}} \times_1 \mathbf{B}^{(1)} \cdots \times_N \mathbf{B}^{(N)}\|_F^2$	of $\mathbf{B}^{(n)} \in \mathbb{R}^{I_n \times R}, \ \ \forall n$
$+\gamma \sum_{n=1}^{N} \sum_{r=1}^{R} \|\mathbf{L}\mathbf{b}_r^{(n)}\|_2$	via a difference operator $\mathbf{L}$

we can partition the orthogonal factor matrices $\mathbf{U}^{(n)\mathrm{T}}$ into the corresponding blocks of matrices $\mathbf{U}_{[k_n, p_n]}^{(n)\mathrm{T}}$, as illustrated in Figure 3.5(c) for 3rd-order tensors (Wang *et al.*, 2005; Suter *et al.*, 2013). For example, the blocks within the resulting tensor $\underline{\mathbf{G}}^{(n)}$ can be computed

---

Algorithm 5: **HOOI using randomization for large-scale data (Zhou *et al.*, 2015)**

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 $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times R_n}$

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

Algorithm 6: **Tucker decomposition with constrained factor matrices via 2-way CA /LRMF**

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

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

---

sequentially or in parallel, as follows:

$$\underline{\mathbf{G}}^{(n)}_{[k_1, k_2, \ldots, q_n, \ldots, k_N]} = \sum_{k_n=1}^{K_n} \mathbf{X}_{[k_1, k_2, \ldots, k_n, \ldots, k_N]} \times_n \mathbf{U}^{(n)\,\mathrm{T}}_{[k_n, q_n]}. \quad (3.36)$$

(a) Sequential computation
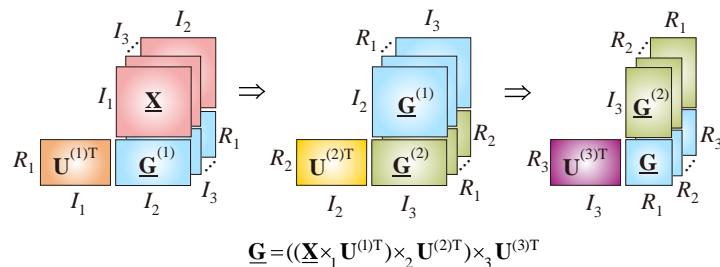
$$\underline{\mathbf{G}} = ((\underline{\mathbf{X}} \times_1 \mathbf{U}^{(1)\mathrm{T}}) \times_2 \mathbf{U}^{(2)\mathrm{T}}) \times_3 \mathbf{U}^{(3)\mathrm{T}}$$

(b) Fast matrix-by-matrix approach

(c) Divide-and-conquer approach

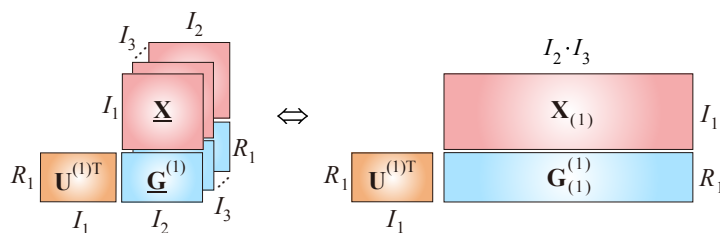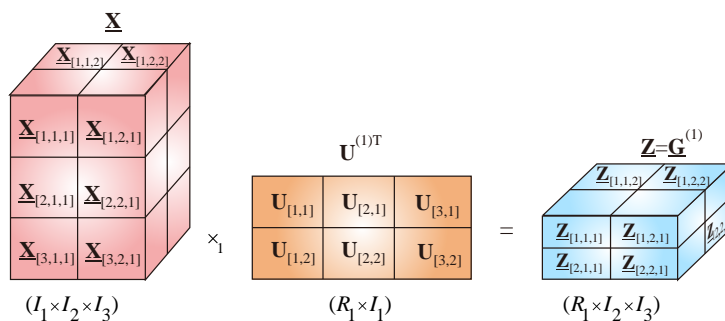**Figure 3.5:** Computation of a multilinear (Tucker) product for large-scale HOSVD. (a) Standard sequential computing of multilinear products (TTM) $\underline{\mathbf{G}} = \underline{\mathbf{S}} = (((\underline{\mathbf{X}} \times_1 \mathbf{U}^{(1)\mathrm{T}}) \times_2 \mathbf{U}^{(2)\mathrm{T}}) \times_3 \mathbf{U}^{(3)\mathrm{T}})$. (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, $\mathbf{U}^{(n)\mathrm{T}}$, are partitioned into suitable small blocks: Subtensors $\underline{\mathbf{X}}_{[k_1,k_2,k_3]}$ and block matrices $\mathbf{U}^{(1)\mathrm{T}}_{[k_1,p_1]}$. The blocks of a tensor, $\underline{\mathbf{Z}} = \underline{\mathbf{G}}^{(1)} = \underline{\mathbf{X}} \times_1 \mathbf{U}^{(1)\mathrm{T}}$, are computed as $\underline{\mathbf{Z}}_{[q_1,k_2,k_3]} = \sum_{k_1=1}^{K_1} \underline{\mathbf{X}}_{[k_1,k_2,k_3]} \times_1 \mathbf{U}^{(1)\mathrm{T}}_{[k_1,q_1]}$ (see Eq. (3.36) for a general case).

Applications. We have shown that the Tucker/HOSVD decomposition may be considered as a multilinear extension of PCA (Kroonenberg, 2008); it therefore generalizes signal subspace techniques and finds application in areas including multilinear blind source separation, classification, feature extraction, and subspace-based harmonic retrieval (Vasilescu and Terzopoulos, 2002; Haardt *et al.*, 2008; Phan and Cichocki, 2010; Lu *et al.*, 2011). 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 also makes Tucker decomposition a natural tool for signal compression and enhancement.

It was recently shown that HOSVD can also perform simultaneous subspace selection (data compression) and K-means clustering, both unsupervised learning tasks (Huang *et al.*, 2008; Papalexakis *et al.*, 2013). This is important, as a combination of these methods can both identify and classify "relevant" data, and in this way not only reveal desired information but also simplify 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 (Chandola *et al.*, 2009; Fanaee-T and Gama, 2016). While such analysis can be performed in different domains, it is most frequently based on 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. 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 considering very large datasets, since the basic Tucker decomposition model generalizes PCA and SVD, it offers a natural framework for anomaly detection via HOSVD, as illustrated in Figure 3.6. 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
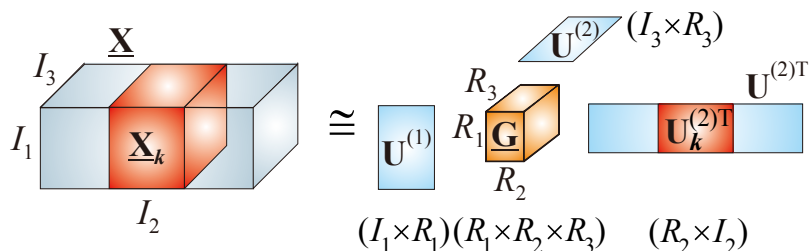
Figure 3.6: Conceptual model for performing the HOSVD for a very large-scale 3rd-order data tensor. This is achieved by dividing the tensor into blocks $\underline{\mathbf{X}}_k \cong \underline{\mathbf{G}} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}_k^{(2)} \times_3 \mathbf{U}^{(3)}$, $(k = 1, 2 \ldots, K)$. 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 (with an overlapping sliding window). 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 $\mathbf{U}^{(n)}$.

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 3.6.

3.5 Tensor Sketching Using Tucker Model

The notion of sketches refers to replacing the original huge matrix or tensor by a new matrix or tensor of a significantly smaller size or compactness, but which approximates well the original matrix/tensor. Finding such sketches in an efficient way is important for the analysis of big data, as a computer processor (and memory) is often incapable of handling the whole data-set in a feasible amount of time. For these reasons, the computation is often spread among a set of processors which for standard "all-in-one" SVD algorithms, are unfeasible.

Given a very large-scale tensor $\underline{\mathbf{X}}$, a useful approach is to compute a sketch tensor $\underline{\mathbf{Z}}$ or set of sketch tensors $\underline{\mathbf{Z}}_n$ that are of significantly smaller sizes than the original one.

There exist several matrix and tensor sketching approaches: sparsi-fication, random projections, fiber subset selections, iterative sketching

techniques and distributed sketching. We review the main sketching approaches which are promising for tensors.

1. Sparsification generates a sparser version of the tensor which, in general, can be stored more efficiently and admit faster multiplications by factor matrices. This is achieved by decreasing the number on non-zero entries and quantizing or rounding up entries. A simple technique is element-wise sparsification which zeroes out all sufficiently small elements (below some threshold) of a data tensor, keeps all sufficiently large elements, and randomly samples the remaining elements of the tensor with sample probabilities proportional to the square of their magnitudes (Nguyen *et al.*, 2015).

2. Random Projection based sketching randomly combines fibers of a data tensor in all or selected modes, and is related to the concept of a randomized subspace embedding, which is used to solve a variety of numerical linear algebra problems (see (Tropp *et al.*, 2016) and references therein).

3. Fiber subset selection, also called tensor cross approximation (TCA), finds a small subset of fibers which approximates the entire data tensor. For the matrix case, this problem is known as the Column/Row Subset Selection or CUR Problem which has been thoroughly investigated and for which there exist several algorithms with almost matching lower bounds (Mahoney, 2011; Desai *et al.*, 2016; Ghashami *et al.*, 2016).

3.5.1 Tensor Sketching via Multiple Random Projections

The random projection framework has been developed for computing structured low-rank approximations of a data tensor from (random) linear projections of much lower dimensions than the data tensor itself (Caiafa and Cichocki, 2015; Tropp *et al.*, 2016). Such techniques have many potential applications in large-scale numerical multilinear algebra and optimization problems.

Notice that for an Nth-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, we can compute the following sketches

$$\underline{\mathbf{Z}} = \underline{\mathbf{X}} \times_1 \mathbf{\Omega}_1 \times_2 \mathbf{\Omega}_2 \cdots \times_N \mathbf{\Omega}_N \tag{3.37}$$

(a)
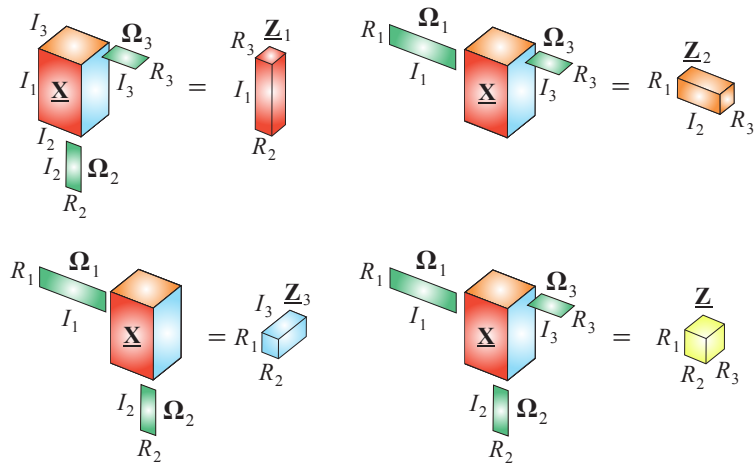
(b)
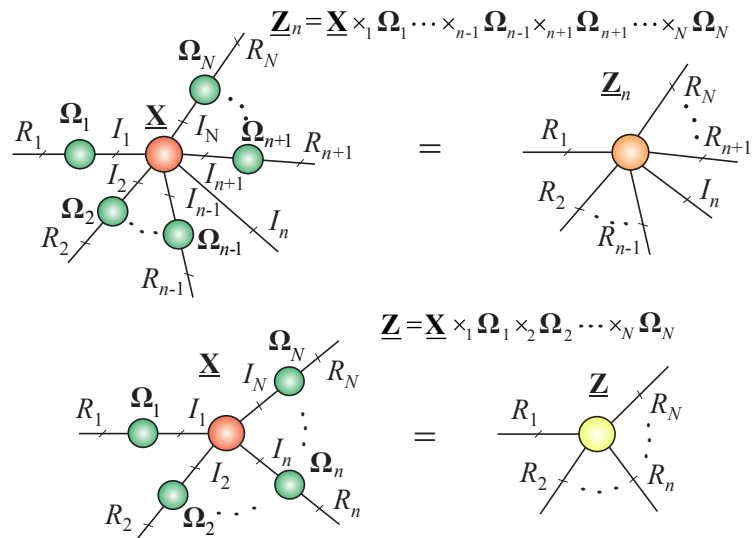
Figure 3.7: Illustration of tensor sketching using random projections of a data tensor. (a) Sketches of a 3rd-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ given by $\underline{\mathbf{Z}}_1 = \underline{\mathbf{X}} \times_2 \mathbf{\Omega}_2 \times_3 \mathbf{\Omega}_3 \in \mathbb{R}^{I_1 \times R_2 \times R_3}$, $\underline{\mathbf{Z}}_2 = \underline{\mathbf{X}} \times_1 \mathbf{\Omega}_1 \times_3 \mathbf{\Omega}_3 \in \mathbb{R}^{R_1 \times I_2 \times R_3}$, $\underline{\mathbf{Z}}_3 = \underline{\mathbf{X}} \times_1 \mathbf{\Omega}_1 \times_2 \mathbf{\Omega}_2 \in \mathbb{R}^{R_1 \times R_2 \times I_3}$, and $\underline{\mathbf{Z}} = \underline{\mathbf{X}} \times_1 \mathbf{\Omega}_1 \times_2 \mathbf{\Omega}_2 \times_3 \mathbf{\Omega}_3 \in \mathbb{R}^{R_1 \times R_2 \times R_3}$. (b) Sketches for an Nth-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$.

and

$$\underline{\mathbf{Z}}_n = \underline{\mathbf{X}} \times_1 \mathbf{\Omega}_1 \cdots \times_{n-1} \mathbf{\Omega}_{n-1} \times_{n+1} \mathbf{\Omega}_{n+1} \cdots \times_N \mathbf{\Omega}_N, \qquad (3.38)$$

for $n =, 1, 2, \ldots, N$, where $\mathbf{\Omega}_n \in \mathbb{R}^{R_n \times I_n}$ are statistically independent random matrices with $R_n \ll I_n$, usually called test (or sensing) matrices.

A sketch can be implemented using test matrices drawn from various distributions. The choice of a distribution leads to some tradeoffs (Tropp *et al.*, 2016), especially regarding (i) the costs of randomization, computation, and communication to generate the test matrices; (ii) the storage costs for the test matrices and the sketch; (iii) the arithmetic costs for sketching and updates; (iv) the numerical stability of reconstruction algorithms; and (v) the quality of a priori error bounds. The most important distributions of random test matrices include:

- **Gaussian random projections** which generate random matrices with standard normal distribution. Such matrices usually provide excellent performance in practical scenarios and accurate a priori error bounds.

- **Random matrices with orthonormal columns** that span uniformly distributed random subspaces of dimensions R_n. Such matrices behave similar to Gaussian case, but usually exhibit even better numerical stability, especially when R_n are large.

- **Rademacher and super-sparse Rademacher random projections** that have independent Rademacher entries which take the values ± 1 with equal probability. Their properties are similar to standard normal test matrices, but exhibit some improvements in the cost of storage and computational complexity. In a special case, we may use ultra sparse Rademacher test matrices, whereby in each column of a test matrix independent Rademacher random variables are placed only in very few uniformly random locations determined by a sampling parameter s; the remaining entries are set to zero. In an extreme case of maximum sparsity, $s = 1$, and each column of a test matrix has exactly only one nonzero entry.

- **Subsampled randomized Fourier transforms** based on test matrices take the following form

$$\boldsymbol{\Omega}_n = \mathbf{P}_n \mathbf{F}_n \mathbf{D}_n, \tag{3.39}$$

  where $\mathbf{D}_n$ are diagonal square matrices with independent Rademacher entries, $\mathbf{F}_n$ are discrete cosine transform (DCT) or discrete Fourier transform (DFT) matrices, and entries of the matrix $\mathbf{P}_n$ are drawn at random from a uniform distribution.

Example. The concept of tensor sketching via random projections is illustrated in Figure 3.7 for a 3rd-order tensor and for a general case of Nth-order tensors. For a 3rd-order tensor with volume (number of entries) $I_1 I_2 I_3$ we have four possible sketches which are subtensors of much smaller sizes, e.g., $I_1 R_2 R_3$, with $R_n \ll I_n$, if the sketching is performed along mode-2 and mode-3, or $R_1 R_2 R_3$, if the sketching is performed along all three modes (Figure 3.7(a) bottom right). From these subtensors we can reconstruct any huge tensor if it has low a multilinear rank (lower than $\{R_1, R_2, \ldots, R_n\}$).

In more general scenario, it can be shown (Caiafa and Cichocki, 2015) that the Nth order tensor data tensor $\underline{\mathbf{X}}$ with sufficiently low-multilinear rank can be reconstructed perfectly from the sketch tensors $\underline{\mathbf{Z}}_n$, for $n = 1, 2, \ldots, N$, as follows

$$\hat{\underline{\mathbf{X}}} = \underline{\mathbf{Z}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)}, \tag{3.40}$$

where $\mathbf{B}^{(n)} = [\underline{\mathbf{Z}}_n]_{(n)} \mathbf{Z}_{(n)}^{\dagger}$ for $n = 1, 2, \ldots, N$ (for more detail see the next section).

3.5.2   Matrix/Tensor Cross-Approximation (MCA/TCA)

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 (Goreinov *et al.*, 1997b,a; Mahoney *et al.*, 2008; Mahoney and Drineas, 2009; Oseledets and Tyrtyshnikov, 2010; Bebendorf, 2011; Bebendorf *et al.*, 2015; Khoromskij and Veit, 2016). The main idea behind the MCA is to provide reduced dimensionality of data through a linear combination
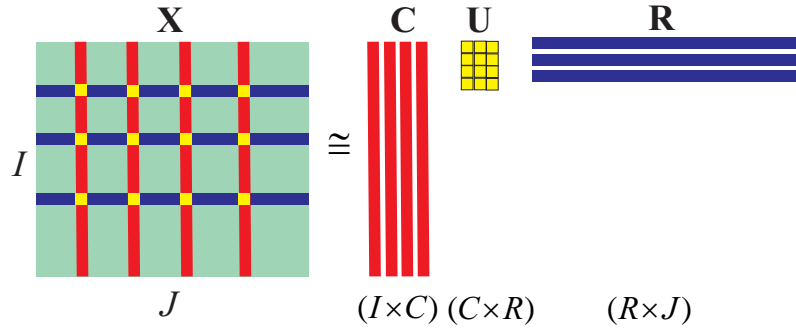
Figure 3.8: Principle of the matrix cross-approximation which decomposes a huge matrix **X** into a product of three matrices, whereby only a small-size core matrix **U** needs to be computed.

of only a few "meaningful" components, which are exact replicas of columns and rows of the original data matrix. 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.

The MCA method factorizes a data matrix $\mathbf{X} \in \mathbb{R}^{I \times J}$ as (Goreinov *et al.*, 1997a,b) (see Figure 3.8)

$$\mathbf{X} = \mathbf{CUR} + \mathbf{E}, \tag{3.41}$$

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 appropriately selected rows of $\mathbf{X}$, and matrix $\mathbf{U} \in \mathbb{R}^{C \times R}$ is calculated so as to minimize the norm of the error $\mathbf{E} \in \mathbb{R}^{I \times J}$.

A simple modification of this formula, whereby the matrix **U** is absorbed into either **C** or **R**, yields the so-called CR matrix factorization or Column/Row Subset selection:

$$\mathbf{X} \cong \mathbf{C\tilde{R}} = \mathbf{\tilde{C}R} \tag{3.42}$$

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

For dimensionality reduction, $C \ll J$ and $R \ll I$, and the columns and rows of **X** should be chosen optimally, in the sense of providing a
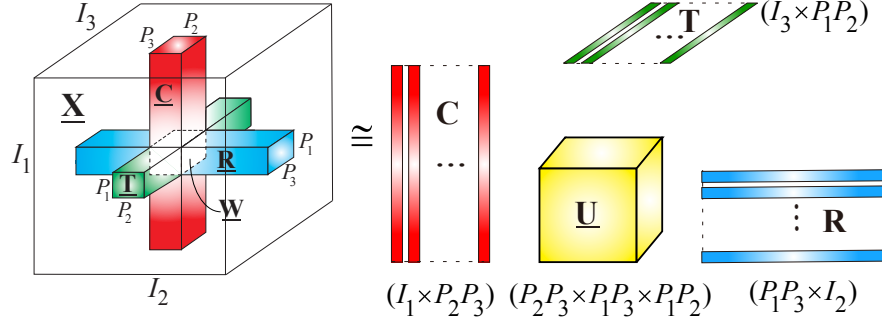
Figure 3.9: The principle of the tensor cross-approximation (TCA) algorithm, illustrated for a large-scale 3rd-order tensor $\underline{\mathbf{X}} \cong \underline{\mathbf{U}} \times_1 \mathbf{C} \times_2 \mathbf{R} \times_3 \mathbf{T} = [\![\underline{\mathbf{U}}; \mathbf{C}, \mathbf{R}, \mathbf{T}]\!]$, where $\underline{\mathbf{U}} = \underline{\mathbf{W}} \times_1 \mathbf{W}^{\dagger}_{(1)} \times_2 \mathbf{W}^{\dagger}_{(2)} \times_3 \mathbf{W}^{\dagger}_{(3)} = [\![\underline{\mathbf{W}}; \mathbf{W}^{\dagger}_{(1)}, \mathbf{W}^{\dagger}_{(2)}, \mathbf{W}^{\dagger}_{(3)}]\!] \in \mathbb{R}^{P_2 P_3 \times P_1 P_3 \times P_1 P_2}$ and $\underline{\mathbf{W}} \in \mathbb{R}^{P_1 \times P_2 \times P_3}$. For simplicity of illustration, we assume that the selected fibers are permuted, 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 $\underline{\mathbf{T}} \in \mathbb{R}^{P_1 \times P_2 \times I_3}$.

high "statistical leverage" and the best low-rank fit to the data matrix, while at the same time minimizing the cost function $\|\mathbf{E}\|^2_F$. For a given set of columns, $\mathbf{C}$, and rows, $\mathbf{R}$, the optimal choice for the core matrix is $\mathbf{U} = \mathbf{C}^{\dagger}\mathbf{X}(\mathbf{R}^{\dagger})^{\mathrm{T}}$. 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 for $\mathrm{rank}(\mathbf{X}) \leqslant \min\{C, R\}$ the cross-approximation is exact. For the general case, it has been proven that when the intersection submatrix $\mathbf{W}$ is of maximum volume[7], the matrix cross-approximation is close to the optimal SVD solution. The problem of finding a submatrix with maximum volume has exponential complexity, however, suboptimal matrices can be found using fast greedy algorithms (Wang and Zhang, 2013; Mikhalev and Oseledets, 2015; Rakhuba and Oseledets, 2015; Anderson *et al.*, 2015).

The concept of MCA can be generalized to tensor cross-approximation (TCA) (see Figure 3.9) through several approaches, including:

---

[7]The volume of a square submatrix $\mathbf{W}$ is defined as $|\det(\mathbf{W})|$.

(a)
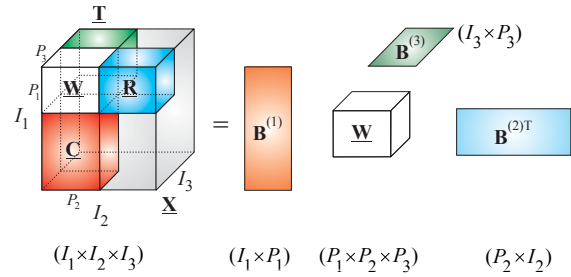
(b)
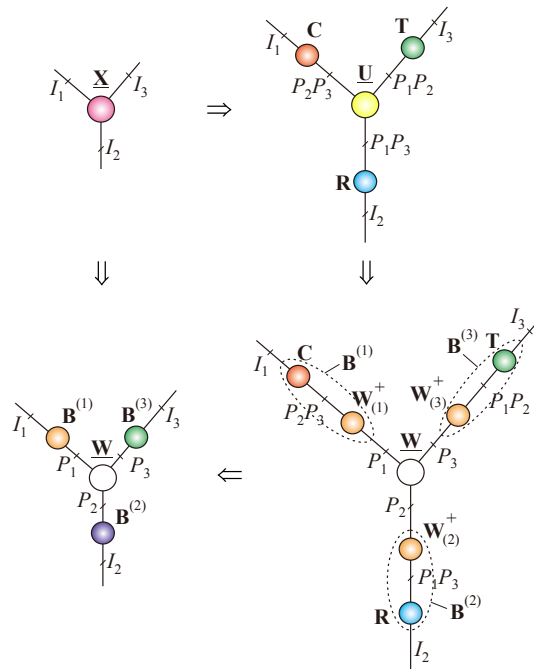
Figure 3.10: The Tucker decomposition of a low multilinear rank 3rd-order tensor using the cross-approximation approach. (a) Standard block diagram. (b) Transformation from the TCA in the Tucker format, $\underline{\mathbf{X}} \cong \underline{\mathbf{U}} \times_1 \mathbf{C} \times_2 \mathbf{R} \times_3 \mathbf{T}$, into a standard Tucker representation, $\underline{\mathbf{X}} \cong \underline{\mathbf{W}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \times_3 \mathbf{B}^{(3)} = [\![\underline{\mathbf{W}}; \mathbf{C}\mathbf{W}_{(1)}^\dagger, \mathbf{R}\mathbf{W}_{(2)}^\dagger, \mathbf{T}\mathbf{W}_{(3)}^\dagger]\!]$, with a prescribed core tensor $\underline{\mathbf{W}}$.

- Applying the MCA decomposition to a matricized version of the tensor data (Mahoney *et al.*, 2008);

- Operating directly on fibers of a data tensor which admits a low-rank Tucker approximation, an approach termed the Fiber Sampling Tucker Decomposition (FSTD) (Caiafa and Cichocki, 2010, 2013, 2015).

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 subtensor, $\underline{\mathbf{W}} \in \mathbb{R}^{P_1 \times P_2 \times P_3}$, so that the following exact Tucker representation can be obtained (see Figure 3.10)

$$\underline{\mathbf{X}} = [\![\underline{\mathbf{U}}; \mathbf{C}, \mathbf{R}, \mathbf{T}]\!], \tag{3.43}$$

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

$$\underline{\mathbf{X}} = [\![\underline{\mathbf{W}}; \mathbf{C}\mathbf{W}_{(1)}^{\dagger}, \mathbf{R}\mathbf{W}_{(2)}^{\dagger}, \mathbf{T}\mathbf{W}_{(3)}^{\dagger}]\!]. \tag{3.44}$$

Observe that for $N = 2$, the TCA model simplifies into the MCA for a matrix case, $\mathbf{X} = \mathbf{C}\mathbf{U}\mathbf{R}$, 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 (Caiafa and Cichocki, 2010) that 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 $\{R_1, R_2, \ldots, R_N\}$.

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 (Caiafa and Cichocki, 2010). These entries are selected sequentially using a deflation approach, thus making the tensor cross-approximation FSTD algorithm suitable for the approximation of very large-scale but relatively low-order tensors (including tensors with missing fibers or entries).

It should be noted that an alternative efficient way to estimate subtensors $\underline{\mathbf{W}}, \underline{\mathbf{C}}, \underline{\mathbf{R}}$ and $\underline{\mathbf{T}}$ is to apply random projections as follows

$$\underline{\mathbf{W}} = \underline{\mathbf{Z}} = \underline{\mathbf{X}} \times_1 \mathbf{\Omega}_1 \times_2 \mathbf{\Omega}_2 \times_3 \mathbf{\Omega}_3 \in \mathbb{R}^{P_1 \times P_2 \times P_3},$$
$$\underline{\mathbf{C}} = \underline{\mathbf{Z}}_1 = \underline{\mathbf{X}} \times_2 \mathbf{\Omega}_2 \times_3 \mathbf{\Omega}_3 \in \mathbb{R}^{I_1 \times P_2 \times P_3},$$
$$\underline{\mathbf{R}} = \underline{\mathbf{Z}}_2 = \underline{\mathbf{X}} \times_1 \mathbf{\Omega}_1 \times_3 \mathbf{\Omega}_3 \in \mathbb{R}^{P_1 \times I_2 \times P_3},$$
$$\underline{\mathbf{T}} = \underline{\mathbf{Z}}_3 = \underline{\mathbf{X}} \times_1 \mathbf{\Omega}_1 \times_2 \mathbf{\Omega}_2 \in \mathbb{R}^{P_1 \times P_2 \times I_3}, \tag{3.45}$$

where $\mathbf{\Omega}_n \in \mathbb{R}^{P_n \times I_n}$ with $P_n \geqslant R_n$ for $n = 1, 2, 3$ are independent random matrices. We explicitly assume that the multilinear rank $\{P_1, P_2, \ldots, P_N\}$ of approximated tensor to be somewhat larger than a true multilinear rank $\{R_1, R_2, \ldots, R_N\}$ of target tensor, because it is easier to obtain an accurate approximation in this form.

3.6 Multiway Component Analysis (MWCA)

3.6.1 Multilinear Component Analysis Using Constrained Tucker Decomposition

The great success of 2-way component analyses (PCA, ICA, NMF, SCA) is largely due to the existence of very efficient algorithms for their computation 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, to exploit the full potential of tensor factorization/decompositions, it is a prerequisite to impose suitable constraints

on the desired components. In fact, there is much more flexibility for tensors, since different constraints can be imposed on the matrix factorizations in every mode n a matricized tensor $\mathbf{X}_{(n)}$ (see Algorithm 6 and Figure 3.11).

Such physically meaningful representation through flexible modewise constraints underpins the concept of multiway component analysis (MWCA). The Tucker representation of MWCA naturally accommodates such diversities in different modes. Besides the orthogonality, alternative constraints in the Tucker format include statistical independence, sparsity, smoothness and nonnegativity (Vasilescu and Terzopoulos, 2002; Cichocki *et al.*, 2009; Zhou and Cichocki, 2012a; Cichocki *et al.*, 2015b) (see Table 3.3).

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 algorithms.

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

3. Optionally, perform fine tuning of factor matrices and the core tensor by the ALS minimization 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.

3.6.2 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 multiway component analysis (MWCA) methods.
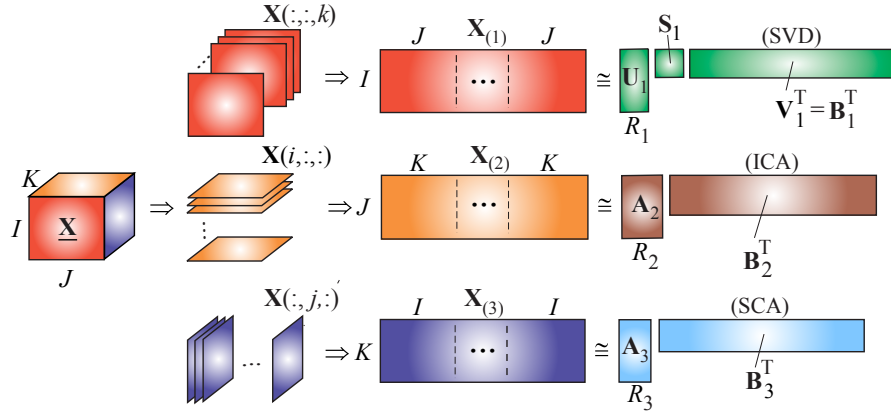
Figure 3.11: Multiway Component Analysis (MWCA) for a third-order tensor via constrained matrix factorizations, assuming that the components are: orthogonal in the first mode, statistically independent in the second mode and sparse in the third mode.

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

$$\mathbf{X}^{(k)} \cong \mathbf{A}\mathbf{G}^{(k)}\mathbf{B}^{\mathrm{T}}, \quad (k = 1, 2, \ldots, K), \qquad (3.46)$$

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) (Crainiceanu *et al.*, 2011). Note that the PVD is equivalent to the unconstrained or constrained Tucker-2 model, as illustrated in Figure 3.12. In a special case with square diagonal matrices, $\mathbf{G}^{(k)}$, the model is equivalent to the CP decomposition and is related to joint matrix diagonalization (De Lathauwer, 2006; Tichavský and Yeredor, 2009; Chabriel *et al.*, 2014). Furthermore, if $\mathbf{A} = \mathbf{B}$ then the PVD model is equivalent to the RESCAL model (Nickel *et al.*, 2016).

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

---

Algorithm 7: **Population Value Decomposition (PVD) with orthogonality constraints**

Input: A set of matrices $\mathbf{X}_k \in \mathbb{R}^{I \times J}$, for $k = 1, \ldots, K$ (typically, $K \gg \max\{I, J\}$)

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 $\mathbf{A}^{\mathrm{T}}\mathbf{A} = \mathbf{I}_{R_1}$ and $\mathbf{B}^{\mathrm{T}}\mathbf{B} = \mathbf{I}_{R_2}$

1: **for** $k = 1$ to K **do**
2:    Perform truncated SVD, $\mathbf{X}_k = \mathbf{U}_k \mathbf{S}_k \mathbf{V}_k^{\mathrm{T}}$, using R largest singular values
3: **end for**
4: Construct short and wide matrices:
   $\mathbf{U} = \begin{bmatrix} \mathbf{U}_1 \mathbf{S}_1, \ldots, \mathbf{U}_K \mathbf{S}_K \end{bmatrix} \in \mathbb{R}^{I \times KR}$ and
   $\mathbf{V} = \begin{bmatrix} \mathbf{V}_1 \mathbf{S}_1, \ldots, \mathbf{V}_K \mathbf{S}_K \end{bmatrix} \in \mathbb{R}^{J \times KR}$
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 $\mathbf{G}_k = \mathbf{A}^{\mathrm{T}}\mathbf{X}_k\mathbf{B}$
8: **end for**

---

reshaped and optionally permuted into a "skinny and tall" 3rd-order tensor, $\underline{\widetilde{\mathbf{X}}} \in \mathbb{R}^{J \times J \times K}$, with e.g., $I = I_1$, $J = I_2$ and $K = I_3 I_4 \cdots I_N$, for which PVD/Tucker-2 Algorithm 8 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 7 (Crainiceanu *et al.*, 2011; Constantine *et al.*, 2014; Wang *et al.*, 2016). 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, $\sum_{k=1}^{K} \|\mathbf{X}_k - \mathbf{A}\mathbf{G}_k\mathbf{B}^{\mathrm{T}}\|_F^2$, and for data corrupted by Gaussian noise, better performance can be achieved using the HOOI-2 given in Algorithm 4, for $N = 3$. An improved PVD algorithm referred to as Tucker-2 algorithm is given in Algorithm 8 (Phan *et al.*, 2016).

Linked MWCA. Consider the analysis of multi-modal high-dimensional data collected under the same or very similar conditions,

(a)
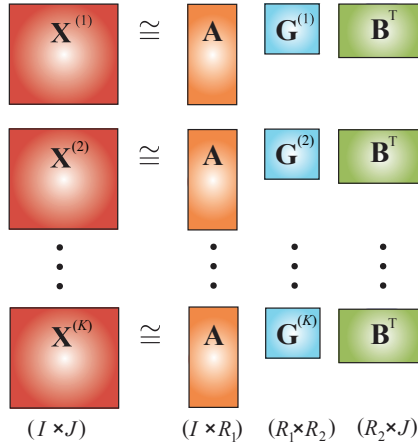
(b)
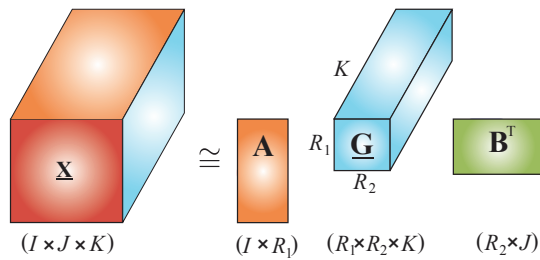
Figure 3.12: Concept of the Population Value Decomposition (PVD). (a) Principle of simultaneous multi-block matrix factorizations. (b) Equivalent representation of the PVD as the constrained or unconstrained Tucker-2 decomposition, $\underline{\mathbf{X}} \cong \underline{\mathbf{G}} \times_1 \mathbf{A} \times_2 \mathbf{B}$. The objective is to find the common factor matrices, $\mathbf{A}$, $\mathbf{B}$ and the core tensor, $\underline{\mathbf{G}} \in \mathbb{R}^{R_1 \times R_2 \times K}$.

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 own independent features. As a result, it is advantageous and natural to analyze such data

---

Algorithm 8: **Orthogonal Tucker-2 decomposition with a prescribed approximation accuracy (Phan *et al.*, 2016)**

Input: A 3rd-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$ (typically, $K \gg \max\{I, J\}$) 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 the constraint $\|\underline{\mathbf{X}} - \underline{\mathbf{G}} \times_1 \mathbf{A} \times \mathbf{B}\|_F^2 \leqslant \varepsilon^2$, s.t, $\mathbf{A}^{\mathrm{T}}\mathbf{A} = \mathbf{I}_{R_1}$ and $\mathbf{B}^{\mathrm{T}}\mathbf{B} = \mathbf{I}_{R_2}$.

1: Initialize $\mathbf{A} = \mathbf{I}_I \in \mathbb{R}^{I \times I}$, $R_1 = I$

2: **while** not converged or iteration limit is not reached **do**

3:    Compute the tensor $\underline{\mathbf{Z}}^{(1)} = \underline{\mathbf{X}} \times_1 \mathbf{A}^{\mathrm{T}} \in \mathbb{R}^{R_1 \times J \times K}$

4:    Compute EVD of a small matrix $\mathbf{Q}_1 = \mathbf{Z}^{(1)}_{(2)} \mathbf{Z}^{(1)\,\mathrm{T}}_{(2)} \in \mathbb{R}^{J \times J}$ as $\mathbf{Q}_1 = \mathbf{B} \ \mathrm{diag}\,(\lambda_1, \cdots, \lambda_{R_2}) \ \mathbf{B}^{\mathrm{T}}$, such that $\sum_{r_2=1}^{R_2} \lambda_{r_2} \geqslant \|\underline{\mathbf{X}}\|_F^2 - \varepsilon^2 \geqslant \sum_{r_2=1}^{R_2-1} \lambda_{r_2}$

5:    Compute tensor $\underline{\mathbf{Z}}^{(2)} = \underline{\mathbf{X}} \times_2 \mathbf{B}^{\mathrm{T}} \in \mathbb{R}^{I \times R_2 \times K}$

6:    Compute EVD of a small matrix $\mathbf{Q}_2 = \mathbf{Z}^{(2)}_{(1)} \mathbf{Z}^{(2)\,\mathrm{T}}_{(1)} \in \mathbb{R}^{I \times I}$ as $\mathbf{Q}_2 = \mathbf{A} \ \mathrm{diag}\,(\lambda_1, \ldots, \lambda_{R_1}) \ \mathbf{A}^{\mathrm{T}}$, such that $\sum_{r_1=1}^{R_1} \lambda_{r_1} \geqslant \|\underline{\mathbf{X}}\|_F^2 - \varepsilon^2 \geqslant \sum_{r_1=1}^{R_1-1} \lambda_{r_1}$

7: **end while**

8: Compute the core tensor $\underline{\mathbf{G}} = \underline{\mathbf{X}} \times_1 \mathbf{A}^{\mathrm{T}} \times_2 \mathbf{B}^{\mathrm{T}}$

9: **return** $\mathbf{A}, \mathbf{B}$ and $\underline{\mathbf{G}}$.

---

in a linked way instead of treating them independently. In such a scenario, the PVD model can be generalized to multi-block matrix/tensor datasets (Cichocki, 2013a; Zhou *et al.*, 2016a,b).

The linked multiway component analysis (LMWCA) for multi-block tensor data can therefore be formulated as a set of approximate simultaneous (joint) Tucker-$(1, N)$ decompositions of a set of data tensors, $\underline{\mathbf{X}}^{(k)} \in \mathbb{R}^{I_1^{(k)} \times I_2^{(k)} \times \cdots \times I_N^{(k)}}$, with $I_1^{(k)} = I_1$ for $k = 1, 2, \ldots, K$, in the form (see Figure 3.13)

$$\underline{\mathbf{X}}^{(k)} = \underline{\mathbf{G}}^{(k)} \times_1 \mathbf{B}^{(1,k)}, \quad (k = 1, 2, \ldots K) \tag{3.47}$$

where each factor (component) matrix, $\mathbf{B}^{(1,k)} = [\mathbf{B}_C^{(1)}, \ \mathbf{B}_I^{(1,k)}] \in \mathbb{R}^{I_1 \times R_k}$, comprises two sets of components: (1) Components $\mathbf{B}_C^{(1)} \in \mathbb{R}^{I_1 \times C}$ (with $0 \leqslant C \leqslant R_k$), $\forall k$, which are common for all the available blocks and correspond to identical or maximally correlated components, and (2) components $\mathbf{B}_I^{(1,k)} \in \mathbb{R}^{I_1 \times (R_k - C)}$, which are different
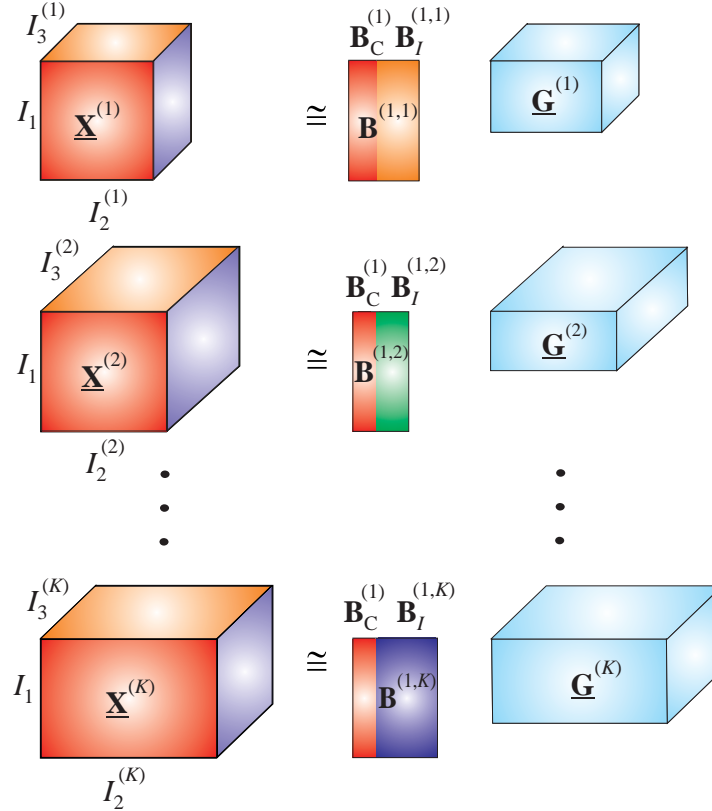
Figure 3.13: Linked Multiway Component Analysis (LMWCA) for coupled 3rd-order data tensors $\underline{\mathbf{X}}^{(1)}, \ldots, \underline{\mathbf{X}}^{(K)}$; these can have different dimensions in every mode, except for the mode-1 for which the size is I_1 for all $\underline{\mathbf{X}}^{(k)}$. Linked Tucker-1 decompositions are then performed in the form $\underline{\mathbf{X}}^{(k)} \cong \underline{\mathbf{G}}^{(k)} \times_1 \mathbf{B}^{(1,k)}$, where partially correlated factor matrices are $\mathbf{B}^{(1,k)} = [\mathbf{B}_C^{(1)}, \mathbf{B}_I^{(1,k)}] \in \mathbb{R}^{I_1 \times R_k}$, $(k = 1, 2, \ldots, K)$. The objective is to find the common components, $\mathbf{B}_C^{(1)} \in \mathbb{R}^{I_1 \times C}$, and individual components, $\mathbf{B}_I^{(1,k)} \in \mathbb{R}^{I_1 \times (R_k - C)}$, where $C \leqslant \min\{R_1, \ldots, R_K\}$ is the number of common components in mode-1.

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 (strongly correlated) components, $\mathbf{B}_C^{(1)}$, and statistically independent (individual) components, $\mathbf{B}_I^{(1,k)}$ (Cichocki, 2013a).

(a)
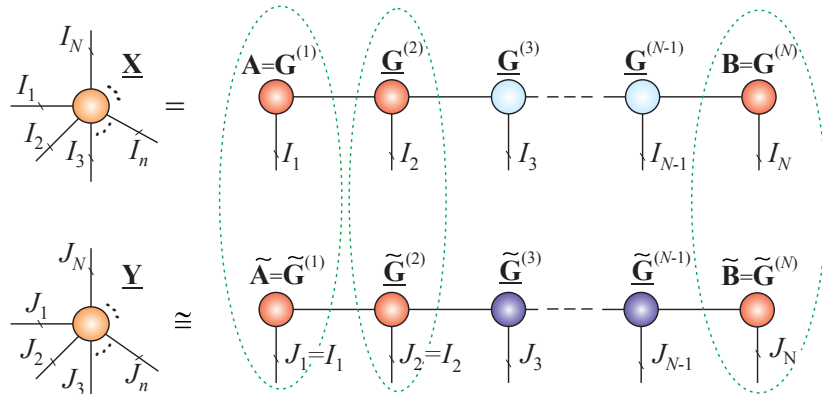
(b)
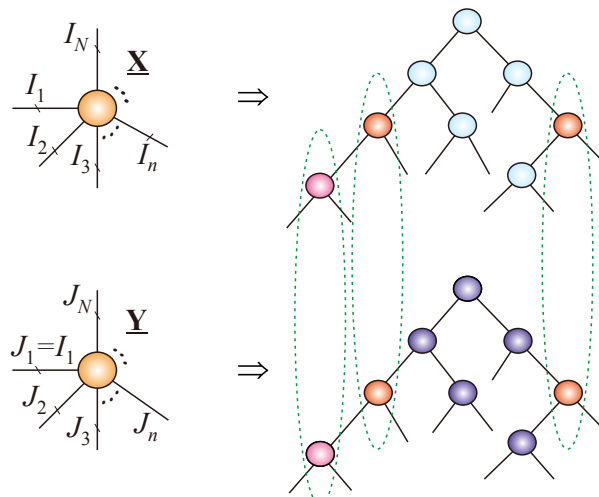
Figure 3.14: Conceptual models of generalized Linked Multiway Component Analysis (LMWCA) applied to the cores of high-order TNs. The objective is to find a suitable tensor decomposition which yields the maximum number of cores that are as much correlated as possible. (a) Linked Tensor Train (TT) networks. (b) Linked Hierarchical Tucker (HT) networks with the correlated cores indicated by ellipses in broken lines.

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 (Phan and Cichocki, 2010).

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, e.g., $\mathbf{X}_{(1)}^{(k)} \cong \mathbf{B}_C^{(1)}\mathbf{A}_C^{(1,k)} + \mathbf{B}_I^{(1,k)}\mathbf{A}_I^{(1,k)}$, through solving the constrained optimization problems

$$
\begin{aligned}
&\min \ \sum_{k=1}^{K} \|\mathbf{X}_{(1)}^{(k)} - \mathbf{B}_C^{(1)}\mathbf{A}_C^{(1,k)} - \mathbf{B}_I^{(1,k)}\mathbf{A}_I^{(1,k)}\|_F \\
&+ P(\mathbf{B}_C^{(1)}), \ \ s.t. \ \ \mathbf{B}_C^{(1)\ \mathrm{T}}\mathbf{B}_I^{(1,k)} = \mathbf{0} \ \forall k,
\end{aligned}
\tag{3.48}
$$

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. The key point is to assume that common factor submatrices, $\mathbf{B}_C^{(1)}$, 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* (Zhou *et al.*, 2016a).

The linked multiway component analysis (LMWCA) model complements currently available techniques for group component analysis and feature extraction from multi-block datasets, and is a natural extension of group ICA, PVD, and CCA/PLS methods (see (Cichocki, 2013a; Zhao *et al.*, 2013a; Zhou *et al.*, 2016a,b) and references therein). Moreover, the concept of LMWCA can be generalized to tensor networks, as illustrated in Figure 3.14.

3.7   Nonlinear Tensor Decompositions – Infinite Tucker

The Infinite Tucker model and its modification, the Distributed Infinite Tucker (DinTucker), generalize the standard Tucker decomposition to infinitely dimensional feature spaces using kernel and Bayesian approaches (Tang *et al.*, 2013; Xu *et al.*, 2012; Zhe *et al.*, 2016).

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

$$\underline{\mathbf{X}} = \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)}$$
$$= [\![\underline{\mathbf{G}}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N)}]\!] \tag{3.49}$$

in its vectorized version

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

Furthermore, assume that the noisy data tensor is modeled as

$$\underline{\mathbf{Y}} = \underline{\mathbf{X}} + \underline{\mathbf{E}}, \tag{3.50}$$

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 $\underline{\mathbf{X}}$ (Chu and Ghahramani, 2009; Xu *et al.*, 2012; Zhe *et al.*, 2016) in the form

$$p\left(\underline{\mathbf{X}} \,|\, \mathbf{B}^{(1)}, \ldots, \mathbf{B}^{(N)}\right) = \mathcal{N}\left(\operatorname{vec}(\underline{\mathbf{X}}); \mathbf{0}, \mathbf{C}^{(1)} \otimes_L \cdots \otimes_L \mathbf{C}^{(N)}\right)$$
$$= \frac{\exp\left(-\frac{1}{2}\|[\![\underline{\mathbf{X}}; (\mathbf{C}^{(1)})^{-1/2}, \ldots, (\mathbf{C}^{(N)})^{-1/2}]\!]\|_F^2\right)}{(2\pi)^{I/2} \prod_{n=1}^{N} |\mathbf{C}^{(n)}|^{-I/(2I_n)}} \tag{3.51}$$

where $I = \prod_n I_n$ and $\mathbf{C}^{(n)} = \mathbf{B}^{(n)} \mathbf{B}^{(n)\,\mathrm{T}} \in \mathbb{R}^{I_n \times I_n}$ 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 (Zhao *et al.*, 2013b), whereby the nonlinear covariance matrix $\mathbf{C}^{(n)} = k(\mathbf{B}^{(n)}, \mathbf{B}^{(n)})$ replaces the standard covariance

matrix, $\mathbf{B}^{(n)}\mathbf{B}^{(n)\,\mathrm{T}}$. Using such a nonlinear feature mapping, the original Tucker factorization is performed in an infinite feature space, while Eq. (3.51) defines a Gaussian process (GP) on a tensor, called the Tensor-variate GP (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 $\underline{\mathbf{Y}}$, the joint probability density function is given by

$$p(\underline{\mathbf{Y}}, \underline{\mathbf{X}}, \{\mathbf{B}^{(n)}\}) = p(\{\mathbf{B}^{(n)}\})\, p(\underline{\mathbf{X}} \,|\, \{\mathbf{B}^{(n)}\})\, p(\underline{\mathbf{Y}}|\underline{\mathbf{X}}). \qquad (3.52)$$

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 with 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

$$\begin{aligned}
p(\{\tilde{\mathbf{B}}_k^{(n)}\}|\{\mathbf{B}^{(n)}\}) &= \prod_{n=1}^{N} p(\tilde{\mathbf{B}}_k^{(n)}|\mathbf{B}^{(n)}) \\
&= \prod_{n=1}^{N} \mathcal{N}(\mathrm{vec}(\tilde{\mathbf{B}}_k^{(n)})|\mathrm{vec}(\mathbf{B}^{(n)})), \lambda\mathbf{I}), \qquad (3.53)
\end{aligned}$$

where $\lambda > 0$ is a variance parameter which controls the similarity between the corresponding factor matrices. The above model is referred to as DinTucker (Zhe *et al.*, 2016).

The full covariance matrix, $\mathbf{C}^{(1)} \otimes \cdots \otimes \mathbf{C}^{(N)} \in \mathbb{R}^{\prod_n I_n \times \prod_n I_n}$, may have a prohibitively large size and can 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 (Cichocki *et al.*, 2015a). 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

$$\mathbf{b}_{\mathbf{i}} = [\bar{\mathbf{b}}_{i_1}^{(1)}, \ldots, \bar{\mathbf{b}}_{i_N}^{(N)}] \in \mathbb{R}^{1 \times \sum_{n=1}^{N} R_n}. \qquad (3.54)$$

We can formalize an (unknown) nonlinear transformation as

$$x_{\mathbf{i}} = f(\mathbf{b}_{\mathbf{i}}) = f([\bar{\mathbf{b}}_{i_1}^{(1)}, \ldots, \bar{\mathbf{b}}_{i_N}^{(N)}]) \qquad (3.55)$$

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

$$p\left(\mathbf{f}_{\mathcal{S}}|\{\mathbf{B}^{(n)}\}\right) = \mathcal{N}\left(\mathbf{f}_{\mathcal{S}}|\mathbf{0}, \ k(\mathbf{B}_{\mathcal{S}}, \ \mathbf{B}_{\mathcal{S}})\right), \tag{3.56}$$

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 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

$$\begin{aligned} p(\mathbf{y}, \mathbf{x}, \{\mathbf{B}^{(n)}\}) \qquad\qquad\qquad\qquad\qquad &(3.57)\\ = \prod_{n=1}^{N} \mathcal{N}(\mathrm{vec}(\mathbf{B}^{(n)})|\mathbf{0}, \mathbf{I}) \, \mathcal{N}(\mathbf{x}|\mathbf{0}, k(\mathbf{B}_{\mathcal{S}}, \ \mathbf{B}_{\mathcal{S}})) \, \mathcal{N}(\mathbf{y}|\mathbf{x}, \beta^{-1}\mathbf{I}), \end{aligned}$$

where β represents noise variance.

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, which is based on sparse GP, together with an efficient MapReduce framework which uses a small set of inducing points to break up the dependencies between random function values (Zhe *et al.*, 2016; Titsias, 2009).

4

Tensor Train Decompositions: Graphical Interpretations and Algorithms

Efficient implementation of the various operations in tensor train (TT) formats requires compact and easy-to-understand mathematical and graphical representations (Cichocki, 2013b, 2014). To this end, we next present mathematical formulations of the TT decompositions and demonstrate their advantages in both theoretical and practical scenarios.

4.1   Tensor Train Decomposition – Matrix Product State

The tensor train (TT/MPS) representation of an Nth-order data tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, can be described in several equivalent forms (see Figures 4.1, 4.2 and Table 4.1) listed below:

1. The entry-wise scalar form, given by

$$x_{i_1,i_2,\ldots,i_N} \cong \sum_{r_1,r_2,\ldots,r_{N-1}=1}^{R_1,R_2,\ldots,R_{N-1}} g_{1,i_1,r_1}^{(1)} \; g_{r_1,i_2,r_2}^{(2)} \cdots g_{r_{N-1},i_N,1}^{(N)}.$$

(4.1)

2. The slice representation (see Figure 2.19) in the form

$$x_{i_1,i_2,\ldots,i_N} \cong \mathbf{G}_{i_1}^{(1)} \; \mathbf{G}_{i_2}^{(2)} \cdots \mathbf{G}_{i_N}^{(N)},$$

(4.2)

where the slice matrices are defined as

$$\mathbf{G}_{i_n}^{(n)} = \underline{\mathbf{G}}^{(n)}(:,i_n,:) \in \mathbb{R}^{R_{n-1} \times R_n}, \quad i_n = 1,2,\ldots,I_n$$

363

(a)
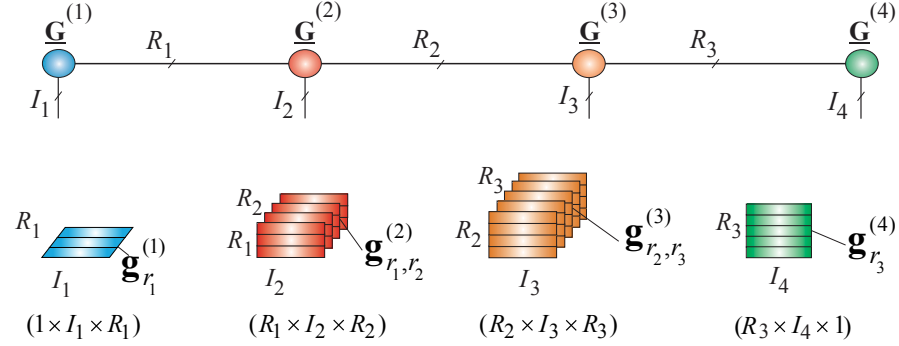
(b)
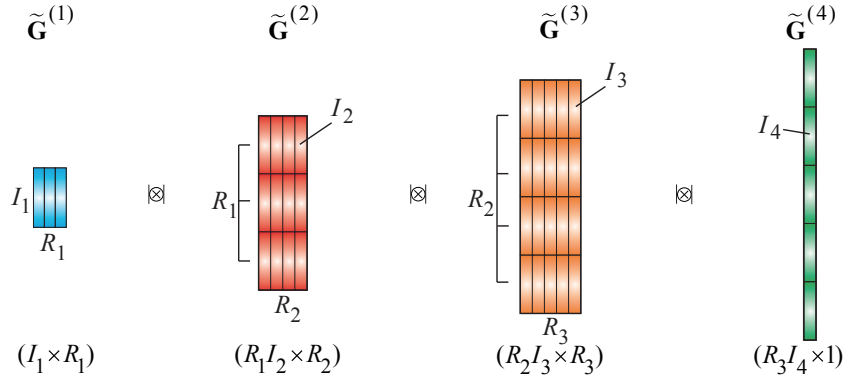
Figure 4.1: TT decomposition of a 4th-order tensor, $\underline{\mathbf{X}}$, for which the TT rank is $R_1 = 3$, $R_2 = 4$, $R_3 = 5$. (a) (Upper panel) Representation of the TT via a multilinear product of the cores, $\underline{\mathbf{X}} \cong \underline{\mathbf{G}}^{(1)} \times^1 \underline{\mathbf{G}}^{(2)} \times^1 \underline{\mathbf{G}}^{(3)} \times^1 \underline{\mathbf{G}}^{(4)} = \langle\!\langle \underline{\mathbf{G}}^{(1)}, \underline{\mathbf{G}}^{(2)}, \underline{\mathbf{G}}^{(3)}, \underline{\mathbf{G}}^{(4)} \rangle\!\rangle$, and (lower panel) an equivalent representation via the outer product of mode-2 fibers (sum of rank-1 tensors) in the form, $\underline{\mathbf{X}} \cong \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} \sum_{r_4=1}^{R_4} (\mathbf{g}_{r_1}^{(1)} \circ \mathbf{g}_{r_1,r_2}^{(2)} \circ \mathbf{g}_{r_2,r_3}^{(3)} \circ \mathbf{g}_{r_3}^{(4)})$. (b) TT decomposition in a vectorized form represented via strong Kronecker products of block matrices, $\mathbf{x} \cong \widetilde{\mathbf{G}}^{(1)} |\!\otimes\!| \widetilde{\mathbf{G}}^{(2)} |\!\otimes\!| \widetilde{\mathbf{G}}^{(3)} |\!\otimes\!| \widetilde{\mathbf{G}}^{(4)} \in \mathbb{R}^{I_1 I_2 I_3 I_4}$, where the block matrices are defined as $\widetilde{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} I_n \times R_n}$, with block vectors $\mathbf{g}_{r_{n-1}, r_n}^{(n)} \in \mathbb{R}^{I_n \times 1}$, $n = 1, \dots, 4$ and $R_0 = R_4 = 1$.

(a)
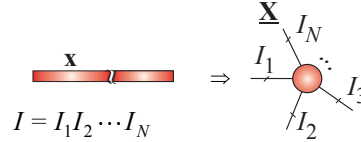
(b)
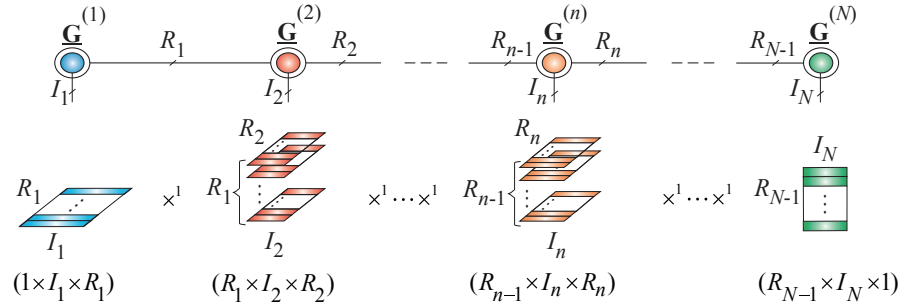
(c)
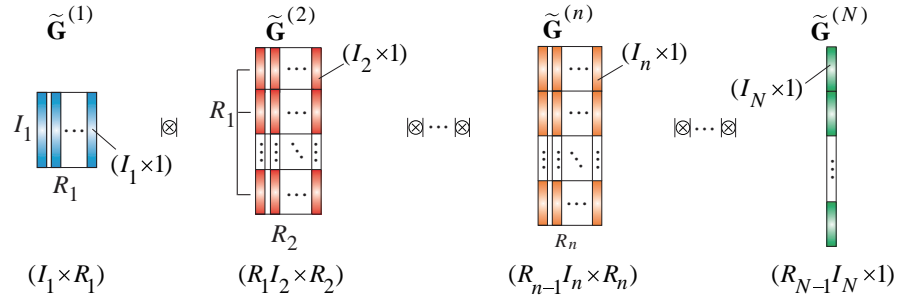
Figure 4.2: TT/MPS decomposition of an Nth-order data tensor, $\underline{\mathbf{X}}$, for which the TT rank is $\{R_1, R_2, \ldots, R_{N-1}\}$. (a) Tensorization of a huge-scale vector, $\mathbf{x} \in \mathbb{R}^I$, into an Nth-order tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. (b) The data tensor can be represented exactly or approximately via a tensor train (TT/MPS), consisting of 3rd-order cores in the form $\underline{\mathbf{X}} \cong \underline{\mathbf{G}}^{(1)} \times^1 \underline{\mathbf{G}}^{(2)} \times^1 \cdots \times^1 \underline{\mathbf{G}}^{(N)} = \langle\!\langle \underline{\mathbf{G}}^{(1)}, \underline{\mathbf{G}}^{(2)}, \ldots, \underline{\mathbf{G}}^{(N)} \rangle\!\rangle$, where $\underline{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$ for $n = 1, 2, \ldots, N$ with $R_0 = R_N = 1$. (c) Equivalently, using the strong Kronecker products, the TT tensor can be expressed in a vectorized form, $\mathbf{x} \cong \widetilde{\mathbf{G}}^{(1)} |\otimes| \widetilde{\mathbf{G}}^{(2)} |\otimes| \cdots |\otimes| \widetilde{\mathbf{G}}^{(N)} \in \mathbb{R}^{I_1 I_2 \cdots I_N}$, where the block matrices are defined as $\widetilde{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} I_n \times R_n}$, with blocks $\mathbf{g}_{r_{n-1}, r_n}^{(n)} \in \mathbb{R}^{I_n \times 1}$.

Table 4.1: Equivalent representations of the Tensor Train decomposition (MPS with open boundary conditions) approximating an Nth-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. It is assumed that the TT rank is $\mathbf{r}_{TT} = \{R_1, R_2, \ldots, R_{N-1}\}$, with $R_0 = R_N = 1$.

Tensor representation: Multilinear products of TT-cores

$$\underline{\mathbf{X}} = \underline{\mathbf{G}}^{(1)} \times^1 \underline{\mathbf{G}}^{(2)} \times^1 \cdots \times^1 \underline{\mathbf{G}}^{(N)} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$$

with the 3rd-order cores $\underline{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}, \ (n = 1, 2, \ldots, N)$

Tensor representation: Outer products

$$\underline{\mathbf{X}} = \sum_{r_1, r_2, \ldots, r_{N-1}=1}^{R_1, R_2, \ldots, R_{N-1}} \mathbf{g}^{(1)}_{1, r_1} \circ \mathbf{g}^{(2)}_{r_1, r_2} \circ \cdots \circ \mathbf{g}^{(N-1)}_{r_{N-2}, r_{N-1}} \circ \mathbf{g}^{(N)}_{r_{N-1}, 1}$$

where $\mathbf{g}^{(n)}_{r_{n-1}, r_n} = \underline{\mathbf{G}}^{(n)}(r_{n-1}, :, r_n) \in \mathbb{R}^{I_n}$ are fiber vectors.

Vector representation: Strong Kronecker products

$$\mathbf{x} = \tilde{\mathbf{G}}^{(1)} \,|\otimes|\, \tilde{\mathbf{G}}^{(2)} \,|\otimes| \cdots |\otimes|\, \tilde{\mathbf{G}}^{(N)} \in \mathbb{R}^{I_1 I_2 \cdots I_N}, \quad \text{where}$$

$\tilde{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} I_n \times R_n}$ are block matrices with blocks $\mathbf{g}^{(n)}_{r_{n-1}, r_n} \in \mathbb{R}^{I_n}$

Scalar representation

$$x_{i_1, i_2, \ldots, i_N} = \sum_{r_1, r_2, \ldots, r_{N-1}=1}^{R_1, R_2, \ldots, R_{N-1}} g^{(1)}_{1, i_1, r_1} \, g^{(2)}_{r_1, i_2, r_2} \cdots g^{(N-1)}_{r_{N-2}, i_{N-1}, r_{N-1}} g^{(N)}_{r_{N-1}, i_N, 1}$$

where $g^{(n)}_{r_{n-1}, i_n, r_n}$ are entries of a 3rd-order core $\underline{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$

Slice (MPS) representation

$$x_{i_1, i_2, \ldots, i_N} = \mathbf{G}^{(1)}_{i_1} \mathbf{G}^{(2)}_{i_2} \cdots \mathbf{G}^{(N)}_{i_N}, \quad \text{where}$$

$\mathbf{G}^{(n)}_{i_n} = \underline{\mathbf{G}}^{(n)}(:, i_n, :) \in \mathbb{R}^{R_{n-1} \times R_n}$ are lateral slices of $\underline{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$

Table 4.2: Equivalent representations of the Tensor Chain (TC) decomposition (MPS with periodic boundary conditions) approximating an Nth-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. It is assumed that the TC rank is $\mathbf{r}_{TC} = \{R_1, R_2, \ldots, R_{N-1}, R_N\}$.

Tensor representation: Trace of multilinear products of cores

$$\underline{\mathbf{X}} = \mathrm{Tr}\,(\underline{\mathbf{G}}^{(1)} \times^1 \underline{\mathbf{G}}^{(2)} \times^1 \cdots \times^1 \underline{\mathbf{G}}^{(N)}) \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$$

with the 3rd-order cores $\underline{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$, $R_0 = R_N$, $n = 1, 2, \ldots, N$

Tensor/Vector representation: Outer/Kronecker products

$$\underline{\mathbf{X}} = \sum_{r_1, r_2, \ldots, r_N = 1}^{R_1, R_2, \ldots, R_N} \mathbf{g}^{(1)}_{r_N, r_1} \circ \mathbf{g}^{(2)}_{r_1, r_2} \circ \cdots \circ \mathbf{g}^{(N)}_{r_{N-1}, r_N} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$$

$$\mathbf{x} = \sum_{r_1, r_2, \ldots, r_N = 1}^{R_1, R_2, \ldots, R_N} \mathbf{g}^{(1)}_{r_N, r_1} \otimes_L \mathbf{g}^{(2)}_{r_1, r_2} \otimes_L \cdots \otimes_L \mathbf{g}^{(N)}_{r_{N-1}, r_N} \in \mathbb{R}^{I_1 I_2 \cdots I_N}$$

where $\mathbf{g}^{(n)}_{r_{n-1}, r_n} \in \mathbb{R}^{I_n}$ are fiber vectors within $\underline{\mathbf{G}}^{(n)}(r_{n-1}, :, r_n) \in \mathbb{R}^{I_n}$

Vector representation: Strong Kronecker products

$$\mathbf{x} = \sum_{r_N = 1}^{R_N} (\widetilde{\mathbf{G}}^{(1)}_{r_N} \,|\!\otimes\!|\, \widetilde{\mathbf{G}}^{(2)} \,|\!\otimes\!|\, \cdots \,|\!\otimes\!|\, \widetilde{\mathbf{G}}^{(N-1)} \,|\!\otimes\!|\, \widetilde{\mathbf{G}}^{(N)}_{r_N}) \in \mathbb{R}^{I_1 I_2 \cdots I_N} \qquad \text{where}$$

$\widetilde{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} I_n \times R_n}$ are block matrices with blocks $\mathbf{g}^{(n)}_{r_{n-1}, r_n} \in \mathbb{R}^{I_n}$,

$\widetilde{\mathbf{G}}^{(1)}_{r_N} \in \mathbb{R}^{I_1 \times R_1}$ is a matrix with blocks (columns) $\mathbf{g}^{(1)}_{r_N, r_1} \in \mathbb{R}^{I_1}$,

$\widetilde{\mathbf{G}}^{(N)}_{r_N} \in \mathbb{R}^{R_{N-1} I_N \times 1}$ is a block vector with blocks $\mathbf{g}^{(N)}_{r_{N-1}, r_N} \in \mathbb{R}^{I_N}$

Scalar representations

$$x_{i_1, i_2, \ldots, i_N} = \mathrm{tr}(\mathbf{G}^{(1)}_{i_1} \mathbf{G}^{(2)}_{i_2} \cdots \mathbf{G}^{(N)}_{i_N}) = \sum_{r_N = 1}^{R_N} (\mathbf{g}^{(1)\,\mathrm{T}}_{r_N, i_1, :} \mathbf{G}^{(2)}_{i_2} \cdots \mathbf{G}^{(N-1)}_{i_{N-1}} \mathbf{g}^{(N)}_{:, i_N, r_N})$$

where $\mathbf{g}^{(1)}_{r_N, i_1, :} = \underline{\mathbf{G}}^{(1)}(r_N, i_1, :) \in \mathbb{R}^{R_1}$, $\mathbf{g}^{(N)}_{:, i_N, r_N} = \underline{\mathbf{G}}^{(N)}(:, i_N, r_N) \in \mathbb{R}^{R_{N-1}}$

---

with $\mathbf{G}_{i_n}^{(n)}$ being the i_nth lateral slice of the core $\underline{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$, $n = 1, 2, \ldots, N$ and $R_0 = R_N = 1$.

3. The (global) tensor form, based on multilinear products (contraction) of cores (see Figure 4.1(a)) given by

$$
\begin{aligned}
\underline{\mathbf{X}} &\cong \underline{\mathbf{G}}^{(1)} \times^1 \underline{\mathbf{G}}^{(2)} \times^1 \cdots \times^1 \underline{\mathbf{G}}^{(N-1)} \times^1 \underline{\mathbf{G}}^{(N)} \\
&= \langle\!\langle \underline{\mathbf{G}}^{(1)}, \underline{\mathbf{G}}^{(2)}, \ldots, \underline{\mathbf{G}}^{(N-1)}, \underline{\mathbf{G}}^{(N)} \rangle\!\rangle,
\end{aligned} \tag{4.3}
$$

where the 3rd-order cores[1] $\underline{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$, $n = 1, 2, \ldots, N$ and $R_0 = R_N = 1$ (see also Figure 4.2(b)).

4. The tensor form, expressed as a sum of rank-1 tensors (see Figure 4.1(a))

$$
\underline{\mathbf{X}} \cong \sum_{r_1, r_2, \ldots, r_{N-1}=1}^{R_1, R_2, \ldots, R_{N-1}} \mathbf{g}_{1, r_1}^{(1)} \circ \mathbf{g}_{r_1, r_2}^{(2)} \circ \cdots \circ \mathbf{g}_{r_{N-2}, r_{N-1}}^{(N-1)} \circ \mathbf{g}_{r_{N-1}, 1}^{(N)}, \tag{4.4}
$$

where $\mathbf{g}_{r_{n-1}, r_n}^{(n)} = \underline{\mathbf{G}}^{(n)}(r_{n-1}, :, r_n) \in \mathbb{R}^{I_n}$ are mode-2 fibers, $n = 1, 2, \ldots, N$ and $R_0 = R_N = 1$.

5. A vector form, expressed by Kronecker products of the fibers

$$
\begin{aligned}
\mathbf{x} &\cong \sum_{r_1, r_2, \ldots, r_{N-1}=1}^{R_1, R_2, \ldots, R_{N-1}} \mathbf{g}_{1, r_1}^{(1)} \otimes_L \mathbf{g}_{r_1, r_2}^{(2)} \otimes_L \\
&\quad \cdots \otimes_L \mathbf{g}_{r_{N-2}, r_{N-1}}^{(N-1)} \otimes_L \mathbf{g}_{r_{N-1}, 1}^{(N)},
\end{aligned} \tag{4.5}
$$

where $\mathbf{x} = \mathrm{vec}(\mathbf{X}) \in \mathbb{R}^{I_1 I_2 \cdots I_N}$.

6. An alternative vector form, produced by strong Kronecker products of block matrices (see Figure 4.1(b)) and Figure 4.2(c)), given by

$$
\mathbf{x} \cong \widetilde{\mathbf{G}}^{(1)} \,|\!\otimes\!|\, \widetilde{\mathbf{G}}^{(2)} \,|\!\otimes\!|\, \cdots \,|\!\otimes\!|\, \widetilde{\mathbf{G}}^{(N)}, \tag{4.6}
$$

where the block matrices $\widetilde{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} I_n \times R_n}$, for $n = 1, 2, \ldots, N$, consist of blocks $\mathbf{g}_{r_{n-1}, r_n}^{(n)} \in \mathbb{R}^{I_n \times 1}$, $n = 1, 2, \ldots, N$, with $R_0 =$

---

[1] Note that the cores $\underline{\mathbf{G}}^{(1)}$ and $\underline{\mathbf{G}}^{(N)}$ are now two-dimensional arrays (matrices), but for a uniform representation, we assume that these matrices are treated as 3rd-order cores of sizes $1 \times I_1 \times R_1$ and $R_{N-1} \times I_N \times 1$, respectively.

$R_N = 1$, and the symbol $|\otimes|$ denotes the strong Kronecker product.

Analogous relationships can be established for Tensor Chain (i.e., MPS with PBC (see Figure 2.19(b)) and summarized in Table 4.2.

4.2 Matrix TT Decomposition – Matrix Product Operator

The matrix tensor train, also called the Matrix Product Operator (MPO) with open boundary conditions (TT/MPO), is an important TN model which first represents huge-scale structured matrices, $\mathbf{X} \in \mathbb{R}^{I \times J}$, as $2N$th-order tensors, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times J_1 \times I_2 \times J_2 \times \cdots I_N \times J_N}$, where $I = I_1 I_2 \cdots I_N$ and $J = J_1 J_2 \cdots J_N$ (see Figures 4.3, 4.4 and Table 4.3). Then, the matrix TT/MPO converts such a $2N$th-order tensor into a chain (train) of 4th-order cores[2]. It should be noted that the matrix TT decomposition is equivalent to the vector TT, created by merging all index pairs (i_n, j_n) into a single index ranging from 1 to $I_n J_n$, in a reverse lexicographic order.

Similarly to the vector TT decomposition, a large scale $2N$th-order tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times J_1 \times I_2 \times J_2 \times \cdots \times I_N \times J_N}$, can be represented in a TT/MPO format via the following mathematical representations:

1. The scalar (entry-wise) form

$$x_{i_1,j_1,\ldots,i_N,j_N} \cong \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_{N-1}=1}^{R_{N-1}} g_{1,i_1,j_1,r_1}^{(1)} \ g_{r_1,i_2,j_2,r_2}^{(2)}$$
$$\cdots g_{r_{N-2},i_{N-1},j_{N-1},r_{N-1}}^{(N-1)} \ g_{r_{N-1},i_N,j_N,1}^{(N)}. \quad (4.7)$$

2. The slice representation

$$x_{i_1,j_1,\ldots,i_N,j_N} \cong \mathbf{G}_{i_1,j_1}^{(1)} \ \mathbf{G}_{i_2,j_2}^{(2)} \cdots \mathbf{G}_{i_N,j_N}^{(N)}, \quad (4.8)$$

where $\mathbf{G}_{i_n,j_n}^{(n)} = \underline{\mathbf{G}}^{(n)}(:, i_n, j_n, :) \in \mathbb{R}^{R_{n-1} \times R_n}$ are slices of the cores $\underline{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times J_n \times R_n}$, $n = 1, 2, \ldots, N$ and $R_0 = R_N = 1$.

---

[2]The cores $\underline{\mathbf{G}}^{(1)}$ and $\underline{\mathbf{G}}^{(N)}$ are in fact three-dimensional arrays, however for uniform representation, we treat them as 4th-order cores of sizes $1 \times I_1 \times J_1 \times R_1$ and $R_{N-1} \times I_N \times J_N \times 1$.

3. The compact tensor form based on multilinear products (Figure 4.4(b))

$$
\begin{aligned}
\underline{\mathbf{X}} &\cong \underline{\mathbf{G}}^{(1)} \times^1 \underline{\mathbf{G}}^{(2)} \times^1 \cdots \times^1 \underline{\mathbf{G}}^{(N)} \\
&= \langle\!\langle \underline{\mathbf{G}}^{(1)}, \underline{\mathbf{G}}^{(2)}, \ldots, \underline{\mathbf{G}}^{(N)} \rangle\!\rangle,
\end{aligned}
\tag{4.9}
$$

where the TT-cores are defined as $\underline{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times J_n \times R_n}$, $n = 1, 2, \ldots, N$ and $R_0 = R_N = 1$.

4. A matrix form, based on strong Kronecker products of block matrices (Figures 4.3(b) and 4.4(c))

$$
\mathbf{X} \cong \widetilde{\mathbf{G}}^{(1)} \,|\!\otimes\!|\, \widetilde{\mathbf{G}}^{(2)} \,|\!\otimes\!| \cdots |\!\otimes\!| \, \widetilde{\mathbf{G}}^{(N)} \in \mathbb{R}^{I_1 \cdots I_N \times J_1 \cdots J_N},
\tag{4.10}
$$

where $\widetilde{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} I_n \times R_n J_n}$ are block matrices with blocks $\mathbf{G}_{r_{n-1}, r_n}^{(n)} \in \mathbb{R}^{I_n \times J_n}$ and the number of blocks is $R_{n-1} \times R_n$. In a special case, when the TT ranks $R_n = 1$, $\forall n$, the strong Kronecker products simplify into standard (left) Kronecker products.

The strong Kronecker product representation of a TT is probably the most comprehensive and useful form for displaying tensor trains in their vector/matrix form, since it allows us to perform many operations using relatively small block matrices.

Example. For two matrices (in the TT format) expressed via the strong Kronecker products, $\mathbf{A} = \widetilde{\mathbf{A}}^{(1)} \,|\!\otimes\!|\, \widetilde{\mathbf{A}}^{(2)} \,|\!\otimes\!| \cdots |\!\otimes\!|\, \widetilde{\mathbf{A}}^{(N)}$ and $\mathbf{B} = \widetilde{\mathbf{B}}^{(1)} \,|\!\otimes\!|\, \widetilde{\mathbf{B}}^{(2)} \,|\!\otimes\!| \cdots |\!\otimes\!|\, \widetilde{\mathbf{B}}^{(N)}$, their Kronecker product can be efficiently computed as $\mathbf{A} \otimes_L \mathbf{B} = \widetilde{\mathbf{A}}^{(1)} \,|\!\otimes\!| \cdots |\!\otimes\!|\, \widetilde{\mathbf{A}}^{(N)} \,|\!\otimes\!|\, \widetilde{\mathbf{B}}^{(1)} \,|\!\otimes\!| \cdots |\!\otimes\!|\, \widetilde{\mathbf{B}}^{(N)}$. Furthermore, if the matrices $\mathbf{A}$ and $\mathbf{B}$ have the same mode sizes[3], then their linear combination, $\mathbf{C} = \alpha\mathbf{A} + \beta\mathbf{B}$ can be compactly expressed as (Oseledets, 2011; Kazeev *et al.*, 2013a,b)

$$
\mathbf{C} = [\widetilde{\mathbf{A}}^{(1)} \ \widetilde{\mathbf{B}}^{(1)}] \,|\!\otimes\!| \begin{bmatrix} \widetilde{\mathbf{A}}^{(2)} & \mathbf{0} \\ \mathbf{0} & \widetilde{\mathbf{B}}^{(2)} \end{bmatrix} |\!\otimes\!| \cdots |\!\otimes\!| \begin{bmatrix} \widetilde{\mathbf{A}}^{(N-1)} & 0 \\ 0 & \widetilde{\mathbf{B}}^{(N-1)} \end{bmatrix} |\!\otimes\!| \begin{bmatrix} \alpha\widetilde{\mathbf{A}}^{(N)} \\ \beta\widetilde{\mathbf{B}}^{(N)} \end{bmatrix}.
$$

Consider its reshaped tensor $\underline{\mathbf{C}} = \langle\!\langle \underline{\mathbf{C}}^{(1)}, \underline{\mathbf{C}}^{(2)}, \ldots, \underline{\mathbf{C}}^{(N)} \rangle\!\rangle$ in the TT format; then its cores $\underline{\mathbf{C}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times J_n \times R_n}$, $n = 1, 2, \ldots, N$ can be

---

[3]Note that, wile original matrices $\mathbf{A} \in \mathbb{R}^{I_1 \cdots I_N \times J_1 \cdots J_N}$ and $\mathbf{B} \in \mathbb{R}^{I_1 \cdots I_N \times J_1 \cdots J_N}$ must have the same mode sizes, the corresponding core tenors, $\underline{\mathbf{A}}^{(n)} = \in \mathbb{R}^{R_{n-1}^A \times I_n \times J_n \times R_n^A}$ and $\underline{\mathbf{B}}^{(n)} = \in \mathbb{R}^{R_{n-1}^B \times I_n \times J_n \times R_n^B}$, may have arbitrary mode sizes.

(a)

$(1 \times I_1 \times J_1 \times R_1)$ $(R_1 \times I_2 \times J_2 \times R_2)$ $(R_2 \times I_3 \times J_3 \times R_3)$ $(R_3 \times I_4 \times J_4 \times 1)$

(b)

$(I_1 \times R_1 J_1)$ $(R_1 I_2 \times R_2 J_2)$ $(R_2 I_3 \times R_3 J_3)$ $(R_3 I_4 \times J_4)$
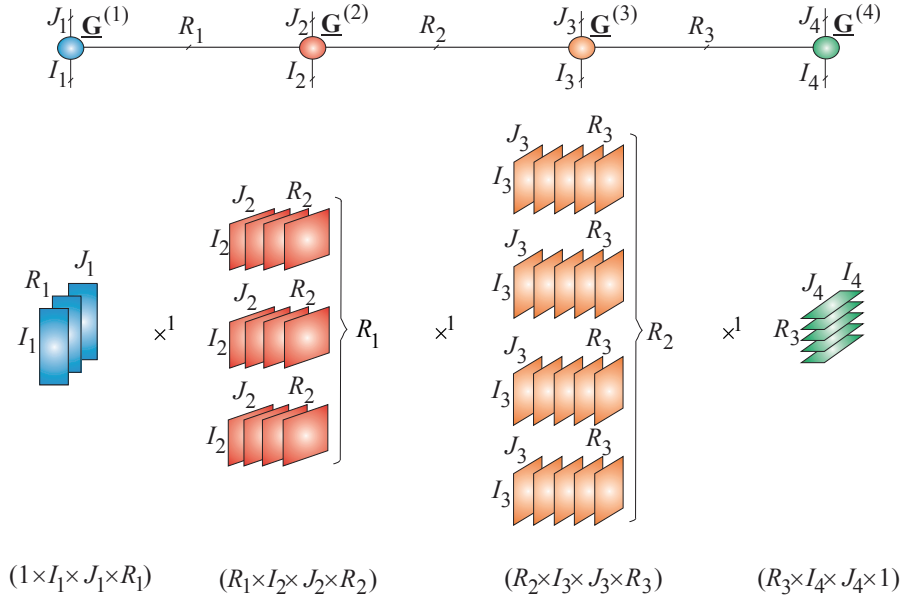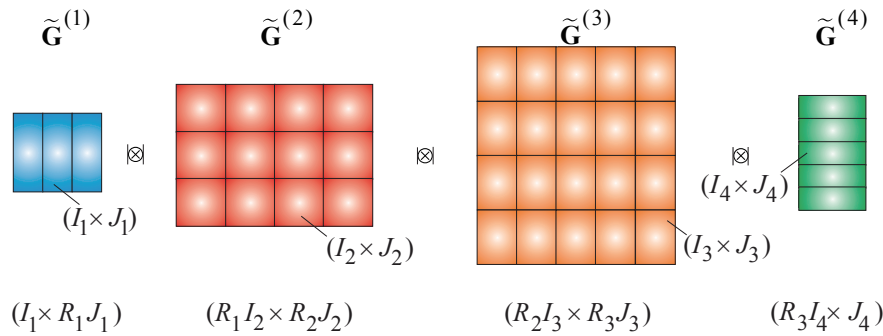
Figure 4.3: TT/MPO decomposition of a matrix, $\mathbf{X} \in \mathbb{R}^{I \times J}$, reshaped as an 8th-order tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times J_1 \times \cdots \times I_4 \times J_4}$, where $I = I_1 I_2 I_3 I_4$ and $J = J_1 J_2 J_3 J_4$. (a) Basic TT representation via multilinear products (tensor contractions) of cores $\underline{\mathbf{X}} = \underline{\mathbf{G}}^{(1)} \times^1 \underline{\mathbf{G}}^{(2)} \times^1 \underline{\mathbf{G}}^{(3)} \times^1 \underline{\mathbf{G}}^{(4)}$, with $\underline{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$ for $R_1 = 3, R_2 = 4, R_3 = 5, R_0 = R_4 = 1$. (b) Representation of a matrix or a matricized tensor via strong Kronecker products of block matrices, in the form $\mathbf{X} = \widetilde{\mathbf{G}}^{(1)} |\otimes| \widetilde{\mathbf{G}}^{(2)} |\otimes| \widetilde{\mathbf{G}}^{(3)} |\otimes| \widetilde{\mathbf{G}}^{(4)} \in \mathbb{R}^{I_1 I_2 I_3 I_4 \times J_1 J_2 J_3 J_4}$.

(a)

(b)

$(1 \times I_1 \times J_1 \times R_1)$ $(R_1 \times I_2 \times J_2 \times R_2)$ $(R_{n-1} \times I_n \times J_n \times R_n)$ $(R_{N-1} \times I_N \times J_N \times 1)$

(c)

$(I_1 \times R_1 J_1)$ $(R_1 I_2 \times R_2 J_2)$ $(R_{n-1} I_n \times R_n J_n)$ $(R_{N-1} I_N \times J_N)$
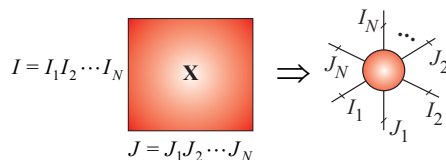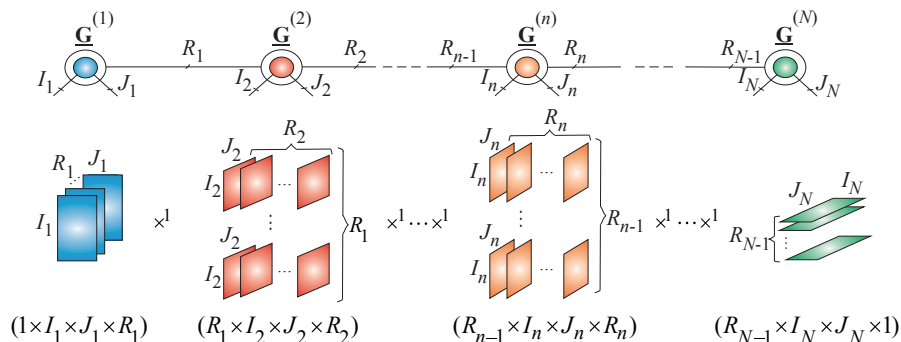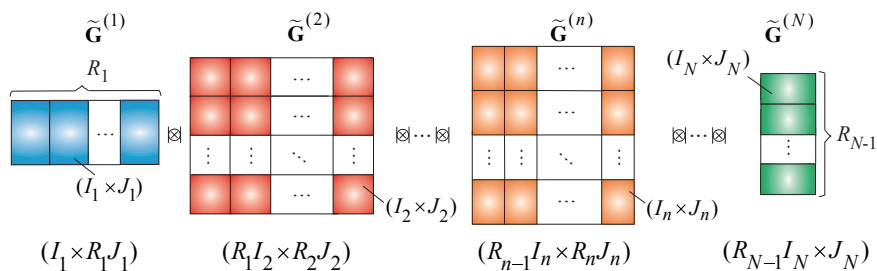
Figure 4.4: Representations of huge matrices by "linked" block matrices. (a) Tensorization of a huge-scale matrix, $\mathbf{X} \in \mathbb{R}^{I \times J}$, into a $2N$th-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times J_2 \times \cdots \times I_N \times J_N}$. (b) The TT/MPO decomposition of a huge matrix, $\mathbf{X}$, expressed by 4th-order cores, $\underline{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times J_n \times R_n}$. (c) Alternative graphical representation of a matrix, $\mathbf{X} \in \mathbb{R}^{I_1 I_2 \cdots I_N \times J_1 J_2 \cdots J_N}$, via strong Kronecker products of block matrices $\widetilde{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} I_n \times R_n J_n}$ for $n = 1, 2, \ldots, N$ with $R_0 = R_N = 1$.

Table 4.3: Equivalent forms of the matrix Tensor Train decomposition (MPO with open boundary conditions) for a $2N$th-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times J_1 \times I_2 \times J_2 \times \cdots \times I_N \times J_N}$. It is assumed that the TT rank is $\{R_1, R_2, \ldots, R_{N-1}\}$, with $R_0 = R_N = 1$.

Tensor representation: Multilinear products (tensor contractions)

$$\underline{\mathbf{X}} = \underline{\mathbf{G}}^{(1)} \times^1 \underline{\mathbf{G}}^{(2)} \times^1 \cdots \times^1 \underline{\mathbf{G}}^{(N-1)} \times^1 \underline{\mathbf{G}}^{(N)}$$

with 4th-order cores $\underline{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times J_n \times R_n}, \quad (n = 1, 2, \ldots, N)$

Tensor representation: Outer products

$$\underline{\mathbf{X}} = \sum_{r_1, r_2, \ldots, r_{N-1}=1}^{R_1, R_2, \ldots, R_{N-1}} \mathbf{G}^{(1)}_{1, r_1} \circ \mathbf{G}^{(2)}_{r_1, r_2} \circ \cdots \circ \mathbf{G}^{(N-1)}_{r_{N-2}, r_{N-1}} \circ \mathbf{G}^{(N)}_{r_{N-1}, 1}$$

where $\mathbf{G}^{(n)}_{r_{n-1}, r_n} \in \mathbb{R}^{I_n \times J_n}$ are blocks of $\tilde{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} I_n \times R_n J_n}$

Matrix representation: Strong Kronecker products

$$\mathbf{X} = \tilde{\mathbf{G}}^{(1)} \;|\otimes|\; \tilde{\mathbf{G}}^{(2)} \;|\otimes|\; \cdots \;|\otimes|\; \tilde{\mathbf{G}}^{(N)} \in \mathbb{R}^{I_1 \cdots I_N \times J_1 \cdots J_N}$$

where $\tilde{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} I_n \times R_n J_n}$ are block matrices with blocks

$\underline{\mathbf{G}}^{(n)}(r_{n-1}, :, :, r_n)$

Scalar representation

$$x_{i_1, j_1, i_2, j_2, \ldots, i_N, j_N} = \sum_{r_1, r_2, \ldots, r_{N-1}=1}^{R_1, R_2, \ldots, R_{N-1}} g^{(1)}_{1, i_1, j_1, r_1} \, g^{(2)}_{r_1, i_2, j_2, r_2} \cdots g^{(N)}_{r_{N-1}, i_N, j_N, 1}$$

where $g^{(n)}_{r_{n-1}, i_n, j_n, r_n}$ are entries of a 4th-order core

$\underline{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times J_n \times R_n}$

Slice (MPS) representation

$$x_{i_1, j_1, i_2, j_2, \ldots, i_N, j_N} = \mathbf{G}^{(1)}_{i_1, j_1} \mathbf{G}^{(2)}_{i_2, j_2} \cdots \mathbf{G}^{(N)}_{i_N, j_N} \qquad \text{where}$$

$$\mathbf{G}^{(n)}_{i_n, j_n} = \underline{\mathbf{G}}^{(n)}(:, i_n, j_n, :) \in \mathbb{R}^{R_{n-1} \times R_n} \text{ are slices of } \underline{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times J_n \times R_n}$$

---

expressed through their unfolding matrices, $\mathbf{C}^{(n)}_{<n>} \in \mathbb{R}^{R_{n-1}I_n \times R_n J_n}$, or equivalently by the lateral slices, $\mathbf{C}^{(n)}_{i_n,j_n} \in \mathbb{R}^{R_{n-1} \times R_n}$, as follows

$$\mathbf{C}^{(n)}_{i_n,j_n} = \begin{bmatrix} \mathbf{A}^{(n)}_{i_n,j_n} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}^{(n)}_{i_n,j_n} \end{bmatrix}, \quad n = 2, 3, \ldots, N-1, \qquad (4.11)$$

while for the border cores

$$\mathbf{C}^{(1)}_{i_1,j_1} = \begin{bmatrix} \mathbf{A}^{(1)}_{i_1,j_1} & \mathbf{B}^{(1)}_{i_1,j_1} \end{bmatrix}, \qquad \mathbf{C}^{(N)}_{i_N,j_N} = \begin{bmatrix} \alpha \, \mathbf{A}^{(N)}_{i_N,j_N} \\ \beta \, \mathbf{B}^{(N)}_{i_N,j_N} \end{bmatrix} \qquad (4.12)$$

for $i_n = 1, 2, \ldots, I_n, \ j_n = 1, 2, \ldots, J_N, \ n = 1, 2, \ldots, N$.

Note that the various mathematical and graphical representations of TT/MPS and TT/MPO can be used interchangeably for different purposes or applications. With these representations, all basic mathematical operations in TT format can be performed on the constitutive block matrices, even without the need to explicitly construct core tensors (Oseledets, 2011; Dolgov, 2014).

Remark. In the TT/MPO paradigm, compression of large matrices is not performed by global (standard) low-rank matrix approximations, but by low-rank approximations of block-matrices (submatrices) arranged in a hierarchical (linked) fashion. However, to achieve a low-rank TT and consequently a good compression ratio, ranks of all the corresponding unfolding matrices of a specific structured data tensor must be low, i.e., their singular values must rapidly decrease to zero. While this is true for many structured matrices, unfortunately in general, this assumption does not hold.

4.3   Links Between CP, BTD Formats and TT/TC Formats

It is important to note that any specific TN format can be converted into the TT format. This very useful property is next illustrated for two simple but important cases which establish links between the CP and TT and the BTD and TT formats.

(a)
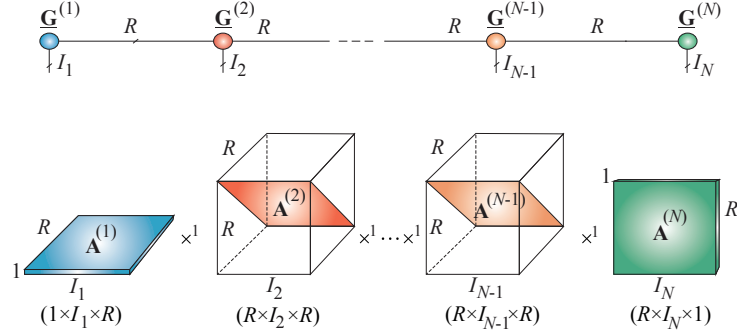
(b)
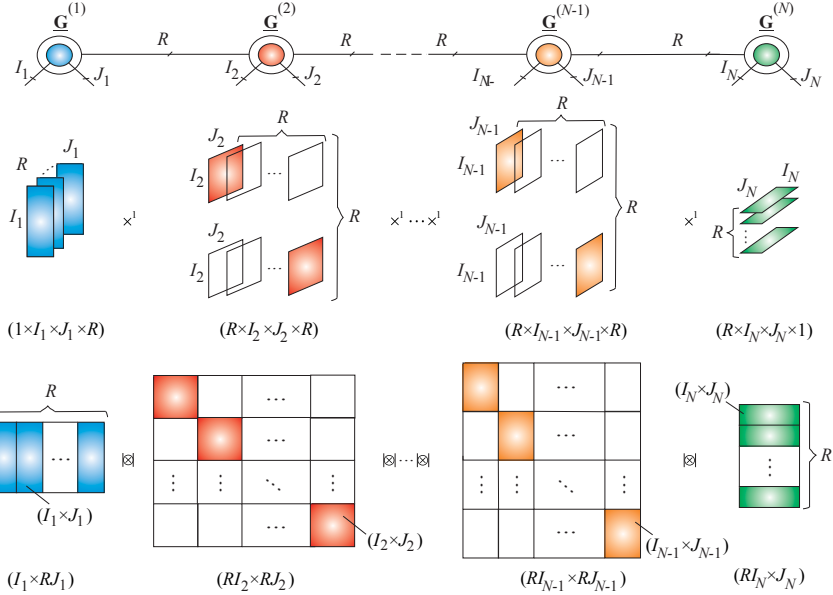
Figure 4.5: Links between the TT format and other tensor network formats. (a) Representation of the CP decomposition for an Nth-order tensor, $\underline{\mathbf{X}} = \underline{\mathbf{I}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \cdots \times_N \mathbf{A}^{(N)}$, in the TT format. (b) Representation of the BTD model given by Eqs. (4.15) and (4.16) in the TT/MPO format. Observe that the TT-cores are very sparse and the TT ranks are $\{R, R, \ldots, R\}$. Similar relationships can be established straightforwardly for the TC format.

1. A tensor in the CP format, given by

$$\underline{\mathbf{X}} = \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \cdots \circ \mathbf{a}_r^{(N)}, \qquad (4.13)$$

can be straightforwardly converted into the TT/MPS format as follows. Since each of the R rank-1 tensors can be represented in the TT format of TT rank $(1, 1, \ldots, 1)$, using formulas (4.11) and (4.12), we have

$$\begin{aligned} \underline{\mathbf{X}} &= \sum_{r=1}^{R} \langle\!\langle \mathbf{a}_r^{(1)\mathrm{T}}, \mathbf{a}_r^{(2)\mathrm{T}}, \ldots, \mathbf{a}_r^{(N)\mathrm{T}} \rangle\!\rangle \qquad (4.14) \\ &= \langle\!\langle \underline{\mathbf{G}}^{(1)}, \underline{\mathbf{G}}^{(2)}, \ldots, \underline{\mathbf{G}}^{(N-1)}, \underline{\mathbf{G}}^{(N)} \rangle\!\rangle, \end{aligned}$$

where the TT-cores $\underline{\mathbf{G}}^{(n)} \in \mathbb{R}^{R \times I_n \times R}$ have diagonal lateral slices $\mathbf{G}^{(n)}(:, i_n, :) = \mathbf{G}_{i_n}^{(n)} = \mathrm{diag}(a_{i_n,1}, a_{i_n,2}, \ldots, a_{i_n,R}) \in \mathbb{R}^{R \times R}$ for $n = 2, 3, \ldots, N-1$ and $\mathbf{G}^{(1)} = \mathbf{A}^{(1)} \in \mathbb{R}^{I_1 \times R}$ and $\mathbf{G}^{(N)} = \mathbf{A}^{(N)\,\mathrm{T}} \in \mathbb{R}^{R \times I_N}$ (see Figure 4.5(a)).

2. A more general Block Term Decomposition (BTD) for a $2N$th-order data tensor

$$\underline{\mathbf{X}} = \sum_{r=1}^{R} (\mathbf{A}_r^{(1)} \circ \mathbf{A}_r^{(2)} \circ \cdots \circ \mathbf{A}_r^{(N)}) \in \mathbb{R}^{I_1 \times J_1 \times \cdots \times I_N \times J_N} \,(4.15)$$

with full rank matrices, $\mathbf{A}_r^{(n)} \in \mathbb{R}^{I_n \times J_n}$, $\forall r$, can be converted into a matrix TT/MPO format, as illustrated in Figure 4.5(b).

Note that (4.15) can be expressed in a matricized (unfolding) form via strong Kronecker products of block diagonal matrices (see formulas (4.11)), given by

$$\begin{aligned} \mathbf{X} &= \sum_{r=1}^{R} (\mathbf{A}_r^{(1)} \otimes_L \mathbf{A}_r^{(2)} \otimes_L \cdots \otimes_L \mathbf{A}_r^{(N)}) \qquad (4.16) \\ &= \widetilde{\mathbf{G}}^{(1)} \,|\!\otimes\!|\, \widetilde{\mathbf{G}}^{(2)} \,|\!\otimes\!|\, \cdots \,|\!\otimes\!|\, \widetilde{\mathbf{G}}^{(N)} \in \mathbb{R}^{I_1 \cdots I_N \,\times\, J_1 \cdots \times J_N}, \end{aligned}$$

with the TT rank, $R_n = R$ for $n = 1, 2, \ldots N-1$, and the block diagonal matrices, $\widetilde{\mathbf{G}}^{(n)} = \mathrm{diag}(\mathbf{A}_1^{(n)}, \mathbf{A}_2^{(n)}, \ldots, \mathbf{A}_R^{(n)}) \in \mathbb{R}^{RI_n \times RJ_n}$, for $n = 2, 3, \ldots, N-1$, while $\widetilde{\mathbf{G}}^{(1)} = [\mathbf{A}_1^{(1)}, \mathbf{A}_2^{(1)}, \ldots, \mathbf{A}_R^{(1)}] \in$

$\mathbb{R}^{I_1 \times RJ_1}$ is a row block matrix, and $\widetilde{\mathbf{G}}^{(N)} = \begin{bmatrix} \mathbf{A}_1^{(N)} \\ \vdots \\ \mathbf{A}_R^{(N)} \end{bmatrix} \in \mathbb{R}^{RI_N \times J_N}$

a column block matrix (see Figure 4.5(b)).

Several algorithms exist for decompositions in the form (4.15) and (4.16) (Salmi *et al.*, 2009; Batselier *et al.*, 2015; Batselier and Wong, 2015). In this way, TT/MPO decompositions for huge-scale structured matrices can be constructed indirectly.

4.4 Quantized Tensor Train (QTT) – Blessing of Dimensionality

The procedure of creating a higher-order tensor from lower-order original data is referred to as tensorization, while in a special case where each mode has a very small size 2, 3 or 4, it is referred to as quantization. In addition to vectors and matrices, lower-order tensors can also be reshaped into higher-order tensors. By virtue of quantization, low-rank TN approximations with high compression ratios can be obtained, which is not possible to achieve with original raw data formats. (Oseledets, 2010; Khoromskij, 2011a).

Therefore, *the quantization can be considered as a special form of tensorization where size of each mode is very small, typically 2 or 3.* The concept of quantized tensor networks (QTN) was first proposed in (Oseledets, 2010) and (Khoromskij, 2011a), whereby low-size 3rd-order cores are sparsely interconnected via tensor contractions. The so obtained model often provides an efficient, highly compressed, and low-rank representation of a data tensor and helps to mitigate the curse of dimensionality, as illustrated below.

Example. The quantization of a huge vector, $\mathbf{x} \in \mathbb{R}^I$, $I = 2^K$, can be achieved through reshaping to give a $(2 \times 2 \times \cdots \times 2)$ tensor $\underline{\mathbf{X}}$ of order K, as illustrated in Figure 4.6. For structured data such a quantized tensor, $\underline{\mathbf{X}}$, often admits low-rank TN approximation, so that a good compression of a huge vector $\mathbf{x}$ can be achieved by enforcing the maximum possible low-rank structure on the tensor network. Even
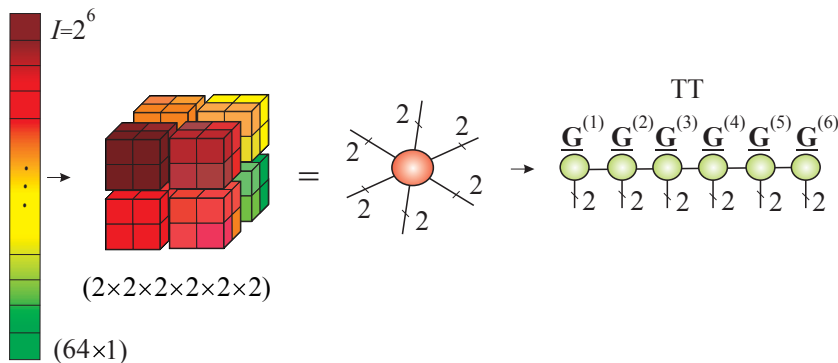
Figure 4.6: Concept of tensorization/quantization of a large-scale vector into a higher-order quantized tensor. In order to achieve a good compression ratio, we need to apply a suitable tensor decomposition such as the quantized TT (QTT) using 3rd-order cores, $\underline{\mathbf{X}} = \underline{\mathbf{G}}^{(1)} \times^1 \underline{\mathbf{G}}^{(2)} \times^1 \cdots \times^1 \underline{\mathbf{G}}^{(6)}$.

more generally, an Nth-order tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, with $I_n = q^{K_n}$, can be quantized in all modes simultaneously to yield a $(q \times q \times \cdots \times q)$ quantized tensor of higher-order and with small value of q.

Example. Since large-scale tensors (even of low-order) cannot be loaded directly into the computer memory, our approach to this problem is to represent the huge-scale data by tensor networks in a distributed and compressed TT format, so as to avoid the explicit requirement for unfeasible large computer memory.

In the example shown in Figure 4.7, the tensor train of a huge 3rd-order tensor is expressed by the strong Kronecker products of block tensors with relatively small 3rd-order tensor blocks. The QTT is mathematically represented in a distributed form via strong Kronecker products of block 5th-order tensors. Recall that the strong Kronecker product of two block core tensors, $\underline{\widetilde{\mathbf{G}}}^{(n)} \in \mathbb{R}^{R_{n-1}I_n \times R_n J_n \times K_n}$ and $\underline{\widetilde{\mathbf{G}}}^{(n+1)} \in \mathbb{R}^{R_n I_{n+1} \times R_{n+1} J_{n+1} \times K_{n+1}}$, is defined as the block tensor, $\underline{\mathbf{C}} = \underline{\widetilde{\mathbf{G}}}^{(n)} |\otimes| \underline{\widetilde{\mathbf{G}}}^{(n+1)} \in \mathbb{R}^{R_{n-1}I_n I_{n+1} \times R_{n+1} J_n J_{n+1} \times K_n K_{n+1}}$, with 3rd-order tensor blocks, $\underline{\mathbf{C}}_{r_{n-1}, r_{n+1}} = \sum_{r_n=1}^{R_n} \underline{\mathbf{G}}_{r_{n-1}, r_n}^{(n)} \otimes_L \underline{\mathbf{G}}_{r_n, r_{n+1}}^{(n+1)} \in \mathbb{R}^{I_n I_{n+1} \times J_n J_{n+1} \times K_n K_{n+1}}$, where $\underline{\mathbf{G}}_{r_{n-1}, r_n}^{(n)} \in \mathbb{R}^{I_n \times J_n \times K_n}$ and

$\mathbf{G}_{r_n, r_{n+1}}^{(n+1)} \in \mathbb{R}^{I_{n+1} \times J_{n+1} \times K_{n+1}}$ are the block tensors of $\widetilde{\underline{\mathbf{G}}}^{(n)}$ and $\widetilde{\underline{\mathbf{G}}}^{(n+1)}$, respectively.

In practice, a fine ($q = 2, 3, 4$) quantization is desirable to create as many virtual (additional) modes as possible, thus allowing for the implementation of efficient low-rank tensor approximations. For example, the binary encoding ($q = 2$) reshapes an Nth-order tensor with $(2^{K_1} \times 2^{K_2} \times \cdots \times 2^{K_N})$ elements into a tensor of order $(K_1 + K_2 + \cdots + K_N)$, with the same number of elements. In other words, the idea is to quantize each of the n "physical" modes (dimensions) by replacing them with K_n "virtual" modes, provided that the corresponding mode sizes, I_n, are factorized as $I_n = I_{n,1} I_{n,2} \cdots I_{n,K_n}$. This, in turn, corresponds to reshaping the nth mode of size I_n into K_n modes of sizes $I_{n,1}, I_{n,2}, \ldots, I_{n,K_n}$.

The TT decomposition applied to quantized tensors is referred to as the QTT, Quantics-TT or Quantized-TT, and was first introduced as a compression scheme for large-scale matrices (Oseledets, 2010), and also independently for more general settings.

The attractive properties of QTT are:

1. Not only QTT ranks are typically small (usually, below 20) but they are also almost independent[4] of the data size (even for $I = 2^{50}$), thus providing a logarithmic (sub-linear) reduction of storage requirements from $\mathcal{O}(I^N)$ to $\mathcal{O}(NR^2 \log_q(I))$ which is referred to as super-compression (Khoromskij, 2011a; Kazeev and Khoromskij, 2012; Kazeev *et al.*, 2013a; Dolgov and Khoromskij, 2013; Dolgov *et al.*, 2014). Comparisons of the storage complexity of various tensor formats are given in Table 4.4.

2. Compared to the TT decomposition (without quantization), *the QTT format often represents deep structures in the data by introducing "virtual" dimensions or modes.* For data which exhibit high degrees of structure, the high compressibility of the QTT approximation is a consequence of the better separability properties of the quantized tensor.

---

[4]At least uniformly bounded.

(a)
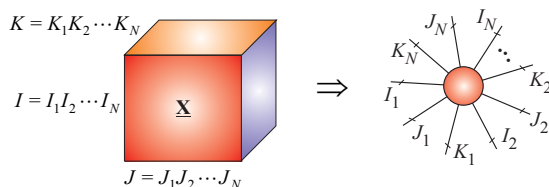
(b)
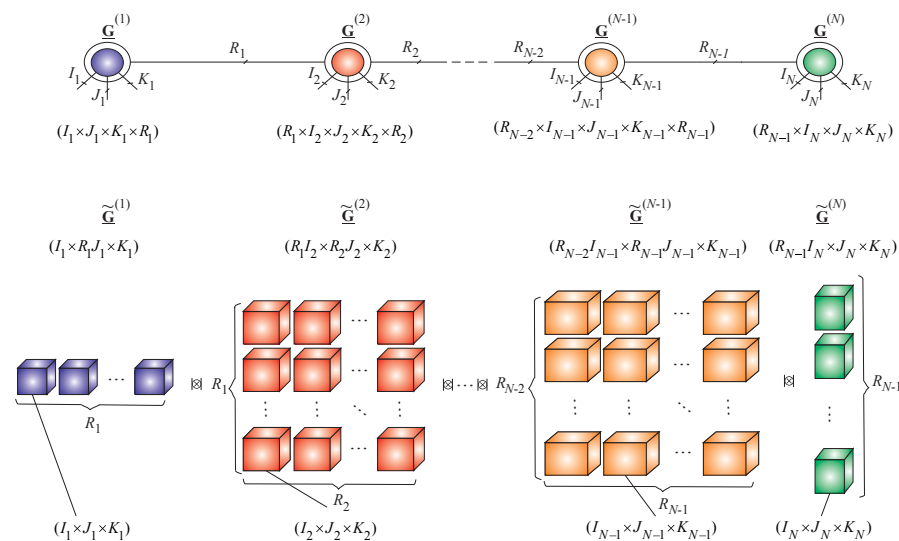
Figure 4.7: Tensorization/quantization of a huge-scale 3rd-order tensor into a higher order tensor and its TT representation. (a) Example of tensorization/quantization of a 3rd-order tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$, into a $3N$th-order tensor, assuming that the mode sizes can be factorized as, $I = I_1 I_2 \cdots I_N$, $J = J_1 J_2 \cdots J_N$ and $K = K_1 K_2 \cdots K_N$. (b) Decomposition of the high-order tensor via a generalized Tensor Train and its representation by the strong Kronecker product of block tensors as $\underline{\mathbf{X}} \cong \underline{\widetilde{\mathbf{G}}}^{(1)} \; |\otimes| \; \underline{\widetilde{\mathbf{G}}}^{(2)} \; |\otimes| \cdots |\otimes| \; \underline{\widetilde{\mathbf{G}}}^{(N)} \in \mathbb{R}^{I_1 \cdots I_N \times J_1 \cdots J_N \times K_1 \cdots K_N}$, where each block $\underline{\widetilde{\mathbf{G}}}^{(n)} \in \mathbb{R}^{R_{n-1} I_n \times R_n J_n \times K_n}$ is also a 3rd-order tensor of size $(I_n \times J_n \times K_n)$, for $n = 1, 2, \ldots, N$ with $R_0 = R_N = 1$. In the special case when $J = K = 1$, the model simplifies into the standard TT/MPS model.

Table 4.4: Storage complexities of tensor decomposition models for an Nth-order tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, for which the original storage complexity is $\mathcal{O}(I^N)$, where $I = \max\{I_1, I_2, \ldots, I_N\}$, while R is the upper bound on the ranks of tensor decompositions considered, that is, $R = \max\{R_1, R_2, \ldots, R_{N-1}\}$ or $R = \max\{R_1, R_2, \ldots, R_N\}$.

1. Full (raw) tensor format	$\mathcal{O}(I^N)$
2. CP	$\mathcal{O}(NIR)$
3. Tucker	$\mathcal{O}(NIR + R^N)$
4. TT/MPS	$\mathcal{O}(NIR^2)$
5. TT/MPO	$\mathcal{O}(NI^2R^2)$
6. Quantized TT/MPS (QTT)	$\mathcal{O}(NR^2 \log_q(I))$
7. QTT+Tucker	$\mathcal{O}(NR^2 \log_q(I) + NR^3)$
8. Hierarchical Tucker (HT)	$\mathcal{O}(NIR + NR^3)$

3. The fact that the QTT ranks are often moderate or even low[5] offers unique advantages in the context of big data analytics (see (Khoromskij, 2011a,b; Kazeev *et al.*, 2013a) and references therein), together with high efficiency of multilinear algebra within the TT/QTT algorithms which rests upon the well-posedness of the low-rank TT approximations.

The ranks of the QTT format often grow dramatically with data size, but with a linear increase in the approximation accuracy. To overcome this problem, Dolgov and Khoromskij proposed the QTT-Tucker format (Dolgov and Khoromskij, 2013) (see Figure 4.8), which exploits the TT approximation not only for the Tucker core tensor, but also for the factor matrices. This model naturally admits distributed computation, and often yields bounded ranks, thus avoiding the curse of dimensionality.

---

[5]The TT/QTT ranks are constant or growing linearly with respect to the tensor order N and are constant or growing logarithmically with respect to the dimension of tensor modes I.

(a)
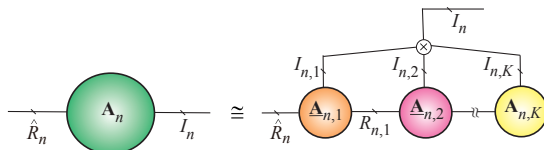
(b)
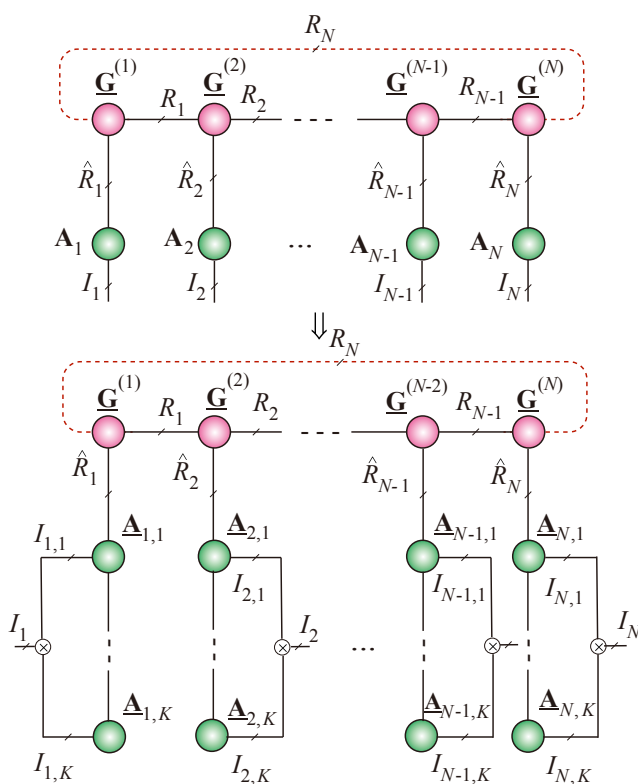
Figure 4.8: The QTT-Tucker or alternatively QTC-Tucker (Quantized Tensor-Chain-Tucker) format. (a) Distributed representation of a matrix $\mathbf{A}_n \in \mathbb{R}^{I_n \times \hat{R}_n}$ with a very large value of I_n via QTT, by tensorization to a high-order quantized tensor, followed by QTT decomposition. (b) Distributed representation of a large-scale Tucker-N model, $\underline{\mathbf{X}} \cong \underline{\mathbf{G}} \times_1 \mathbf{A}_1 \times \mathbf{A}_2 \cdots \times_N \mathbf{A}_N$, via a quantized TC model in which the core tensor $\underline{\mathbf{G}} \in \mathbb{R}^{\hat{R}_1 \times \hat{R}_2 \times \cdots \times \hat{R}_N}$ and optionally all large-scale factor matrices $\mathbf{A}_n$ ($n = 1, 2, \ldots, N$) are represented by MPS models (for more detail see (Dolgov and Khoromskij, 2013)).

The TT/QTT tensor networks have already found application in very large-scale problems in scientific computing, such as in eigenanalysis, super-fast Fourier transforms, and in solving huge systems of large linear equations (see (Dolgov and Khoromskij, 2013; Huckle *et al.*, 2013; Dolgov *et al.*, 2014; Wahls *et al.*, 2014; Kressner *et al.*, 2014a; Kressner and Uschmajew, 2016) and references therein).

4.5 Basic Operations in TT Formats

For big tensors in their TT formats, basic mathematical operations, such as the addition, inner product, computation of tensor norms, Hadamard and Kronecker product, and matrix-by-vector and matrix-by-matrix multiplications can be very efficiently performed using block (slice) matrices of individual (relatively small size) core tensors.

Consider two Nth-order tensors in the TT format

$$\underline{\mathbf{X}} = \langle\!\langle \underline{\mathbf{X}}^{(1)}, \underline{\mathbf{X}}^{(2)}, \dots, \underline{\mathbf{X}}^{(N)} \rangle\!\rangle \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$$
$$\underline{\mathbf{Y}} = \langle\!\langle \underline{\mathbf{Y}}^{(1)}, \underline{\mathbf{Y}}^{(2)}, \dots, \underline{\mathbf{Y}}^{(N)} \rangle\!\rangle \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N},$$

for which the TT ranks are $\mathbf{r}_X = \{R_1, R_2, \dots, R_{N-1}\}$ and $\mathbf{r}_Y = \{\tilde{R}_1, \tilde{R}_2, \dots, \tilde{R}_{N-1}\}$. The following operations can then be performed directly in the TT formats.

Tensor addition. The sum of two tensors

$$\underline{\mathbf{Z}} = \underline{\mathbf{X}} + \underline{\mathbf{Y}} = \langle\!\langle \underline{\mathbf{Z}}^{(1)}, \underline{\mathbf{Z}}^{(2)}, \dots, \underline{\mathbf{Z}}^{(N)} \rangle\!\rangle \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} \qquad (4.17)$$

has the TT rank $\mathbf{r}_Z = \mathbf{r}_X + \mathbf{r}_Y$ and can be expressed via lateral slices of the cores $\underline{\mathbf{Z}} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$ as

$$\mathbf{Z}_{i_n}^{(n)} = \begin{bmatrix} \mathbf{X}_{i_n}^{(n)} & \mathbf{0} \\ \mathbf{0} & \mathbf{Y}_{i_n}^{(n)} \end{bmatrix}, \quad n = 2, 3, \dots, N-1. \qquad (4.18)$$

For the border cores, we have

$$\mathbf{Z}_{i_1}^{(1)} = \begin{bmatrix} \mathbf{X}_{i_1}^{(1)} \ \mathbf{Y}_{i_1}^{(1)} \end{bmatrix}, \qquad \mathbf{Z}_{i_N}^{(N)} = \begin{bmatrix} \mathbf{X}_{i_N}^{(N)} \\ \mathbf{Y}_{i_N}^{(N)} \end{bmatrix} \qquad (4.19)$$

for $i_n = 1, 2, \ldots, I_n, \ \ n = 1, 2, \ldots, N.$

Hadamard product. The computation of the Hadamard (element-wise) product, $\underline{\mathbf{Z}} = \underline{\mathbf{X}} \circledast \underline{\mathbf{Y}}$, of two tensors, $\mathbf{X}$ and $\mathbf{Y}$, of the same order and the same size can be performed very efficiently in the TT format by expressing the slices of the cores, $\underline{\mathbf{Z}} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$, as

$$\mathbf{Z}_{i_n}^{(n)} = \mathbf{X}_{i_n}^{(n)} \otimes \mathbf{Y}_{i_n}^{(n)}, \quad n = 1, \ldots, N, \ \ i_n = 1, \ldots, I_n. \quad (4.20)$$

This increases the TT ranks for the tensor $\underline{\mathbf{Z}}$ to at most $R_n \tilde{R}_n$, $n = 1, 2, \ldots, N$, but the associated computational complexity can be reduced from being exponential in N, $\mathcal{O}(I^N)$, to being linear in both I and N, $\mathcal{O}(IN(R\tilde{R})^2))$.

Super fast Fourier transform of a tensor in the TT format (MAT-LAB functions: fftn($\underline{\mathbf{X}}$) and fft($\underline{\mathbf{X}}^{(n)}, [], 2$)) can be computed as

$$\begin{aligned} \mathcal{F}(\underline{\mathbf{X}}) &= \langle\!\langle \mathcal{F}(\underline{\mathbf{X}}^{(1)}), \mathcal{F}(\underline{\mathbf{X}}^{(2)}), \ldots, \mathcal{F}(\underline{\mathbf{X}}^{(N)}) \rangle\!\rangle \\ &= \mathcal{F}(\underline{\mathbf{X}}^{(1)}) \times^1 \mathcal{F}(\underline{\mathbf{X}}^{(2)}) \times^1 \cdots \times^1 \mathcal{F}(\underline{\mathbf{X}}^{(N)}). \end{aligned} \quad (4.21)$$

It should be emphasized that performing computation of the FFT on relatively small core tensors $\underline{\mathbf{X}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$ *reduces dramatically computational complexity under condition that a data tensor admits low-rank TT approximation.* This approach is referred to as the super fast Fourier transform (SFFT) in TT format. Wavelets, DCT, and other linear integral transformations admit a similar form to the SFFT in (4.21), for example, for the wavelet transform in the TT format, we have

$$\begin{aligned} \mathcal{W}(\underline{\mathbf{X}}) &= \langle\!\langle \mathcal{W}(\underline{\mathbf{X}}^{(1)}), \mathcal{W}(\underline{\mathbf{X}}^{(2)}), \ldots, \mathcal{W}(\underline{\mathbf{X}}^{(N)}) \rangle\!\rangle \\ &= \mathcal{W}(\underline{\mathbf{X}}^{(1)}) \times^1 \mathcal{W}(\underline{\mathbf{X}}^{(2)}) \times^1 \cdots \times^1 \mathcal{W}(\underline{\mathbf{X}}^{(N)}). \end{aligned} \quad (4.22)$$

The N-D discrete convolution in a TT format of tensors $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ with TT rank $\{R_1, R_2, \ldots, R_{N-1}\}$ and $\underline{\mathbf{Y}} \in \mathbb{R}^{J_1 \times \cdots \times J_N}$ with TT rank $\{Q_1, Q_2, \ldots, Q_{N-1}\}$ can be computed as

$$\begin{aligned} \underline{\mathbf{Z}} &= \underline{\mathbf{X}} * \underline{\mathbf{Y}} \hspace{6cm} (4.23) \\ &= \langle\!\langle \underline{\mathbf{Z}}^{(1)}, \underline{\mathbf{Z}}^{(2)}, \ldots, \underline{\mathbf{Z}}^{(N)} \rangle\!\rangle \in \mathbb{R}^{(I_1 + J_1 - 1) \times (I_2 + J_2 - 1) \times \cdots \times (I_N + J_N - 1)}, \end{aligned}$$

with the TT-cores given by

$$\underline{\mathbf{Z}}^{(n)} = \underline{\mathbf{X}}^{(n)} \boxdot_2 \underline{\mathbf{Y}}^{(n)} \in \mathbb{R}^{(R_{n-1}Q_{n-1}) \times (I_n + J_n - 1) \times (R_n Q_n)}, \quad (4.24)$$

or, equivalently, using the standard convolution $\underline{\mathbf{Z}}^{(n)}(s_{n-1}, :, s_n) = \underline{\mathbf{X}}^{(n)}(r_{n-1}, :, r_n) * \underline{\mathbf{Y}}^{(n)}(q_{n-1}, :, q_n) \in \mathbb{R}^{(I_n + J_n - 1)}$ for $s_n = 1, 2, \ldots, R_n Q_n$ and $n = 1, 2, \ldots, N, \ R_0 = R_N = 1$.

Inner product. The computation of the inner (scalar, dot) product of two Nth-order tensors, $\underline{\mathbf{X}} = \langle\!\langle \underline{\mathbf{X}}^{(1)}, \underline{\mathbf{X}}^{(2)}, \ldots, \underline{\mathbf{X}}^{(N)} \rangle\!\rangle \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\mathbf{Y} = \langle\!\langle \underline{\mathbf{Y}}^{(1)}, \underline{\mathbf{Y}}^{(2)}, \ldots, \underline{\mathbf{Y}}^{(N)} \rangle\!\rangle \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, is given by

$$\begin{aligned} \langle \underline{\mathbf{X}}, \underline{\mathbf{Y}} \rangle &= \langle \text{vec}(\underline{\mathbf{X}}), \text{vec}(\underline{\mathbf{Y}}) \rangle \qquad (4.25) \\ &= \sum_{i_1=1}^{I_1} \cdots \sum_{i_N=1}^{I_N} x_{i_1 \ldots i_n} \ y_{i_1 \cdots i_N} \end{aligned}$$

and has the complexity of $\mathcal{O}(I^N)$ in the raw tensor format. In TT formats, the inner product can be computed with the reduced complexity of only $\mathcal{O}(NI(R^2 \tilde{R} + R \tilde{R}^2))$ when the inner product is calculated by moving TT-cores from left to right and performing calculations on relatively small matrices, $\mathbf{S}_n = \underline{\mathbf{X}}^{(n)} \times_{1,2}^{1,2} (\underline{\mathbf{Y}}^{(n)} \times_1 \mathbf{S}_{n-1}) \in \mathbb{R}^{R_n \times \tilde{R}_n}$ for $n = 1, 2, \ldots, N$. The results are then sequentially multiplied by the next core $\underline{\mathbf{Y}}^{(n+1)}$ (see Algorithm 9).

Computation of the Frobenius norm. In a similar way, we can efficiently compute the Frobenius norm of a tensor, $\|\underline{\mathbf{X}}\|_F = \sqrt{\langle \underline{\mathbf{X}}, \underline{\mathbf{X}} \rangle}$, in the TT format. For the so-called n-orthogonal[6] TT format, it is easy to show that

$$\|\underline{\mathbf{X}}\|_F = \|\underline{\mathbf{X}}^{(n)}\|_F. \qquad (4.26)$$

Matrix-by-vector multiplication. Consider a huge-scale matrix equation (see Figure 4.9 and Figure 4.10)

$$\mathbf{A}\mathbf{x} = \mathbf{y}, \qquad (4.27)$$

---

[6]An Nth-order tensor $\underline{\mathbf{X}} = \langle\!\langle \underline{\mathbf{X}}^{(1)}, \underline{\mathbf{X}}^{(2)} \ldots, \underline{\mathbf{X}}^{(N)} \rangle\!\rangle$ in the TT format is called n-orthogonal if all the cores to the left of the core $\underline{\mathbf{X}}^{(n)}$ are left-orthogonalized and all the cores to the right of the core $\mathbf{X}^{(n)}$ are right-orthogonalized (see Part 2 for more detail).

Algorithm 9: **Inner product of two large-scale tensors in the TT Format (Oseledets, 2011; Dolgov, 2014)**

Input: Nth-order tensors, $\underline{\mathbf{X}} = \langle\!\langle \underline{\mathbf{X}}^{(1)}, \underline{\mathbf{X}}^{(2)}, \ldots, \underline{\mathbf{X}}^{(N)} \rangle\!\rangle \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$
and $\underline{\mathbf{Y}} = \langle\!\langle \underline{\mathbf{Y}}^{(1)}, \underline{\mathbf{Y}}^{(2)}, \ldots, \underline{\mathbf{Y}}^{(N)} \rangle\!\rangle \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ in TT formats, with
TT-cores $\underline{\mathbf{X}} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$ and $\underline{\mathbf{Y}} \in \mathbb{R}^{\tilde{R}_{n-1} \times I_n \times \tilde{R}_n}$ and
$R_0 = \tilde{R}_0 = R_N = \tilde{R}_N = 1$

Output: Inner product $\langle \underline{\mathbf{X}}, \underline{\mathbf{Y}} \rangle = \mathrm{vec}(\underline{\mathbf{X}})^{\mathrm{T}} \mathrm{vec}(\underline{\mathbf{Y}})$

1: Initialization $\mathbf{S}_0 = 1$
2: **for** $n = 1$ to N **do**
3:      $\mathbf{Z}^{(n)}_{(1)} = \mathbf{S}_{n-1} \mathbf{Y}^{(n)}_{(1)} \in \mathbb{R}^{R_{n-1} \times I_n \tilde{R}_n}$
4:      $\mathbf{S}_n = \mathbf{X}^{(n)\,\mathrm{T}}_{<2>} \mathbf{Z}^{(n)}_{<2>} \in \mathbb{R}^{R_n \times \tilde{R}_n}$
5: **end for**
6: **return** Scalar $\langle \underline{\mathbf{X}}, \underline{\mathbf{Y}} \rangle = \mathbf{S}_N \in \mathbb{R}^{R_N \times \tilde{R}_N} = \mathbb{R}$, with $R_N = \tilde{R}_N = 1$

---

where $\mathbf{A} \in \mathbb{R}^{I \times J}$, $\mathbf{x} \in \mathbb{R}^J$ and $\mathbf{y} \in \mathbb{R}^I$ are represented approximately in the TT format, with $I = I_1 I_2 \cdots I_N$ and $J = J_1 J_2 \cdots J_N$. As shown in Figure 4.9(a), the cores are defined as $\underline{\mathbf{A}}^{(n)} \in \mathbb{R}^{P_{n-1} \times I_n \times J_n \times P_n}$, $\underline{\mathbf{X}}^{(n)} \in \mathbb{R}^{R_{n-1} \times J_n \times R_n}$ and $\underline{\mathbf{Y}}^{(n)} \in \mathbb{R}^{Q_{n-1} \times I_n \times Q_n}$.

Upon representing the entries of the matrix $\mathbf{A}$ and vectors $\mathbf{x}$ and $\mathbf{y}$ in their tensorized forms, given by

$$
\begin{aligned}
\underline{\mathbf{A}} &= \sum_{p_1, p_2, \ldots, p_{N-1}=1}^{P_1, P_2, \ldots, P_{N-1}} \mathbf{A}^{(1)}_{1, p_1} \circ \mathbf{A}^{(2)}_{p_1, p_2} \circ \cdots \circ \mathbf{A}^{(N)}_{p_{N-1}, 1} \\
\underline{\mathbf{X}} &= \sum_{r_1, r_2, \ldots, r_{N-1}=1}^{R_1, R_2, \ldots, R_{N-1}} \mathbf{x}^{(1)}_{r_1} \circ \mathbf{x}^{(2)}_{r_1, r_2} \circ \cdots \circ \mathbf{x}^{(N)}_{r_{N-1}} \qquad (4.28) \\
\underline{\mathbf{Y}} &= \sum_{q_1, q_2, \ldots, q_{N-1}=1}^{Q_1, Q_2, \ldots, Q_{N-1}} \mathbf{y}^{(1)}_{q_1} \circ \mathbf{y}^{(2)}_{q_1, q_2} \circ \cdots \circ \mathbf{y}^{(N)}_{q_{N-1}},
\end{aligned}
$$

we arrive at a simple formula for the tubes of the tensor $\underline{\mathbf{Y}}$, in the form

$$
\mathbf{y}^{(n)}_{q_{n-1}, q_n} = \mathbf{y}^{(n)}_{\overline{r_{n-1}\,p_{n-1}}, \,\overline{r_n\,p_n}} = \mathbf{A}^{(n)}_{p_{n-1}, p_n} \, \mathbf{x}^{(n)}_{r_{n-1}, r_n} \in \mathbb{R}^{I_n},
$$

with $Q_n = P_n R_n$ for $n = 1, 2, \ldots, N$.

(a)

(b)

Figure   4.9: Linear systems represented by arbitrary tensor networks (*left*) and TT networks (*right*) for (a) $\mathbf{Ax} \cong \mathbf{y}$ and (b) $\mathbf{AX} \cong \mathbf{Y}$.

(a)

(b)
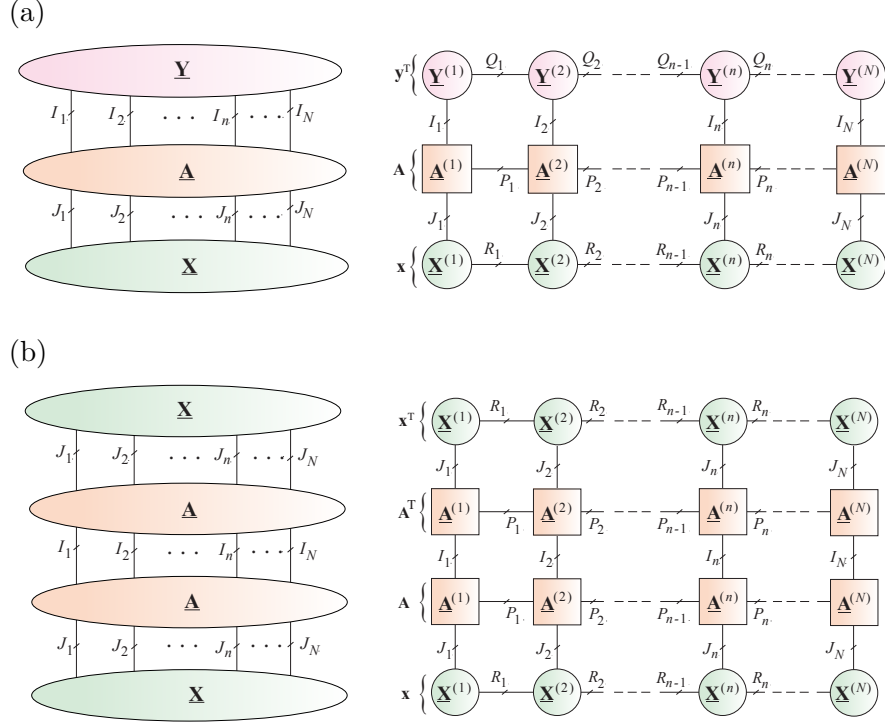
Figure 4.10: Representation of typical cost functions by arbitrary TNs and by TT networks: (a) $J_1(\mathbf{x}) = \mathbf{y}^{\mathrm{T}}\mathbf{A}\mathbf{x}$ and (b) $J_2(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{x}$. Note that tensors $\underline{\mathbf{A}}$, $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ can be, in general, approximated by any TNs that provide good low-rank representations.

Furthermore, by representing the matrix $\mathbf{A}$ and vectors $\mathbf{x}$, $\mathbf{y}$ via the strong Kronecker products

$$
\begin{aligned}
\mathbf{A} &= \tilde{\mathbf{A}}^{(1)} \,|\!\otimes\!|\, \tilde{\mathbf{A}}^{(2)} \,|\!\otimes\!|\, \cdots \,|\!\otimes\!|\, \tilde{\mathbf{A}}^{(N)} \\
\mathbf{x} &= \tilde{\mathbf{X}}^{(1)} \,|\!\otimes\!|\, \tilde{\mathbf{X}}^{(2)} \,|\!\otimes\!|\, \cdots \,|\!\otimes\!|\, \tilde{\mathbf{X}}^{(N)} \\
\mathbf{y} &= \tilde{\mathbf{Y}}^{(1)} \,|\!\otimes\!|\, \tilde{\mathbf{Y}}^{(2)} \,|\!\otimes\!|\, \cdots \,|\!\otimes\!|\, \tilde{\mathbf{Y}}^{(N)},
\end{aligned}
\tag{4.29}
$$

with $\tilde{\mathbf{A}}^{(n)} \in \mathbb{R}^{P_{n-1}I_n \times J_n P_n}$, $\tilde{\mathbf{X}}^{(n)} \in \mathbb{R}^{R_{n-1}J_n \times R_n}$ and $\tilde{\mathbf{Y}}^{(n)} \in \mathbb{R}^{Q_{n-1}I_n \times Q_n}$, we can establish a simple relationship
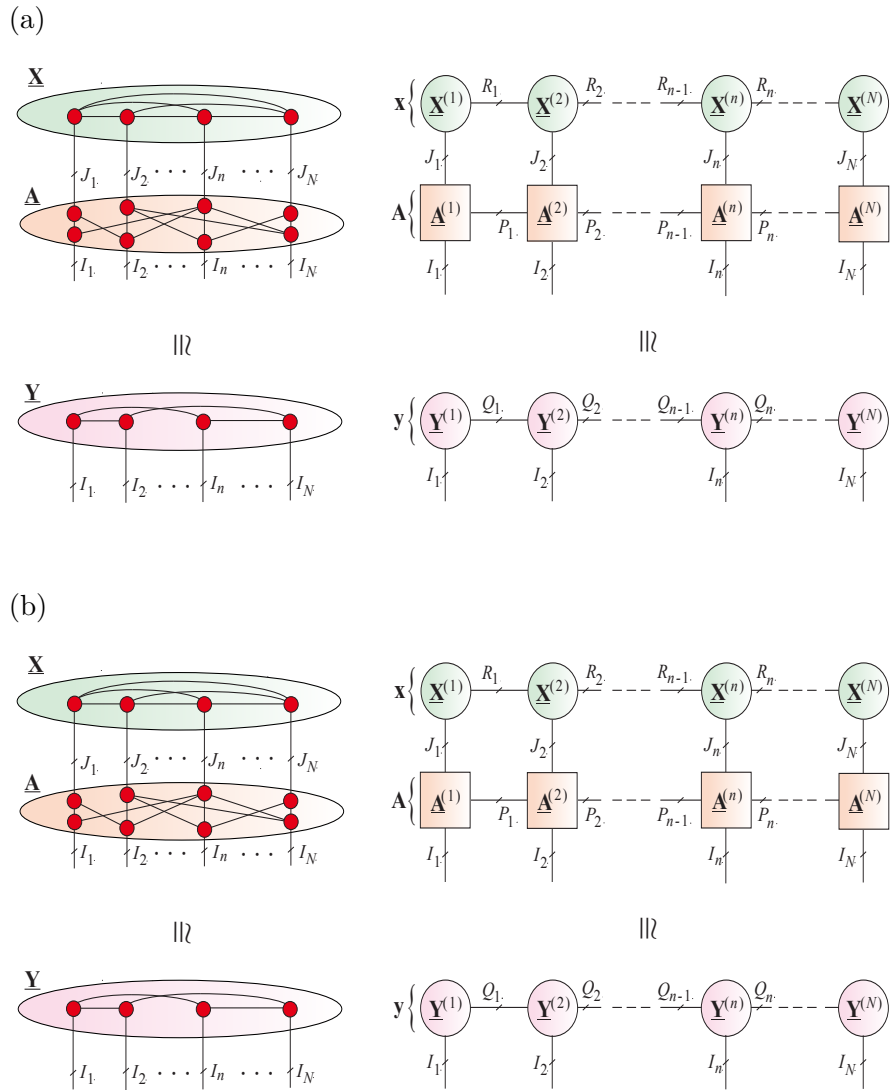
$$
\tilde{\mathbf{Y}}^{(n)} = \tilde{\mathbf{A}}^{(n)} \,|\!\bullet\!|\, \tilde{\mathbf{X}}^{(n)} \in \mathbb{R}^{R_{n-1} P_{n-1} I_n \times R_n P_n}, \quad n = 1, \dots, N, \tag{4.30}
$$
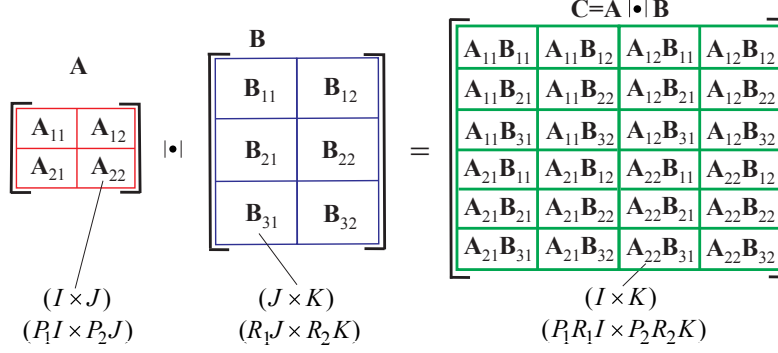
Figure 4.11: Graphical illustration of the C product of two block matrices.

where the operator $|\bullet|$ represents the C (Core) product of two block matrices.

The C product of a block matrix $\mathbf{A}^{(n)} \in \mathbb{R}^{P_{n-1}I_n \times P_n J_n}$ with blocks $\mathbf{A}^{(n)}_{p_{n-1},p_n} \in \mathbb{R}^{I_n \times J_n}$, and a block matrix $\mathbf{B}^{(n)} \in \mathbb{R}^{R_{n-1}J_n \times R_n K_n}$, with blocks $\mathbf{B}^{(n)}_{r_{n-1},r_n} \in \mathbb{R}^{J_n \times K_n}$, is defined as $\mathbf{C}^{(n)} = \mathbf{A}^{(n)} \; |\bullet| \; \mathbf{B}^{(n)} \in \mathbb{R}^{Q_{n-1}I_n \times Q_n K_n}$, the blocks of which are given by $\mathbf{C}^{(n)}_{q_{n-1},q_n} = \mathbf{A}^{(n)}_{p_{n-1},p_n}\mathbf{B}^{(n)}_{r_{n-1},r_n} \in \mathbb{R}^{I_n \times K_n}$, where $q_n = \overline{p_n r_n}$, as illustrated in Figure 4.11.

Note that, equivalently to Eq. (4.30), for $\mathbf{A}\mathbf{x} = \mathbf{y}$, we can use a slice representation, given by

$$\mathbf{Y}^{(n)}_{i_n} = \sum_{j_n=1}^{J_n} (\mathbf{A}^{(n)}_{i_n,j_n} \otimes_L \mathbf{X}^{(n)}_{j_n}), \tag{4.31}$$

which can be implemented by fast matrix-by matrix multiplication algorithms (see Algorithm 10). In practice, for very large scale data, we usually perform TT core contractions (MPO-MPS product) approximately, with reduced TT ranks, e.g., via the "zip-up" method proposed by (Stoudenmire and White, 2010).

In a similar way, the matrix equation

$$\mathbf{Y} \cong \mathbf{A}\mathbf{X}, \tag{4.32}$$

where $\mathbf{A} \in \mathbb{R}^{I \times J}$, $\mathbf{X} \in \mathbb{R}^{J \times K}$, $\mathbf{Y} \in \mathbb{R}^{I \times K}$, with $I = I_1 I_2 \cdots I_N$, $J = J_1 J_2 \cdots J_N$ and $K = K_1 K_2 \cdots K_N$, can be represented in TT

Table 4.5: Basic operations on tensors in TT formats, where $\underline{\mathbf{X}} = \underline{\mathbf{X}}^{(1)} \times^1 \underline{\mathbf{X}}^{(2)} \times^1 \cdots \times^1 \underline{\mathbf{X}}^{(N)} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, $\underline{\mathbf{Y}} = \underline{\mathbf{Y}}^{(1)} \times^1 \underline{\mathbf{Y}}^{(2)} \times^1 \cdots \times^1 \underline{\mathbf{Y}}^{(N)} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$, and $\underline{\mathbf{Z}} = \underline{\mathbf{Z}}^{(1)} \times^1 \underline{\mathbf{Z}}^{(2)} \times^1 \cdots \times^1 \underline{\mathbf{Z}}^{(N)} \in \mathbb{R}^{K_1 \times K_2 \times \cdots \times K_N}$.

Operation	TT-cores

$$\underline{\mathbf{Z}} = \underline{\mathbf{X}} + \underline{\mathbf{Y}} = \left(\underline{\mathbf{X}}^{(1)} \boxplus_2 \underline{\mathbf{Y}}^{(1)} \right) \times^1 \left(\underline{\mathbf{X}}^{(2)} \boxplus_2 \underline{\mathbf{Y}}^{(2)} \right) \times^1 \cdots \times^1 \left(\underline{\mathbf{X}}^{(N)} \boxplus_2 \underline{\mathbf{Y}}^{(N)} \right)$$

$\underline{\mathbf{Z}}^{(n)} = \underline{\mathbf{X}}^{(n)} \boxplus_2 \underline{\mathbf{Y}}^{(n)}$, with TT core slices $\mathbf{Z}_{i_n}^{(n)} = \mathbf{X}_{i_n}^{(n)} \oplus \mathbf{Y}_{i_n}^{(n)}$, $(I_n = J_n = K_n, \ \forall n)$

$$\underline{\mathbf{Z}} = \underline{\mathbf{X}} \oplus \underline{\mathbf{Y}} = \left(\underline{\mathbf{X}}^{(1)} \oplus \underline{\mathbf{Y}}^{(1)} \right) \times^1 \left(\underline{\mathbf{X}}^{(2)} \oplus \underline{\mathbf{Y}}^{(2)} \right) \times^1 \cdots \times^1 \left(\underline{\mathbf{X}}^{(N)} \oplus \underline{\mathbf{Y}}^{(N)} \right)$$

$$\underline{\mathbf{Z}} = \underline{\mathbf{X}} \circledast \underline{\mathbf{Y}} = \left(\underline{\mathbf{X}}^{(1)} \odot_2 \underline{\mathbf{Y}}^{(1)} \right) \times^1 \left(\underline{\mathbf{X}}^{(2)} \odot_2 \underline{\mathbf{Y}}^{(2)} \right) \times^1 \cdots \times^1 \left(\underline{\mathbf{X}}^{(N)} \odot_2 \underline{\mathbf{Y}}^{(N)} \right)$$

$\underline{\mathbf{Z}}^{(n)} = \underline{\mathbf{X}}^{(n)} \odot_2 \underline{\mathbf{Y}}^{(n)}$, with TT core slices $\mathbf{Z}_{i_n}^{(n)} = \mathbf{X}_{i_n}^{(n)} \otimes \mathbf{Y}_{i_n}^{(n)}$, $(I_n = J_n = K_n, \ \forall n)$

$$\underline{\mathbf{Z}} = \underline{\mathbf{X}} \otimes \underline{\mathbf{Y}} = \left(\underline{\mathbf{X}}^{(1)} \otimes \underline{\mathbf{Y}}^{(1)} \right) \times^1 \left(\underline{\mathbf{X}}^{(2)} \otimes \underline{\mathbf{Y}}^{(2)} \right) \times^1 \cdots \times^1 \left(\underline{\mathbf{X}}^{(N)} \otimes \underline{\mathbf{Y}}^{(N)} \right)$$

$\underline{\mathbf{Z}}^{(n)} = \underline{\mathbf{X}}^{(n)} \otimes \underline{\mathbf{Y}}^{(n)}$, with TT core slices $\mathbf{Z}_{k_n}^{(n)} = \mathbf{X}_{i_n}^{(n)} \otimes \mathbf{Y}_{j_n}^{(n)}$ $(k_n = \overline{i_n j_n})$

$$\underline{\mathbf{Z}} = \underline{\mathbf{X}} * \underline{\mathbf{Y}} = (\underline{\mathbf{X}}^{(1)} \boxdot_2 \underline{\mathbf{Y}}^{(1)}) \times^1 \cdots \times^1 (\underline{\mathbf{X}}^{(N)} \boxdot_2 \underline{\mathbf{Y}}^{(N)})$$

$\underline{\mathbf{Z}}^{(n)} = \underline{\mathbf{X}}^{(n)} \boxdot_2 \underline{\mathbf{Y}}^{(n)} \in \mathbb{R}^{(R_{n-1}Q_{n-1}) \times (I_n + J_n - 1) \times (R_n Q_n)}$, with vectors

$\underline{\mathbf{Z}}^{(n)}(s_{n-1}, :, s_n) = \underline{\mathbf{X}}^{(n)}(r_{n-1}, :, r_n) * \underline{\mathbf{Y}}^{(n)}(q_{n-1}, :, q_n) \in \mathbb{R}^{(I_n + J_n - 1)}$

for $s_n = 1, 2, \ldots, R_n Q_n$ and $n = 1, 2, \ldots, N, \ R_0 = R_N = 1.$

$$\underline{\mathbf{Z}} = \underline{\mathbf{X}} \times_n \mathbf{A} = \underline{\mathbf{X}}^{(1)} \times^1 \cdots \times^1 \underline{\mathbf{X}}^{(n-1)} \times^1 \left(\underline{\mathbf{X}}^{(n)} \times_2 \mathbf{A} \right) \times^1 \underline{\mathbf{X}}^{(n+1)} \times^1 \cdots \times^1 \underline{\mathbf{X}}^{(N)}$$

$z = \langle \underline{\mathbf{X}}, \underline{\mathbf{Y}} \rangle = \mathbf{Z}^{(1)} \times^1 \mathbf{Z}^{(2)} \times^1 \cdots \times^1 \mathbf{Z}^{(N)} = \mathbf{Z}^{(1)} \mathbf{Z}^{(2)} \cdots \mathbf{Z}^{(N)}$

$\mathbf{Z}^{(n)} = \left(\underline{\mathbf{X}}^{(n)} \odot_2 \underline{\mathbf{Y}}^{(n)} \right) \overline{\times}_2 \mathbf{1}_{I_n} = \sum_{i_n} \mathbf{X}_{i_n}^{(n)} \otimes \mathbf{Y}_{i_n}^{(n)}$ $(I_n = J_n, \ \forall n)$

Table 4.6: Basic operations in the TT format expressed via the strong Kronecker and C products of block matrices, where $\mathbf{A} = \widetilde{\mathbf{A}}^{(1)} \,|\!\otimes\!|\, \widetilde{\mathbf{A}}^{(2)} \,|\!\otimes\!| \cdots |\!\otimes\!|\, \widetilde{\mathbf{A}}^{(N)}$, $\mathbf{B} = \widetilde{\mathbf{B}}^{(1)} \,|\!\otimes\!|\, \widetilde{\mathbf{B}}^{(2)} \,|\!\otimes\!| \cdots |\!\otimes\!|\, \widetilde{\mathbf{B}}^{(N)}$, $\mathbf{x} = \widetilde{\mathbf{X}}^{(1)} \,|\!\otimes\!|\, \widetilde{\mathbf{X}}^{(2)} \,|\!\otimes\!| \cdots |\!\otimes\!|\, \widetilde{\mathbf{X}}^{(N)}$, $\mathbf{y} = \widetilde{\mathbf{Y}}^{(1)} \,|\!\otimes\!|\, \widetilde{\mathbf{Y}}^{(2)} \,|\!\otimes\!| \cdots |\!\otimes\!|\, \widetilde{\mathbf{Y}}^{(N)}$ and the block matrices $\widetilde{\mathbf{A}}^{(n)} \in \mathbb{R}^{R^A_{n-1} I_n \times J_n R^A_n}$, $\widetilde{\mathbf{B}}^{(n)} \in \mathbb{R}^{R^B_{n-1} J_n \times K_n R^B_n}$, $\widetilde{\mathbf{X}}^{(n)} \in \mathbb{R}^{R^x_{n-1} I_n \times R^x_n}$, $\widetilde{\mathbf{Y}}^{(n)} \in \mathbb{R}^{R^y_{n-1} I_n \times R^y_n}$.

Operation	Block matrices of TT-cores

$\mathbf{Z} = \mathbf{A} + \mathbf{B}$

$$= \begin{bmatrix} \widetilde{\mathbf{A}}^{(1)} & \widetilde{\mathbf{B}}^{(1)} \end{bmatrix} |\!\otimes\!| \begin{bmatrix} \widetilde{\mathbf{A}}^{(2)} & \mathbf{0} \\ \mathbf{0} & \widetilde{\mathbf{B}}^{(2)} \end{bmatrix} |\!\otimes\!| \cdots |\!\otimes\!| \begin{bmatrix} \widetilde{\mathbf{A}}^{(N-1)} & \mathbf{0} \\ \mathbf{0} & \widetilde{\mathbf{B}}^{(N-1)} \end{bmatrix} |\!\otimes\!| \begin{bmatrix} \widetilde{\mathbf{A}}^{(N)} \\ \widetilde{\mathbf{B}}^{(N)} \end{bmatrix}$$

$$\mathbf{Z} = \mathbf{A} \otimes \mathbf{B} = \widetilde{\mathbf{A}}^{(1)} |\!\otimes\!| \cdots |\!\otimes\!| \widetilde{\mathbf{A}}^{(N)} |\!\otimes\!| \widetilde{\mathbf{B}}^{(1)} |\!\otimes\!| \cdots |\!\otimes\!| \widetilde{\mathbf{B}}^{(N)}$$

$$z = \mathbf{x}^{\mathrm{T}} \mathbf{y} = \langle \mathbf{x}, \mathbf{y} \rangle = \left(\widetilde{\mathbf{X}}^{(1)} \,|\!\bullet\!|\, \widetilde{\mathbf{Y}}^{(1)} \right) |\!\otimes\!| \cdots |\!\otimes\!| \left(\widetilde{\mathbf{X}}^{(N)} \,|\!\bullet\!|\, \widetilde{\mathbf{Y}}^{(N)} \right)$$

$$\widetilde{\mathbf{Z}}^{(n)} = \widetilde{\mathbf{X}}^{(n)} \,|\!\bullet\!|\, \widetilde{\mathbf{Y}}^{(n)} \in \mathbb{R}^{R^x_{n-1} R^y_{n-1} \times R^x_n R^y_n}, \text{ with core slices } \mathbf{Z}^{(n)} = \sum_{i_n} \mathbf{X}^{(n)}_{i_n} \otimes \mathbf{Y}^{(n)}_{i_n}$$

$$\mathbf{z} = \mathbf{A}\mathbf{x} = \left(\widetilde{\mathbf{A}}^{(1)} \,|\!\bullet\!|\, \widetilde{\mathbf{X}}^{(1)} \right) |\!\otimes\!| \cdots |\!\otimes\!| \left(\widetilde{\mathbf{A}}^{(N)} \,|\!\bullet\!|\, \widetilde{\mathbf{X}}^{(N)} \right)$$

$$\widetilde{\mathbf{Z}}^{(n)} = \widetilde{\mathbf{A}}^{(n)} \times^1 \widetilde{\mathbf{X}}^{(n)}, \text{ with blocks (vectors)}$$

$$\mathbf{z}^{(n)}_{s_{n-1}, s_n} = \mathbf{A}^{(n)}_{r^A_{n-1}, r^A_n} \mathbf{x}^{(n)}_{r^x_{n-1}, r^x_n} \qquad \left(s_n = \overline{r^A_n \, r^x_n} \right)$$

$$\mathbf{Z} = \mathbf{A}\mathbf{B} = \left(\widetilde{\mathbf{A}}^{(1)} \,|\!\bullet\!|\, \widetilde{\mathbf{B}}^{(1)} \right) |\!\otimes\!| \cdots |\!\otimes\!| \left(\widetilde{\mathbf{A}}^{(N)} \,|\!\bullet\!|\, \widetilde{\mathbf{B}}^{(N)} \right)$$

$$\widetilde{\mathbf{Z}}^{(n)} = \widetilde{\mathbf{A}}^{(n)} \,|\!\bullet\!|\, \widetilde{\mathbf{B}}^{(n)}, \text{ with blocks}$$

$$\mathbf{Z}^{(n)}_{s_{n-1}, s_n} = \mathbf{A}^{(n)}_{r^A_{n-1}, r^A_n} \mathbf{B}^{(n)}_{r^B_{n-1}, r^B_n} \qquad \left(s_n = \overline{r^A_n \, r^B_n} \right)$$

$$z = \mathbf{x}^{\mathrm{T}} \mathbf{A}\mathbf{x} = \langle \mathbf{x}, \mathbf{A}\mathbf{x} \rangle$$
$$= \left(\widetilde{\mathbf{X}}^{(1)} \,|\!\bullet\!|\, \widetilde{\mathbf{A}}^{(1)} \,|\!\bullet\!|\, \widetilde{\mathbf{X}}^{(1)} \right) |\!\otimes\!| \cdots |\!\otimes\!| \left(\widetilde{\mathbf{X}}^{(N)} \,|\!\bullet\!|\, \widetilde{\mathbf{A}}^{(N)} \,|\!\bullet\!|\, \widetilde{\mathbf{X}}^{(N)} \right)$$

$$\widetilde{\mathbf{Z}}^{(n)} = \widetilde{\mathbf{X}}^{(n)} \,|\!\bullet\!|\, \widetilde{\mathbf{A}}^{(n)} \,|\!\bullet\!|\, \widetilde{\mathbf{X}}^{(n)} \in \mathbb{R}^{R^x_{n-1} R^A_{n-1} R^x_{n-1} \times R^x_n R^A_n R^x_n}, \text{ with blocks (entries)}$$

$$z^{(n)}_{s_{n-1}, s_n} = \left\langle \mathbf{x}^{(n)}_{r^x_{n-1}, r^x_n}, \mathbf{A}^{(n)}_{r^A_{n-1}, r^A_n} \mathbf{x}^{(n)}_{r^y_{n-1}, r^y_n} \right\rangle \qquad \left(s_n = \overline{r^x_n \, r^A_n \, r^y_n} \right)$$

Algorithm 10: Computation of a Matrix-by-Vector Product in the TT Format

---

Input: Matrix $\mathbf{A} \in \mathbb{R}^{I \times J}$ and vector $\mathbf{x} \in \mathbb{R}^{J}$ in their respective
   TT format
   $\mathbf{A} = \langle\!\langle \underline{\mathbf{A}}^{(1)}, \underline{\mathbf{A}}^{(2)}, \ldots, \underline{\mathbf{A}}^{(N)} \rangle\!\rangle \in \mathbb{R}^{I_1 \times J_1 \times I_2 \times J_2 \times \cdots \times I_N \times J_N}$,
   and $\underline{\mathbf{X}} = \langle\!\langle \underline{\mathbf{X}}^{(1)}, \underline{\mathbf{X}}^{(2)}, \ldots, \underline{\mathbf{X}}^{(N)} \rangle\!\rangle \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$,
   with TT-cores
   $\underline{\mathbf{X}}^{(n)} \in \mathbb{R}^{R_{n-1} \times J_n \times R_n}$ and $\underline{\mathbf{A}}^{(n)} \in \mathbb{R}^{R_{n-1}^A \times I_n \times I_n \times R_n^A}$

Output: Matrix by vector product $\mathbf{y} = \mathbf{A}\mathbf{x}$ in the TT format
   $\mathbf{Y} = \langle\!\langle \underline{\mathbf{Y}}^{(1)}, \underline{\mathbf{Y}}^{(2)}, \ldots, \underline{\mathbf{Y}}^{(N)} \rangle\!\rangle \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, with cores
   $\underline{\mathbf{Y}}^{(n)} \in \mathbb{R}^{R_{n-1}^Y \times J_n \times R_n^Y}$

1: **for** $n = 1$ to N **do**
2:    **for** $i_n = 1$ to I_n **do**
3:        $\mathbf{Y}_{i_n}^{(n)} = \sum_{j_n=1}^{J_n} \left(\mathbf{A}_{i_n,j_n}^{(n)} \otimes_L \mathbf{X}_{j_n}^{(n)} \right)$
4:    **end for**
5: **end for**
6: **return** Vector $\mathbf{y} \in \mathbb{R}^{I_1 I_2 \cdots I_N}$ in the TT format
   $\underline{\mathbf{Y}} = \langle\!\langle \underline{\mathbf{Y}}^{(1)}, \underline{\mathbf{Y}}^{(2)}, \ldots, \underline{\mathbf{Y}}^{(N)} \rangle\!\rangle$

---

formats. This is illustrated in Figure 4.9(b) for the corresponding TT-cores defined as

$$\underline{\mathbf{A}}^{(n)} \in \mathbb{R}^{P_{n-1} \times I_n \times J_n \times P_n}$$
$$\underline{\mathbf{X}}^{(n)} \in \mathbb{R}^{R_{n-1} \times J_n \times K_n \times R_n}$$
$$\underline{\mathbf{Y}}^{(n)} \in \mathbb{R}^{Q_{n-1} \times I_n \times K_n \times Q_n}.$$

It is straightforward to show that when the matrices, $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{X} \in \mathbb{R}^{J \times K}$, are represented in their TT formats, they can be expressed via a strong Kronecker product of block matrices as $\mathbf{A} = \tilde{\mathbf{A}}^{(1)} |\otimes| \tilde{\mathbf{A}}^{(2)} |\otimes| \cdots |\otimes| \tilde{\mathbf{A}}^{(N)}$ and $\mathbf{X} = \tilde{\mathbf{X}}^{(1)} |\otimes| \tilde{\mathbf{X}}^{(2)} |\otimes| \cdots |\otimes| \tilde{\mathbf{X}}^{(N)}$, where the factor matrices are $\tilde{\mathbf{A}}^{(n)} \in \mathbb{R}^{P_{n-1} I_n \times J_n P_n}$ and $\tilde{\mathbf{X}}^{(n)} \in \mathbb{R}^{R_{n-1} J_n \times K_n R_n}$. Then, the matrix $\mathbf{Y} = \mathbf{A}\mathbf{X}$ can also be expressed via the strong Kronecker products, $\mathbf{Y} = \tilde{\mathbf{Y}}^{(1)} |\otimes| \cdots |\otimes| \tilde{\mathbf{Y}}^{(N)}$, where $\tilde{\mathbf{Y}}^{(n)} = \tilde{\mathbf{A}}^{(n)} |\bullet| \tilde{\mathbf{X}}^{(n)} \in \mathbb{R}^{Q_{n-1} I_n \times K_n Q_n}$, $(n = 1, 2, \ldots, N)$, with blocks $\tilde{\mathbf{Y}}_{q_{n-1}, q_n}^{(n)} = \tilde{\mathbf{A}}_{p_{n-1}, p_n}^{(n)} \tilde{\mathbf{X}}_{r_{n-1}, r_n}^{(n)}$, where $Q_n = R_n P_n$, $q_n = \overline{p_n r_n}$, $\forall n$.

Similarly, a quadratic form, $z = \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}$, for a huge symmetric matrix $\mathbf{A}$, can be computed by first computing (in TT formats), a vector $\mathbf{y} = \mathbf{A}\mathbf{x}$, followed by the inner product $\mathbf{x}^{\mathrm{T}}\mathbf{y}$.

Basic operations in the TT format are summarized in Table 4.5, while Table 4.6 presents these operations expressed via strong Kronecker and C products of block matrices of TT-cores. For more advanced and sophisticated operations in TT/QTT formats, see (Kazeev *et al.*, 2013a,b; Lee and Cichocki, 2016c).

4.6 Algorithms for TT Decompositions

We have shown that a major advantage of the TT decomposition is the existence of efficient algorithms for an exact representation of higher-order tensors and/or their low-rank approximate representations with a prescribed accuracy. Similarly to the quasi-best approximation property of the HOSVD, the TT approximation $\widehat{\underline{\mathbf{X}}} = \langle\!\langle \widehat{\underline{\mathbf{X}}}^{(1)}, \widehat{\underline{\mathbf{X}}}^{(2)}, \ldots, \widehat{\underline{\mathbf{X}}}^{(N)} \rangle\!\rangle \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ (with core tensors denoted by $\mathbf{X}^{(n)} = \underline{\mathbf{G}}^{(n)}$), obtained by the TT-SVD algorithm, satisfies the following inequality

$$\|\underline{\mathbf{X}} - \widehat{\underline{\mathbf{X}}}\|_2^2 \leqslant \sum_{n=1}^{N-1} \sum_{j=R_{n+1}}^{I_n} \sigma_j^2(\mathbf{X}_{<n>}), \qquad (4.33)$$

where the ℓ_2-norm of a tensor is defined via its vectorization and $\sigma_j(\mathbf{X}_{<n>})$ denotes the jth largest singular value of the unfolding matrix $\mathbf{X}_{<n>}$ (Oseledets, 2011).

The two basic approaches to perform efficiently TT decompositions are based on: (1) low-rank matrix factorizations (LRMF), and (2) constrained Tucker-2 decompositions.

4.6.1 Sequential SVD/LRMF Algorithms

The most important algorithm for the TT decomposition is the TT-SVD algorithm (see Algorithm 11) (Vidal, 2003; Oseledets and Tyrtyshnikov, 2009), which applies the truncated SVD sequentially to the unfolding matrices, as illustrated in Figure 4.12. Instead of SVD, alternative and efficient LRMF algorithms can be used (Corona *et al.*, 2015),
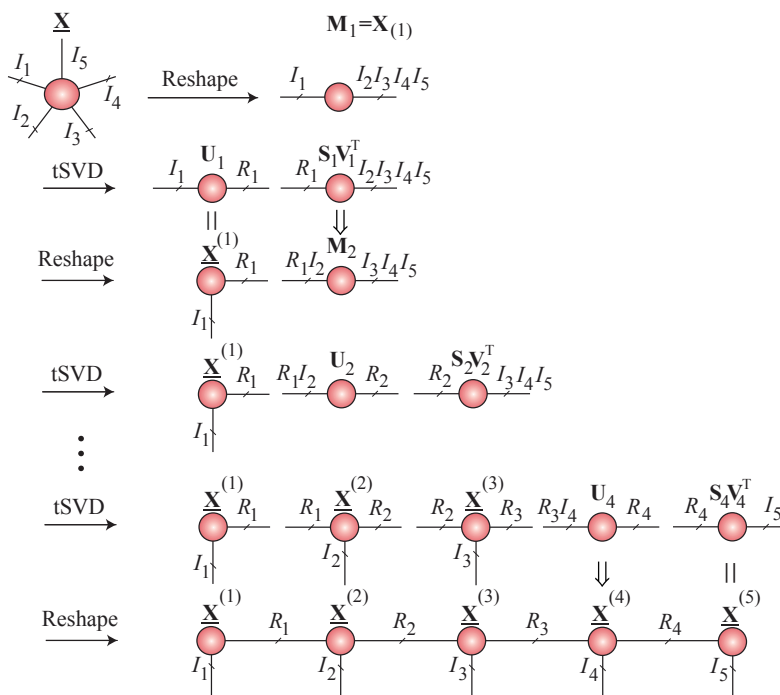
Figure 4.12: The TT-SVD algorithm for a 5th-order data tensor using truncated SVD. Instead of the SVD, any alternative LRMF algorithm can be employed, such as randomized SVD, RPCA, CUR/CA, NMF, SCA, ICA. Top panel: A 6th-order tensor $\underline{\mathbf{X}}$ of size $I_1 \times I_2 \times \cdots \times I_5$ is first reshaped into a long matrix $\mathbf{M}_1$ of size $I_1 \times I_2 \cdots I_5$. Second panel: The tSVD is performed to produce low-rank matrix factorization, with $I_1 \times R_1$ factor matrix $\mathbf{U}_1$ and the $R_1 \times I_2 \cdots I_5$ matrix $\mathbf{S}_1\mathbf{V}_1^{\mathrm{T}}$, so that $\mathbf{M}_1 \cong \mathbf{U}_1\mathbf{S}_1\mathbf{V}_1^{\mathrm{T}}$. Third panel: the matrix $\mathbf{U}_1$ becomes the first core core $\underline{\mathbf{X}}^{(1)} \in \mathbb{R}^{1 \times I_1 \times R_1}$, while the matrix $\mathbf{S}_1\mathbf{V}_1^{\mathrm{T}}$ is reshaped into the $R_1 I_2 \times I_3 I_4 I_5$ matrix $\mathbf{M}_2$. Remaining panels: Perform tSVD to yield $\mathbf{M}_2 \cong \mathbf{U}_2\mathbf{S}_2\mathbf{V}_2^{\mathrm{T}}$, reshape $\mathbf{U}_2$ into an $R_1 \times I_2 \times R_2$ core $\underline{\mathbf{X}}^{(2)}$ and repeat the procedure until all the five cores are extracted (bottom panel). The same procedure applies to higher order tensors of any order.

see also Algorithm 12). For example, in (Oseledets and Tyrtyshnikov, 2010) a new approximate formula for TT decomposition is proposed, where an Nth-order data tensor $\underline{\mathbf{X}}$ is interpolated using a special form of cross-approximation. In fact, the TT-Cross-Approximation is analogous to the TT-SVD algorithm, but uses adaptive cross-approximation instead of the computationally more expensive SVD. The complexity

---

Algorithm 11: **TT-SVD Decomposition using truncated SVD (tSVD) or randomized SVD (rSVD) (Vidal, 2003; Oseledets, 2011)**

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 the TT format
    $\underline{\widehat{\mathbf{X}}} = \langle\!\langle \underline{\widehat{\mathbf{X}}}^{(1)}, \underline{\widehat{\mathbf{X}}}^{(2)}, \ldots, \underline{\widehat{\mathbf{X}}}^{(N)} \rangle\!\rangle$, such that $\|\underline{\mathbf{X}} - \underline{\widehat{\mathbf{X}}}\|_F \leqslant \varepsilon$
 1: Unfolding of tensor $\underline{\mathbf{X}}$ in mode-1 $\mathbf{M}_1 = \mathbf{X}_{(1)}$
 2: Initialization $R_0 = 1$
 3: **for** $n = 1$ to $N - 1$ **do**
 4:    Perform tSVD $[\mathbf{U}_n, \mathbf{S}_n, \mathbf{V}_n] = \text{tSVD}(\mathbf{M}_n, \varepsilon/\sqrt{N-1})$
 5:    Estimate nth TT rank $R_n = \text{size}(\mathbf{U}_n, 2)$
 6:    Reshape orthogonal matrix $\mathbf{U}_n$ into a 3rd-order core
      $\underline{\widehat{\mathbf{X}}}^{(n)} = \text{reshape}(\mathbf{U}_n, [R_{n-1}, I_n, R_n])$
 7:    Reshape the matrix $\mathbf{V}_n$ into a matrix
      $\mathbf{M}_{n+1} = \text{reshape}\left(\mathbf{S}_n \mathbf{V}_n^{\mathrm{T}}, [R_n I_{n+1}, \prod_{p=n+2}^{N} I_p]\right)$
 8: **end for**
 9: Construct the last core as $\underline{\widehat{\mathbf{X}}}^{(N)} = \text{reshape}(\mathbf{M}_N, [R_{N-1}, I_N, 1])$
10: **return** $\langle\!\langle \underline{\widehat{\mathbf{X}}}^{(1)}, \underline{\widehat{\mathbf{X}}}^{(2)}, \ldots, \underline{\widehat{\mathbf{X}}}^{(N)} \rangle\!\rangle$.

---

of the cross-approximation algorithms scales linearly with the order N of a data tensor.

4.6.2 Tucker-2/PVD Algorithms for Large-scale TT Decompositions

The key idea in this approach is to reshape any Nth-order data tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with $N > 3$, into a suitable 3rd-order tensor, e.g., $\underline{\widetilde{\mathbf{X}}} \in \mathbb{R}^{I_1 \times I_N \times I_2 \cdots I_{N-1}}$, in order to apply the Tucker-2 decomposition as follows (see Algorithm 8 and Figure 4.13(a))

$$\underline{\widetilde{\mathbf{X}}} = \underline{\mathbf{G}}^{(2,N-1)} \times_1 \mathbf{X}^{(1)} \times_2 \mathbf{X}^{(N)} = \mathbf{X}^{(1)} \times^1 \underline{\mathbf{G}}^{(2,N-1)} \times^1 \mathbf{X}^{(N)}, \quad (4.34)$$

which, by using frontal slices of the involved tensors, can also be expressed in the matrix form

$$\mathbf{X}_{k_1} = \mathbf{X}^{(1)} \mathbf{G}_{k_1} \mathbf{X}^{(N)}, \quad k_1 = 1, 2, \ldots, I_2 \cdots I_{N-1}. \quad (4.35)$$

Such representations allow us to compute the tensor, $\underline{\mathbf{G}}^{(2,N-1)}$, the first TT-core, $\mathbf{X}^{(1)}$, and the last TT-core, $\mathbf{X}^{(N)}$. The procedure can

---

Algorithm 12: **TT Decomposition using any efficient LRMF**

Input: Tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and the approximation accuracy ε
Output: Approximate tensor representation in the TT format
$\quad \underline{\widehat{\mathbf{X}}} \cong \langle\!\langle \underline{\widehat{\mathbf{X}}}^{(1)}, \underline{\widehat{\mathbf{X}}}^{(2)}, \ldots, \underline{\widehat{\mathbf{X}}}^{(N)} \rangle\!\rangle$
1: Initialization $R_0 = 1$
2: Unfolding of tensor $\underline{\mathbf{X}}$ in mode-1 as $\mathbf{M}_1 = \mathbf{X}_{(1)}$
3: **for** $n = 1$ to $N - 1$ **do**
4: Perform LRMF, e.g., CUR, RPCA, ...
   $[\mathbf{A}_n, \mathbf{B}_n] = \mathrm{LRMF}(\mathbf{M}_n, \varepsilon)$, i.e., $\mathbf{M}_n \cong \mathbf{A}_n \mathbf{B}_n^{\mathrm{T}}$
5: Estimate nth TT rank, $R_n = \mathrm{size}(\mathbf{A}_n, 2)$
6: Reshape matrix $\mathbf{A}_n$ into a 3rd-order core, as
   $\underline{\widehat{\mathbf{X}}}^{(n)} = \mathrm{reshape}\,(\mathbf{A}_n, [R_{n-1}, I_n, R_n])$
7: Reshape the matrix $\mathbf{B}_n$ into the $(n+1)$th unfolding matrix
   $\mathbf{M}_{n+1} = \mathrm{reshape}\left(\mathbf{B}_n^{\mathrm{T}}, [R_n I_{n+1}, \prod_{p=n+2}^{N} I_p]\right)$
8: **end for**
9: Construct the last core as $\underline{\widehat{\mathbf{X}}}^{(N)} = \mathrm{reshape}(\mathbf{M}_N, [R_{N-1}, I_N, 1])$
10: **return** TT-cores: $\langle\!\langle \underline{\widehat{\mathbf{X}}}^{(1)}, \underline{\widehat{\mathbf{X}}}^{(2)}, \ldots, \underline{\widehat{\mathbf{X}}}^{(N)} \rangle\!\rangle$.

---

be repeated sequentially for reshaped tensors $\underline{\widetilde{\mathbf{G}}}_n = \underline{\mathbf{G}}^{(n+1, N-n)}$ for $n = 1, 2, \ldots$, in order to extract subsequent TT-cores in their matricized forms, as illustrated in Figure 4.13(b). See also the detailed step-by-step procedure shown in Figure 4.13(c).

Such a simple recursive procedure for TT decomposition can be used in conjunction with any efficient algorithm for Tucker-2/PVD decompositions or the nonnegative Tucker-2 decomposition (NTD-2) (see also Section 3).

4.6.3   Tensor Train Rounding – TT Recompression

Mathematical operations in TT format produce core tensors with ranks which are not guaranteed to be optimal with respect to the desired approximation accuracy. For example, matrix-by-vector or matrix-by-matrix products considerably increase the TT ranks, which quickly become computationally prohibitive, so that a truncation or low-rank TT approximations are necessary for mathematical tractability. To this end, the TT–rounding (also called truncation or recompression) may be used as a post-processing procedure to reduce the TT ranks. The TT

(a)

(b)

(c)
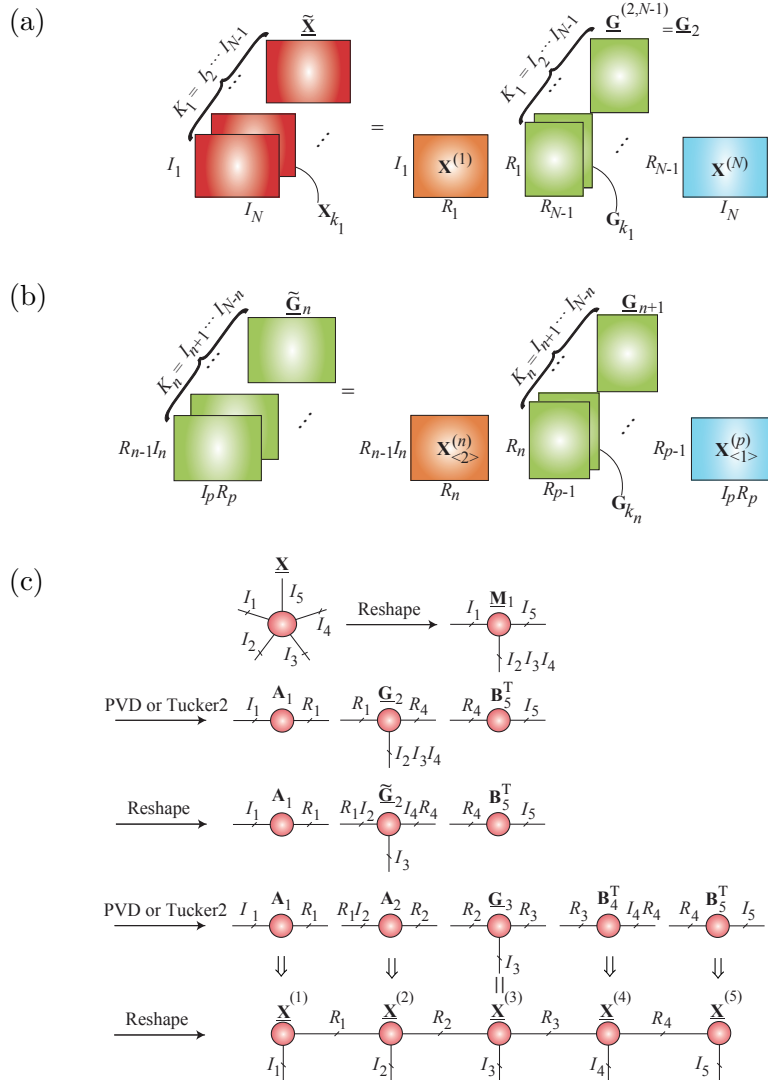
Figure 4.13: TT decomposition based on the Tucker-2/PVD model. (a) Extraction of the first and the last core. (b) The procedure can be repeated sequentially for reshaped 3rd-order tensors $\underline{\mathbf{G}}_n$ (for $n = 2, 3, \ldots$ and $p = N-1, N-2, \ldots$). (c) Illustration of a TT decomposition for a 5th-order data tensor, using an algorithm based on sequential Tucker-2/PVD decompositions.

Algorithm 13: **TT Rounding (Recompression) (Oseledets, 2011)**

Input: Nth-order tensor $\underline{\mathbf{X}} = \langle\!\langle \underline{\mathbf{X}}^{(1)}, \underline{\mathbf{X}}^{(2)}, \ldots, \underline{\mathbf{X}}^{(N)} \rangle\!\rangle \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$,
   in a TT format with an overestimated TT rank,
   $\mathbf{r}_{TT} = \{R_1, R_2, \ldots, R_{N-1}\}$, and TT-cores $\underline{\mathbf{X}} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$,
   absolute tolerance ε, and maximum rank $R_{\max}$

Output: Nth-order tensor $\underline{\hat{\mathbf{X}}}$ with a reduced TT rank; the cores are
   rounded (reduced) according to the input tolerance ε and/or ranks
   bounded by $R_{\max}$, such that $\|\underline{\mathbf{X}} - \underline{\hat{\mathbf{X}}}\|_F \leqslant \varepsilon \, \|\underline{\mathbf{X}}\|_F$

  1: Initialization $\underline{\hat{\mathbf{X}}} = \underline{\mathbf{X}}$ and $\delta = \varepsilon/\sqrt{N-1}$

  2: **for** $n = 1$ to $N - 1$ **do**

  3:      QR decomposition $\mathbf{X}_{<2>}^{(n)} = \mathbf{Q}_n \mathbf{R}$, with $\mathbf{X}_{<2>}^{(n)} \in \mathbb{R}^{R_{n-1} I_n \times R_n}$

  4:      Replace cores $\mathbf{X}_{<2>}^{(n)} = \mathbf{Q}_n$ and $\mathbf{X}_{<1>}^{(n+1)} \leftarrow \mathbf{R}\mathbf{X}_{<1>}^{(n+1)}$, with
   $\mathbf{X}_{<1>}^{(n+1)} \in \mathbb{R}^{R_n \times I_{n+1} R_{n+1}}$

  5: **end for**

  6: **for** $n = N$ to 2 **do**

  7:      Perform δ-truncated SVD $\mathbf{X}_{<1>}^{(n)} = \mathbf{U} \operatorname{diag}\{\boldsymbol{\sigma}\} \mathbf{V}^{\mathrm{T}}$

  8:      Determine minimum rank $\hat{R}_{n-1}$ such that $\sum_{r > R_{n-1}} \sigma_r^2 \leqslant \delta^2 \|\boldsymbol{\sigma}\|^2$

  9:      Replace cores $\hat{\mathbf{X}}_{<2>}^{(n-1)} \leftarrow \hat{\mathbf{X}}_{<2>}^{(n-1)} \hat{\mathbf{U}} \operatorname{diag}\{\hat{\boldsymbol{\sigma}}\}$ and $\hat{\mathbf{X}}_{<1>}^{(n)} = \hat{\mathbf{V}}^{\mathrm{T}}$

10: **end for**

11: **return** Nth-order tensor
   $\hat{\underline{\mathbf{X}}} = \langle\!\langle \underline{\hat{\mathbf{X}}}^{(1)}, \underline{\hat{\mathbf{X}}}^{(2)}, \ldots, \underline{\hat{\mathbf{X}}}^{(N)} \rangle\!\rangle \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$,
   with reduced cores $\underline{\hat{\mathbf{X}}}^{(n)} \in \mathbb{R}^{\hat{R}_{n-1} \times I_n \times \hat{R}_n}$

---

rounding algorithms are typically implemented via QR/SVD with the aim to approximate, with a desired prescribed accuracy, the TT core tensors, $\underline{\mathbf{G}}^{(n)} = \underline{\mathbf{X}}^{(n)}$, by other core tensors with minimum possible TT-ranks (see Algorithm 13). Note that TT rounding is mathematically the same as the TT-SVD, but is more efficient owing to the to use of TT format.

The complexity of TT-rounding procedures is only $\mathcal{O}(NIR^3)$, since all operations are performed in TT format which requires the SVD to be computed only for a relatively small matricized core tensor at each iteration. A similar approach has been developed for the HT format (Grasedyck, 2010; Grasedyck *et al.*, 2013; Kressner and Tobler, 2014; Espig *et al.*, 2012).

4.6.4 Orthogonalization of Tensor Train Network

The orthogonalization of core tensors is an essential procedure in many algorithms for the TT formats (Oseledets, 2011; Holtz *et al.*, 2012; Dolgov *et al.*, 2014; Dolgov, 2014; Kressner *et al.*, 2014a; Steinlechner, 2015, 2016).

For convenience, we divide a TT network, which represents a tensor $\underline{\widehat{\mathbf{X}}} = \langle\!\langle \underline{\widehat{\mathbf{X}}}^{(1)}, \underline{\widehat{\mathbf{X}}}^{(2)}, \ldots, \underline{\widehat{\mathbf{X}}}^{(N)} \rangle\!\rangle \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, into sub-trains. In this way, a large-scale task is replaced by easier-to-handle sub-tasks, whereby the aim is to extract a specific TT core or its slices from the whole TT network. For this purpose, the TT sub-trains can be defined as follows

$$\underline{\widehat{\mathbf{X}}}^{<n} = \langle\!\langle \underline{\widehat{\mathbf{X}}}^{(1)}, \underline{\widehat{\mathbf{X}}}^{(2)}, \ldots, \underline{\widehat{\mathbf{X}}}^{(n-1)} \rangle\!\rangle \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{n-1} \times R_{n-1}} \quad (4.36)$$

$$\underline{\widehat{\mathbf{X}}}^{>n} = \langle\!\langle \underline{\widehat{\mathbf{X}}}^{(n+1)}, \underline{\widehat{\mathbf{X}}}^{(n+2)}, \ldots, \underline{\widehat{\mathbf{X}}}^{(N)} \rangle\!\rangle \in \mathbb{R}^{R_n \times I_{n+1} \times \cdots \times I_N} \quad (4.37)$$

while the corresponding unfolding matrices, also called interface matrices, are defined by

$$\widehat{\mathbf{X}}^{\leqslant n} \in \mathbb{R}^{I_1 I_2 \cdots I_n \times R_n}, \qquad \widehat{\mathbf{X}}^{>n} \in \mathbb{R}^{R_n \times I_{n+1} \cdots I_N}. \quad (4.38)$$

The left and right unfolding of the cores are defined as

$$\widehat{\mathbf{X}}_L^{(n)} = \widehat{\mathbf{X}}_{<2>}^{(n)} \in \mathbb{R}^{R_{n-1} I_n \times R_n} \;\; \text{and} \;\; \widehat{\mathbf{X}}_R^{(n)} = \mathbf{X}_{<1>}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n R_n} \;.$$

The n-orthogonality of tensors. An Nth-order tensor in a TT format $\underline{\widehat{\mathbf{X}}} = \langle\!\langle \underline{\widehat{\mathbf{X}}}^{(1)}, \ldots, \underline{\widehat{\mathbf{X}}}^{(N)} \rangle\!\rangle$, is called n-orthogonal with $1 \leqslant n \leqslant N$, if

$$(\widehat{\mathbf{X}}_L^{(m)})^{\mathrm{T}} \widehat{\mathbf{X}}_L^{(m)} = \mathbf{I}_{R_m}, \;\; m = 1, \ldots, n-1 \quad (4.39)$$

$$\widehat{\mathbf{X}}_R^{(m)} (\widehat{\mathbf{X}}_R^{(m)})^{\mathrm{T}} = \mathbf{I}_{R_{m-1}}, \;\; m = n+1, \ldots, N. \quad (4.40)$$

The tensor is called left-orthogonal if $n = N$ and right-orthogonal if $n = 1$.

When considering the nth TT core, it is usually assumed that all cores to the left are left-orthogonalized, and all cores to the

---

Algorithm 14: **Left-orthogonalization, right-orthogonalization and n-orthogonalization of a tensor in the TT format**

Input: Nth-order tensor $\hat{\underline{\mathbf{X}}} = \langle\!\langle \hat{\underline{\mathbf{X}}}^{(1)}, \hat{\underline{\mathbf{X}}}^{(2)}, \ldots, \hat{\underline{\mathbf{X}}}^{(N)} \rangle\!\rangle \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$,
with TT cores $\hat{\underline{\mathbf{X}}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$ and $R_0 = R_N = 1$

Output: Cores $\hat{\underline{\mathbf{X}}}^{(1)}, \ldots, \hat{\underline{\mathbf{X}}}^{(n-1)}$ become left-orthogonal, while the
remaining cores are right-orthogonal, except for the core $\hat{\underline{\mathbf{X}}}^{(n)}$

1: **for** $m = 1$ to $n - 1$ **do**
2:    Perform the QR decomposition $[\mathbf{Q}, \mathbf{R}] \leftarrow qr(\hat{\mathbf{X}}_L^{(m)})$ for the
       unfolding cores $\hat{\mathbf{X}}_L^{(m)} \in \mathbb{R}^{R_{m-1}I_m \times R_m}$
3:    Replace the cores $\hat{\mathbf{X}}_L^{(m)} \leftarrow \mathbf{Q}$ and $\hat{\underline{\mathbf{X}}}^{(m+1)} \leftarrow \hat{\underline{\mathbf{X}}}^{(m+1)} \times_1 \mathbf{R}$
4: **end for**
5: **for** $m = N$ to $n + 1$ **do**
6:    Perform QR decomposition $[\mathbf{Q}, \mathbf{R}] \leftarrow qr((\hat{\mathbf{X}}_R^{(m)})^{\mathrm{T}})$ for the
       unfolding cores $(\hat{\mathbf{X}}_R^{(m)}) \in \mathbb{R}^{R_{m-1} \times I_m R_m}$,
7:    Replace the cores: $\mathbf{G}_R^{(m)} \leftarrow \mathbf{Q}^{\mathrm{T}}$ and $\hat{\underline{\mathbf{X}}}^{(m-1)} \leftarrow \hat{\underline{\mathbf{X}}}^{(m-1)} \times_3 \mathbf{R}^{\mathrm{T}}$
8: **end for**
9: **return** Left-orthogonal TT cores with $(\hat{\mathbf{X}}_L^{(m)})^{\mathrm{T}} \hat{\mathbf{X}}_L^{(m)} = \mathbf{I}_{R_m}$ for
    $m = 1, 2, \ldots, n - 1$ and right-orthogonal cores $\hat{\mathbf{X}}_R^{(m)} (\hat{\mathbf{X}}_R^{(m)})^{\mathrm{T}} = \mathbf{I}_{R_{m-1}}$
    for $m = N, N - 1, \ldots, n + 1$.

---

right are right-orthogonalized. Notice that if a TT tensor[7], $\hat{\underline{\mathbf{X}}}$, is n-orthogonal then the "left" and "right" interface matrices have orthonormal columns and rows, that is

$$(\hat{\mathbf{X}}^{<n})^{\mathrm{T}} \, \hat{\mathbf{X}}^{<n} = \mathbf{I}_{R_{n-1}}, \qquad \hat{\mathbf{X}}^{>n} \, (\hat{\mathbf{X}}^{>n})^{\mathrm{T}} = \mathbf{I}_{R_n}. \qquad (4.41)$$

A tensor in a TT format can be orthogonalized efficiently using recursive QR and LQ decompositions (see Algorithm 14). From the above definition, for $n = N$ the algorithms perform left-orthogonalization and for $n = 1$ right-orthogonalization of the whole TT network.

---

[7]By a TT-tensor we refer to as a tensor represented in the TT format.

4.6.5 Improved TT Decomposition Algorithm – Alternating Single Core Update (ASCU)

Finally, we next present an efficient algorithm for TT decomposition, referred to as the Alternating Single Core Update (ASCU), which sequentially optimizes a single TT-core tensor while keeping the other TT-cores fixed in a manner similar to the modified ALS (Phan *et al.*, 2016).

Assume that the TT-tensor $\widehat{\underline{\mathbf{X}}} = \langle\!\langle \widehat{\underline{\mathbf{X}}}^{(1)}, \widehat{\underline{\mathbf{X}}}^{(2)}, \ldots, \widehat{\underline{\mathbf{X}}}^{(N)} \rangle\!\rangle$ is left- and right-orthogonalized up to $\widehat{\underline{\mathbf{X}}}^{(n)}$, i.e., the unfolding matrices $\widehat{\underline{\mathbf{X}}}_{<2>}^{(k)}$ for $k = 1, \ldots, n - 1$ have orthonormal columns, and $\widehat{\underline{\mathbf{X}}}_{(1)}^{(m)}$ for $m = n + 1, \ldots, N$ have orthonormal rows. Then, the Frobenius norm of the TT-tensor $\widehat{\underline{\mathbf{X}}}$ is equivalent to the Frobenius norm of $\widehat{\underline{\mathbf{X}}}^{(n)}$, that is, $\|\widehat{\underline{\mathbf{X}}}\|_F^2 = \|\widehat{\underline{\mathbf{X}}}^{(n)}\|_F^2$, so that the Frobenius norm of the approximation error between a data tensor $\underline{\mathbf{X}}$ and its approximate representation in the TT format $\widehat{\underline{\mathbf{X}}}$ can be written as

$$
\begin{aligned}
J(\underline{\mathbf{X}}^{(n)}) &= \|\underline{\mathbf{X}} - \widehat{\underline{\mathbf{X}}}\|_F^2 \qquad\qquad\qquad\qquad\qquad (4.42)\\
&= \|\underline{\mathbf{X}}\|_F^2 + \|\widehat{\underline{\mathbf{X}}}\|_F^2 - 2\langle\underline{\mathbf{X}}, \ \widehat{\underline{\mathbf{X}}}\rangle\\
&= \|\underline{\mathbf{X}}\|_F^2 + \|\widehat{\underline{\mathbf{X}}}^{(n)}\|_F^2 - 2\langle\underline{\mathbf{C}}^{(n)}, \ \widehat{\underline{\mathbf{X}}}^{(n)}\rangle\\
&= \|\underline{\mathbf{X}}\|_F^2 - \|\underline{\mathbf{C}}^{(n)}\|_F^2 + \|\underline{\mathbf{C}}^{(n)} - \widehat{\underline{\mathbf{X}}}^{(n)}\|_F^2, \qquad n = 1, \ldots, N,
\end{aligned}
$$

where $\underline{\mathbf{C}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$ represents a tensor contraction of $\underline{\mathbf{X}}$ and $\widehat{\underline{\mathbf{X}}}$ along all modes but the mode-n, as illustrated in Figure 4.14. The $\underline{\mathbf{C}}^{(n)}$ can be efficiently computed through left contractions along the first $(n - 1)$-modes and right contractions along the last $(N - m)$-modes, expressed as

$$
\underline{\mathbf{L}}^{<n} = \widehat{\underline{\mathbf{X}}}^{<n} \ltimes_{n-1} \underline{\mathbf{X}}, \qquad \underline{\mathbf{C}}^{(n)} = \underline{\mathbf{L}}^{<n} \rtimes_{N-n} \widehat{\underline{\mathbf{X}}}^{>n}. \qquad (4.43)
$$

The symbols $\ltimes_n$ and $\rtimes_m$ stand for the tensor contractions between two Nth-order tensors along their first n modes and last $m = N - n$ modes, respectively.

The optimization problem in (4.42) is usually performed subject to the following constraint

$$
\|\underline{\mathbf{X}} - \widehat{\underline{\mathbf{X}}}\|_F^2 \leqslant \varepsilon^2 \qquad\qquad (4.44)
$$

such that the TT-rank of $\widehat{\underline{\mathbf{X}}}$ is minimum.

Observe that the constraint in (4.44) for left- and right-orthogonalized TT-cores is equivalent to the set of sub-constraints

$$\|\underline{\mathbf{C}}^{(n)} - \widehat{\underline{\mathbf{X}}}^{(n)}\|_F^2 \leqslant \varepsilon_n^2 \qquad n = 1, \ldots, N, \tag{4.45}$$

whereby the nth core $\underline{\mathbf{X}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$ should have minimum ranks R_{n-1} and R_n. Furthermore, $\varepsilon_n^2 = \varepsilon^2 - \|\underline{\mathbf{X}}\|_F^2 + \|\underline{\mathbf{C}}^{(n)}\|_F^2$ is assumed to be non-negative. Finally, we can formulate the following sequential optimization problem

$$\begin{aligned} &\min \ (R_{n-1} \cdot R_n), \\ &\text{s.t.} \quad \|\underline{\mathbf{C}}^{(n)} - \widehat{\underline{\mathbf{X}}}^{(n)}\|_F^2 \leqslant \varepsilon_n^2, \quad n = 1, 2, \ldots, N. \end{aligned} \tag{4.46}$$

By expressing the TT-core tensor $\widehat{\underline{\mathbf{X}}}^{(n)}$ as a TT-tensor of three factors, i.e., in a Tucker-2 format given by

$$\widehat{\underline{\mathbf{X}}}^{(n)} = \mathbf{A}_n \times^1 \tilde{\underline{\mathbf{X}}}^{(n)} \times^1 \mathbf{B}_n \,,$$

the above optimization problem with the constraint (4.45) reduces to performing a Tucker-2 decomposition (see Algorithm 8). The aim is to compute $\mathbf{A}_n$, $\mathbf{B}_n$ (orthogonal factor matrices) and a core tensor $\tilde{\underline{\mathbf{X}}}^{(n)}$ which approximates tensor $\underline{\mathbf{C}}^{(n)}$ with a minimum TT-rank-$(\tilde{R}_{n-1}, \tilde{R}_n)$, such that

$$\|\underline{\mathbf{C}}^{(n)} - \mathbf{A}_n \times^1 \tilde{\underline{\mathbf{X}}}^{(n)} \times^1 \mathbf{B}_n\|_F^2 \leqslant \varepsilon_n^2 \,,$$

where $\mathbf{A}_n \in \mathbb{R}^{R_{n-1} \times \tilde{R}_{n-1}}$ and $\mathbf{B}_n \in \mathbb{R}^{\tilde{R}_n \times R_n}$, with $\tilde{R}_{n-1} \leftarrow R_{n-1}$ and $\tilde{R}_n \leftarrow R_n$.

Note that the new estimate of $\underline{\mathbf{X}}$ is still of Nth-order because the factor matrices $\mathbf{A}_n$ and $\mathbf{B}_n$ can be embedded into $\widehat{\underline{\mathbf{X}}}^{(n-1)}$ and $\widehat{\underline{\mathbf{X}}}^{(n+1)}$ as follows

$$\begin{aligned} \widehat{\underline{\mathbf{X}}} &= \widehat{\underline{\mathbf{X}}}^{(1)} \times^1 \cdots \times^1 (\widehat{\underline{\mathbf{X}}}^{(n-1)} \times^1 \mathbf{A}_n) \times^1 \tilde{\underline{\mathbf{X}}}^{(n)} \times^1 (\mathbf{B}_n \times^1 \widehat{\underline{\mathbf{X}}}^{(n+1)}) \\ &\quad \times^1 \cdots \times^1 \widehat{\underline{\mathbf{X}}}^{(N)} \,. \end{aligned}$$

In this way, the three TT-cores $\widehat{\underline{\mathbf{X}}}^{(n-1)}$, $\widehat{\underline{\mathbf{X}}}^{(n)}$ and $\widehat{\underline{\mathbf{X}}}^{(n+1)}$ are updated. Since $\mathbf{A}_n$ and $\mathbf{B}_n^{\mathrm{T}}$ have respectively orthonormal columns and rows,
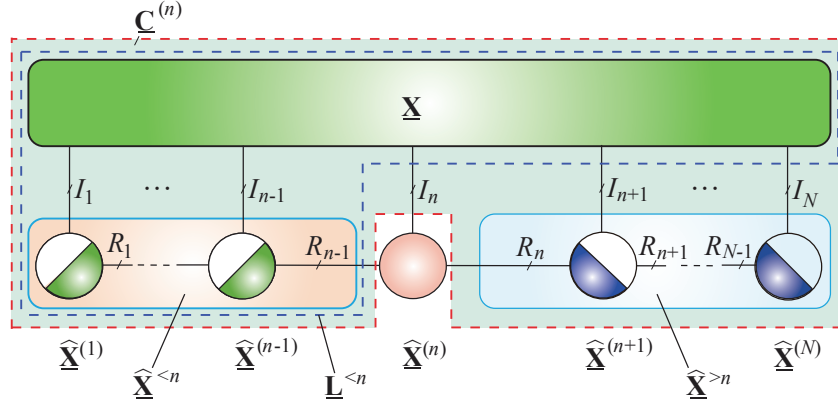
Figure 4.14: Illustration of the contraction of tensors in the Alternating Single Core Update (ASCU) algorithm (see Algorithm 15). All the cores to the left of $\underline{\mathbf{X}}^{(n)}$ are left-orthogonal and all cores to its right are right-orthogonal.

the newly adjusted cores $(\widehat{\underline{\mathbf{X}}}^{(n-1)} \times^1 \mathbf{A}_n)$ and $(\mathbf{B}_n \times^1 \widehat{\underline{\mathbf{X}}}^{(n+1)})$ obey the left- and right-orthogonality conditions. Algorithm 15 outlines such a single-core update algorithm based on the Tucker-2 decomposition. In the pseudo-code, the left contracted tensor $\underline{\mathbf{L}}^{<n}$ is computed efficiently through a progressive contraction in the form (Schollwöck, 2011; Hubig *et al.*, 2015)

$$\underline{\mathbf{L}}^{<n} = \widehat{\underline{\mathbf{X}}}^{(n-1)} \ltimes_2 \underline{\mathbf{L}}^{<(n-1)}, \tag{4.47}$$

where $\underline{\mathbf{L}}^{<1} = \underline{\mathbf{X}}$.

Alternatively, instead of adjusting the two TT ranks, R_{n-1} and R_n, of $\widehat{\underline{\mathbf{X}}}^{(n)}$, we can update only one rank, either R_{n-1} or R_n, corresponding to the right-to-left or left-to-right update order procedure. Assuming that the core tensors are updated in the left-to-right order, we need to find $\widehat{\underline{\mathbf{X}}}^{(n)}$ which has a minimum rank-R_n and satisfies the constraints

$$\|\underline{\mathbf{C}}^{(n)} - \widehat{\underline{\mathbf{X}}}^{(n)} \times^1 \mathbf{B}_n\|_F^2 \leqslant \varepsilon_n^2, \qquad n = 1, \dots, N.$$

This problem reduces to the truncated SVD of the mode-$\{1, 2\}$ matricization of $\underline{\mathbf{C}}^{(n)}$ with an accuracy ε_n^2, that is

$$[\underline{\mathbf{C}}^{(n)}]_{<2>} \approx \mathbf{U}_n \, \mathbf{\Sigma} \, \mathbf{V}_n^{\mathrm{T}},$$

Algorithm 15: **The Alternating Single-Core Update Algorithm (two-sides rank adjustment) (Phan *et al.*, 2016)**

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

Output: TT-tensor $\widehat{\underline{\mathbf{X}}} = \widehat{\underline{\mathbf{X}}}^{(1)} \times^1 \widehat{\underline{\mathbf{X}}}^{(2)} \times^1 \cdots \times^1 \widehat{\underline{\mathbf{X}}}^{(N)}$ of minimum TT-rank such that $\|\underline{\mathbf{X}} - \widehat{\underline{\mathbf{X}}}\|_F^2 \leqslant \varepsilon^2$

1: Initialize $\widehat{\underline{\mathbf{X}}} = \langle\!\langle \widehat{\underline{\mathbf{X}}}^{(1)}, \widehat{\underline{\mathbf{X}}}^{(2)}, \ldots, \widehat{\underline{\mathbf{X}}}^{(N)} \rangle\!\rangle$

2: **repeat**

3:     **for** $n = 1, 2, \ldots, N-1$ **do**

4:         Compute contracted tensor $\underline{\mathbf{C}}^{(n)} = \underline{\mathbf{L}}^{<n} \bowtie_{N-n} \widehat{\underline{\mathbf{X}}}^{>n}$

5:         Solve a Tucker-2 decomposition

        $\|\underline{\mathbf{C}}^{(n)} - \mathbf{A}_n \times^1 \widehat{\underline{\mathbf{X}}}^{(n)} \times^1 \mathbf{B}_n\|_F^2 \leqslant \varepsilon^2 - \|\underline{\mathbf{X}}\|_F^2 + \|\underline{\mathbf{C}}^{(n)}\|_F^2$

6:         Adjust adjacent cores

        $\widehat{\underline{\mathbf{X}}}^{(n-1)} \leftarrow \widehat{\underline{\mathbf{X}}}^{(n-1)} \times^1 \mathbf{A}_n, \qquad \widehat{\underline{\mathbf{X}}}^{(n+1)} \leftarrow \mathbf{B}_n \times^1 \widehat{\underline{\mathbf{X}}}^{(n+1)}$

7:         Perform left-orthogonalization of $\widehat{\underline{\mathbf{X}}}^{(n)}$

8:         Update left-side contracted tensors

        $\underline{\mathbf{L}}^{<n} \leftarrow \mathbf{A}_n^{\mathrm{T}} \times^1 \underline{\mathbf{L}}^{<n}, \qquad \underline{\mathbf{L}}^{<(n+1)} \leftarrow \widehat{\underline{\mathbf{X}}}^{(n)} \ltimes_2 \underline{\mathbf{L}}^{<n}$

9:     **end for**

10:    **for** $n = N, N-1, \ldots, 2$ **do**

11:        Compute contracted tensor $\underline{\mathbf{C}}^{(n)} = \underline{\mathbf{L}}^{<n} \bowtie_{N-n} \widehat{\underline{\mathbf{X}}}^{>n}$

12:        Solve a constrained Tucker-2 decomposition

        $\|\underline{\mathbf{C}}^{(n)} - \mathbf{A}_n \times^1 \widehat{\underline{\mathbf{X}}}^{(n)} \times^1 \mathbf{B}_n\|_F^2 \leqslant \varepsilon^2 - \|\underline{\mathbf{X}}\|_F^2 + \|\underline{\mathbf{C}}^{(n)}\|_F^2$

13:        $\widehat{\underline{\mathbf{X}}}^{(n-1)} \leftarrow \widehat{\underline{\mathbf{X}}}^{(n-1)} \times^1 \mathbf{A}_n, \qquad \widehat{\underline{\mathbf{X}}}^{(n+1)} \leftarrow \mathbf{B}_n \times^1 \widehat{\underline{\mathbf{X}}}^{(n+1)}$

14:        Perform right-orthogonalization of $\widehat{\underline{\mathbf{X}}}^{(n)}$

15:    **end for**

16: **until** a stopping criterion is met

17: **return** $\langle\!\langle \widehat{\underline{\mathbf{X}}}^{(1)}, \widehat{\underline{\mathbf{X}}}^{(2)}, \ldots, \widehat{\underline{\mathbf{X}}}^{(N)} \rangle\!\rangle$.

---

where $\mathbf{\Sigma} = \mathrm{diag}(\sigma_{n,1}, \ldots, \sigma_{n,R_n^\star})$. Here, for the new optimized rank $R_n^\star$, the following holds

$$\sum_{r=1}^{R_n^\star} \sigma_{n,r}^2 \geqslant \|\underline{\mathbf{X}}\|_F^2 - \varepsilon^2 > \sum_{r=1}^{R_n^\star - 1} \sigma_{n,r}^2 . \qquad (4.48)$$

The core tensor $\widehat{\underline{\mathbf{X}}}^{(n)}$ is then updated by reshaping $\mathbf{U}_n$ to an order-3 tensor of size $R_{n-1} \times I_n \times R_n^\star$, while the core $\widehat{\underline{\mathbf{X}}}^{(n+1)}$ needs to be adjusted

---

Algorithm 16: The Alternating Single-Core Update Algorithm (one-side rank adjustment) (Phan *et al.*, 2016)

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

Output: TT-tensor $\widehat{\underline{\mathbf{X}}} = \widehat{\underline{\mathbf{X}}}^{(1)} \times^1 \widehat{\underline{\mathbf{X}}}^{(2)} \times^1 \cdots \times^1 \widehat{\underline{\mathbf{X}}}^{(N)}$ of minimum TT-rank such that $\|\underline{\mathbf{X}} - \widehat{\underline{\mathbf{X}}}\|_F^2 \leqslant \varepsilon^2$

1: Initialize TT-cores $\widehat{\underline{\mathbf{X}}}^{(n)}, \quad \forall n$
2: **repeat**
3:     **for** $n = 1, 2, \ldots, N-1$ **do**
4:         Compute the contracted tensor $\underline{\mathbf{C}}^{(n)} = \underline{\mathbf{L}}^{<n} \bowtie_{N-n} \widehat{\underline{\mathbf{X}}}^{>n}$
5:         Truncated SVD:
        $\|[\underline{\mathbf{C}}^{(n)}]_{<2>} - \mathbf{U}\,\boldsymbol{\Sigma}\,\mathbf{V}^{\mathrm{T}}\|_F^2 \leqslant \varepsilon^2 - \|\underline{\mathbf{X}}\|_F^2 + \|\underline{\mathbf{C}}^{(n)}\|_F^2$
6:         Update $\widehat{\underline{\mathbf{X}}}^{(n)} = \mathrm{reshape}(\mathbf{U}, R_{n-1} \times I_n \times R_n)$
7:         Adjust adjacent core $\widehat{\underline{\mathbf{X}}}^{(n+1)} \leftarrow (\boldsymbol{\Sigma}\,\mathbf{V}^{\mathrm{T}}) \times^1 \widehat{\underline{\mathbf{X}}}^{(n+1)}$
8:         Update left-side contracted tensors
        $\underline{\mathbf{L}}^{<(n+1)} \leftarrow \widehat{\underline{\mathbf{X}}}^{(n)} \bowtie_2 \underline{\mathbf{L}}^{<n}$
9:     **end for**
10:    **for** $n = N, N-1, \ldots, 2$ **do**
11:        Compute contracted tensor $\underline{\mathbf{C}}^{(n)} = \underline{\mathbf{L}}^{<n} \bowtie_{N-n} \widehat{\underline{\mathbf{X}}}^{>n}$
12:        Truncated SVD:
        $\|[\underline{\mathbf{C}}^{(n)}]_{(1)} - \mathbf{U}\,\boldsymbol{\Sigma}\,\mathbf{V}^{\mathrm{T}}\|_F^2 \leqslant \varepsilon^2 - \|\underline{\mathbf{X}}\|_F^2 + \|\underline{\mathbf{C}}^{(n)}\|_F^2;$
13:        $\widehat{\underline{\mathbf{X}}}^{(n)} = \mathrm{reshape}(\mathbf{V}^{\mathrm{T}}, R_{n-1} \times I_n \times R_n)$
14:        $\widehat{\underline{\mathbf{X}}}^{(n-1)} \leftarrow \widehat{\underline{\mathbf{X}}}^{(n-1)} \times^1 (\mathbf{U}\,\boldsymbol{\Sigma})$
15:     **end for**
16: **until** a stopping criterion is met
17: **return** $\langle\!\langle \widehat{\underline{\mathbf{X}}}^{(1)}, \widehat{\underline{\mathbf{X}}}^{(2)}, \ldots, \widehat{\underline{\mathbf{X}}}^{(N)} \rangle\!\rangle.$

---

accordingly as

$$\widehat{\underline{\mathbf{X}}}^{(n+1)\star} = \boldsymbol{\Sigma}\,\mathbf{V}_n^{\mathrm{T}} \times^1 \widehat{\underline{\mathbf{X}}}^{(n+1)}. \tag{4.49}$$

When the algorithm updates the core tensors in the right-to-left order, we update $\widehat{\underline{\mathbf{X}}}^{(n)}$ by using the $R_{n-1}^\star$ leading right singular vectors of the mode-1 matricization of $\underline{\mathbf{C}}^{(n)}$, and adjust the core $\widehat{\underline{\mathbf{X}}}^{(n-1)}$ accordingly,

that is,

$$
\begin{aligned}
[\underline{\mathbf{C}}^{(n)}]_{(1)} &\cong \mathbf{U}_n \, \boldsymbol{\Sigma} \, \mathbf{V}_n^{\mathrm{T}} \\
\widehat{\underline{\mathbf{X}}}^{(n)\star} &= \mathrm{reshape}(\mathbf{V}_n^{\mathrm{T}}, [R_{n-1}^{\star}, I_n, R_n]) \\
\widehat{\underline{\mathbf{X}}}^{(n-1)\star} &= \widehat{\underline{\mathbf{X}}}^{(n-1)} \times^1 (\mathbf{U}_n \, \boldsymbol{\Sigma}) .
\end{aligned}
\tag{4.50}
$$

To summarise, the ASCU method performs a sequential update of one core and adjusts (or rotates) another core. Hence, it updates two cores at a time (for detail see Algorithm 16).

The ASCU algorithm can be implemented in an even more efficient way, if the data tensor $\underline{\mathbf{X}}$ is already given in a TT format (with a non-optimal TT ranks for the prescribed accuracy). Detailed MATLAB implementations and other variants of the TT decomposition algorithm are provided in (Phan *et al.*, 2016).

5

Discussion and Conclusions

In Part 1 of this monograph, we have provided a systematic and example-rich guide to the basic properties and applications of tensor network 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 intrinsic compression ability that stems from the distributed way in which they represent data and process information, TNs can be naturally employed for 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 strong and weak couplings among multiple variables, and to cater for multi-modal, incomplete and noisy data.

In Part 2 of this monograph we introduce a scalable framework for distributed implementation of optimization algorithms, in order to transform 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 standard numerical algorithm to be applied to such local optimization sub-problems.

Although research on tensor networks for dimensionality reduction and optimization problems is only emerging, 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 a variety of machine learning problems for data of otherwise prohibitively large volume, variety, or veracity. We also hope that the readers will find the approaches presented in this monograph helpful in advancing seamlessly from numerical linear algebra to numerical multilinear algebra.

Acknowledgements

References

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

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.

A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15:2773–2832, 2014.

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 method. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, pages 19–27, 2015.

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

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

M. Bachmayr, R. Schneider, and A. Uschmajew. Tensor networks and hierarchical tensors for the solution of high-dimensional partial differential equations. *Foundations of Computational Mathematics*, pages 1–50, 2016.

B. W. Bader and T. G. Kolda. MATLAB tensor toolbox version 2.6, February 2015. URL http://csmr.ca.sandia.gov/~tgkolda/TensorToolbox/.

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

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.

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*, 2015a.

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, 2015b.

G. Barcza, Ö. Legeza, K. H. Marti, and M. Reiher Quantum information analysis of electronic states at different molecular structures. In *Physical Review A*, 83(1):012508, 2011.

K. Batselier and N. Wong. A constructive arbitrary-degree Kronecker product decomposition of tensors. *arXiv preprint arXiv:1507.08805*, 2015.

K. Batselier, H. Liu, and N. Wong. A constructive algorithm for decomposing a tensor into a finite sum of orthonormal rank-1 terms. *SIAM Journal on Matrix Analysis and Applications*, 36(3):1315–1337, 2015.

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

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

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

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.

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. URL http://arxiv.org/abs/1402.6964.

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.

M. Bolten, K. Kahl, and S. Sokolović. Multigrid methods for tensor structured Markov chains with low rank approximation. *SIAM Journal on Scientific Computing*, 38(2):A649–A667, 2016. URL http://adsabs.harvard.edu/abs/2016arXiv160506246B.

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.

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.

H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.

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.

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

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.

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

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.

G. Chabriel, M. Kleinsteuber, E. Moreau, H. Shen, P. Tichavský, 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.

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

T.-L. Chen, D. D. Chang, S.-Y. Huang, H. Chen, C. Lin, and W. Wang. Integrating multiple random sketches for singular value decomposition. *arXiv e-prints*, 2016.

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.

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

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.

A. Cichocki. Tensor decompositions: A new concept in brain data analysis? *arXiv preprint arXiv:1305.0395*, 2013a.

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 2013b. URL http://arxiv.org/abs/1403.2048.

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

A. Cichocki and S. Amari. *Adaptive Blind Signal and Image Processing: Learning Algorithms and Applications.* John Wiley & Sons, Ltd, 2003.

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.

A. Cichocki, S. Cruces, and S. Amari. Log-determinant divergences revisited: Alpha-beta and gamma log-det divergences. *Entropy*, 17(5):2988–3034, 2015a.

A. Cichocki, D. Mandic, A.-H. Phan, C. Caiafa, 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, 2015b.

N. Cohen and A. Shashua. Convolutional rectifier networks as generalized tensor decompositions. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 955–963, 2016.

N. Cohen, O. Sharir, and A. Shashua. On the expressive power of deep learning: A tensor analysis. In *29th Annual Conference on Learning Theory*, pages 698–728, 2016.

P. Comon. Tensors: a brief introduction. *IEEE Signal Processing Magazine*, 31(3):44–53, 2014.

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

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.

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.

E. Corona, A. Rahimian, and D. Zorin. A Tensor-Train accelerated solver for integral equations in complex geometries. *arXiv preprint arXiv:1511.06029*, November 2015.

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*, 106(495):775–790, 2011. URL http://pubs.amstat.org/doi/abs/10.1198/jasa.2011.ap10089.

A. Critch and J. Morton. Algebraic geometry of matrix product states. *Symmetry, Integrability and Geometry: Methods and Applications (SIGMA)*, 10: 095, 2014. .

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

A. L. F. de Almeida, G. Favier, J. C. M. Mota, and J. P. C. L. da Costa. Overview of tensor decompositions with applications to communications. In R. F. Coelho, V. H. Nascimento, R. L. de Queiroz, J. M. T. Romano, and C. C. Cavalcante, editors, *Signals and Images: Advances and Results in Speech, Estimation, Compression, Recognition, Filtering, and Processing*, chapter 12, pages 325–355. CRC Press, 2015.

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

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.

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. URL http://publi-etis.ensea.fr/2008/De08e. Special Issue on Tensor Decompositions and Applications.

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.

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.

L. De Lathauwer, B. De Moor, and J. Vandewalle. A Multilinear singular value decomposition. *SIAM Journal on Matrix Analysis Applications*, 21:1253–1278, 2000a.

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

W. de Launey and J. Seberry. The strong Kronecker product. *Journal of Combinatorial Theory, Series A*, 66(2):192–213, 1994. URL http://dblp.uni-trier.de/db/journals/jct/jcta66.html#LauneyS94.

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.

A. Desai, M. Ghashami, and J. M. Phillips. Improved practical matrix sketching with guarantees. *IEEE Transactions on Knowledge and Data Engineering*, 28(7):1678–1690, 2016.

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

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.

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

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.

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.

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.

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.

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.

G. Ehlers, J. Sólyom, Ö. Legeza, and R.M. Noack. Entanglement structure of the Hubbard model in momentum space. *Physical Review B*, 92(23): 235116, 2015.

M. Espig, M. Schuster, A. Killaitis, N. Waldren, P. Wähnert, S. Handschuh, and H. Auer. TensorCalculus library, 2012. URL http://gitorious.org/tensorcalculus.

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.

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

G. Evenbly and S. R. White. Entanglement renormalization and wavelets. *Physical Review Letters*, 116(14):140403, 2016.

H. Fanaee-T and J. Gama. Tensor-based anomaly detection: An interdisciplinary survey. *Knowledge-Based Systems*, 2016.

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

J. Garcke, M. Griebel, and M. Thess. Data mining with sparse grids. *Computing*, 67(3):225–253, 2001.

S. Garreis and M. Ulbrich. Constrained optimization with low-rank tensors and applications to parametric problems with PDEs. *SIAM Journal on Scientific Computing* (accepted), 2016.

M. Ghashami, E. Liberty, and J. M. Phillips. Efficient frequent directions algorithm for sparse matrices. *arXiv preprint arXiv:1602.00412*, 2016.

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

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

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

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

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

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.

Z.-C. Gu, M. Levin, B. Swingle, and X.-G. Wen. Tensor-product representations for string-net condensed states. *Physical Review B*, 79(8):085118, 2009.

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.

W. Hackbusch. *Tensor Spaces and Numerical Tensor Calculus*, volume 42 of *Springer Series in Computational Mathematics*. Springer, Heidelberg, 2012.

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

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.

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

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.

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

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. URL <http://dx.doi.org/10.1137/100818893>.

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.

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.

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

C. Hubig, I. P. McCulloch, U. Schollwöck, and F. A. Wolf. Strictly single-site DMRG algorithm with subspace expansion. *Physical Review B*, 91(15): 155115, 2015.

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

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

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.

B. Jiang, F. Yang, and S. Zhang. Tensor and its Tucker core: The invariance relationships. *arXiv e-prints arXiv:1601.01469*, January 2016.

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.

Y.-J. Kao, Y.-D. 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.

L. Karlsson, D. Kressner, and A. Uschmajew. Parallel algorithms for tensor completion in the CP format. *Parallel Computing*, 57:222–234, 2016.

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.

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.

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, 2013a.

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, 2013b.

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.

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

B. N. Khoromskij. Tensors-structured numerical methods in scientific computing: Survey on recent advances. *Chemometrics and Intelligent Laboratory Systems*, 110(1):1–19, 2011b. URL http://www.mis.mpg.de/de/publications/preprints/2010/prepr2010-21.html.

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.

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.

S. Klus and C. Schütte. Towards tensor-based methods for the numerical approximation of the Perron-Frobenius and Koopman operator. *arXiv e-prints arXiv:1512.06527*, December 2015.

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

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.

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.

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, 2014a.

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

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

J. B. Kruskal. Three-way arrays: Rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear Algebra and its Applications*, 18(2):95–138, 1977.

V. Kuleshov, A. T. Chaganty, and P. Liang. Tensor factorization via matrix factorization. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, pages 507–516, 2015.

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.

N. Lee and A. Cichocki. Tensor train decompositions for higher order regression with LASSO penalties. In *Workshop on Tensor Decompositions and Applications (TDA2016)*, 2016a.

N. Lee and A. Cichocki. Regularized computation of approximate pseudoinverse of large matrices using low-rank tensor train decompositions. *SIAM Journal on Matrix Analysis and Applications*, 37(2):598–623, 2016b. URL http://adsabs.harvard.edu/abs/2015arXiv150601959L.

N. Lee and A. Cichocki. Fundamental tensor operations for large-scale data analysis using tensor network formats. *Multidimensional Systems and Signal Processing* (accepted), 2016c.

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.

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

S. Liao, T. Vejchodský, 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.

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.

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

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.

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

M. Lubasch, J. I. Cirac, and M.-C. Banuls. Unifying projected entangled pair state contractions. *New Journal of Physics*, 16(3):033014, 2014. URL http://stacks.iop.org/1367-2630/16/i=3/a=033014.

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.

M. W. Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends in Machine Learning*, 3(2):123–224, 2011.

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

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.

H. Matsueda. Analytic optimization of a MERA network and its relevance to quantum integrability and wavelet. *arXiv preprint arXiv:1608.02205*, 2016.

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

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

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

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.

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.

N. Nakatani and G. K. L. Chan. Efficient tree tensor network states (TTNS) for quantum chemistry: Generalizations of the density matrix renormalization group algorithm. *The Journal of Chemical Physics*, 2013. URL http://arxiv.org/pdf/1302.2298.pdf.

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

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

N. H. Nguyen, P. Drineas, and T. D. Tran. Tensor sparsification via a bound on the spectral norm of random tensors. *Information and Inference*, page iav004, 2015.

M. 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.

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

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

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

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.

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

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.

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.

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

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

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.

E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos. Tensors for data mining and data fusion: Models, applications, and scalable algorithms. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(2):16, 2016.

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

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. URL http://dl.acm.org/citation.cfm?id=2011832.2011833.

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

N. Pham and R. Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 239–247. ACM, 2013.

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.

A.-H. Phan, P. Tichavský, and A. Cichocki. Fast alternating ls algorithms for high order candecomp/parafac tensor factorizations. *IEEE Transactions on Signal Processing*, 61(19):4834–4846, 2013a.

A.-H. Phan, P. Tichavský, and A. Cichocki. Tensor deflation for CANDE-COMP/PARAFAC – Part I: Alternating subspace update algorithm. *IEEE Transactions on Signal Processing*, 63(22):5924–5938, 2015a.

A.-H. Phan, A. Cichocki, A. Uschmajew, P. Tichavský, G. Luta, and D. Mandic. Tensor networks for latent variable analysis. Part I: Algorithms for tensor train decomposition. *ArXiv e-prints*, 2016.

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.

A.-H. Phan, A. Cichocki, P. Tichavský, 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.

A.-H. Phan, A. Cichocki, P. Tichavský, R. Zdunek, and S. R. Lehky. From basis components to complex structural patterns. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26–31, 2013*, pages 3228–3232, 2013b. URL http://dx.doi.org/10.1109/ICASSP.2013.6638254.

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, 2013c. URL http://arxiv.org/pdf/1205.2584.pdf.

A.-H. 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 2015b. URL http://dx.doi.org/10.1109/ICASSP.2015.7178355.

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

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.

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

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.

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

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

N. Schuch, I. Cirac, and D. Pérez-García. PEPS as ground states: Degeneracy and topology. *Annals of Physics*, 325(10):2153–2192, 2010.

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

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

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. URL http://www.sandia.gov/~tgkolda/tdw2004/Nikos04.pdf.

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

N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos. Tensor decomposition for signal processing and machine learning. *arXiv e-prints arXiv:1607.01668*, 2016.

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

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.

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.

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.

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. URL http://dx.doi.org/10.1109/TSP.2013.2276416.

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.

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

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

E. M. Stoudenmire and S. R. White. Minimally entangled typical thermal state algorithms. *New Journal of Physics*, 12(5):055026, 2010.

J. Sun, D. 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.

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.

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

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.

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

M. K. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, pages 567–574, 2009.

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

L. N. Trefethen. Cubature, approximation, and isotropy in the hypercube. *SIAM Review*, Forthcoming, 2017. URL https://people.maths.ox.ac.uk/trefethen/hypercube.pdf.

V. Tresp, C. Esteban, Y. Yang, S. Baier, and D. Krompaß. Learning with memory embeddings. *arXiv preprint arXiv:1511.07972*, 2015.

J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher. Randomized single-view algorithms for low-rank matrix approximation. *arXiv e-prints*, 2016.

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

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.

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

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.

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.

F. Verstraete, V. Murg, and 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.

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.

G. Vidal. Efficient classical simulation of slightly entangled quantum computations. *Physical Review Letters*, 91(14):147902, 2003.

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.

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.

D. Wang, H. Shen, and Y. Truong. Efficient dimension reduction for high-dimensional matrix-valued data. *Neurocomputing*, 190:25–34, 2016.

H. Wang and M. Thoss. Multilayer formulation of the multiconfiguration time-dependent Hartree theory. *Journal of Chemical Physics*, 119(3):1289–1299, 2003.

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.

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.

Y. Wang, H.-Y. Tung, A. Smola, and A. Anandkumar. Fast and guaranteed tensor decomposition via sketching. In *Advances in Neural Information Processing Systems*, pages 991–999, 2015.

S. R. White. Density-matrix algorithms for quantum renormalization groups. *Physical Review B*, 48(14):10345, 1993.

Z. Xu, F. Yan, and Y. 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.

Y. Yang and T. Hospedales. Deep multi-task representation learning: A tensor factorisation approach. *arXiv preprint arXiv:1605.06391*, 2016. URL http://adsabs.harvard.edu/abs/2016arXiv160506391Y.

T. Yokota, Q. Zhao, and A. Cichocki. Smooth PARAFAC decomposition for tensor completion. *IEEE Transactions on Signal Processing*, 64(20): 5423–5436, 2016.

T. Yokota, N. Lee, and A. Cichocki. Robust multilinear tensor rank estimation using higher order singular value decomposition and information criteria. *IEEE Transactions on Signal Processing*, accepted, 2017.

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.

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.

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, 2013a.

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, 2013b.

Q. Zhao, G. Zhou, S. Xie, L. Zhang, and A. Cichocki. Tensor ring decomposition. *CoRR*, abs/1606.05535, 2016. URL http://arxiv.org/abs/1606.05535.

S. Zhe, Y. Qi, Y. Park, Z. Xu, I. Molloy, and S. Chari. DinTucker: Scaling up Gaussian process models on large multidimensional arrays. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11959/11888.

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, 2012a.

G. Zhou and A. Cichocki. Canonical Polyadic Decomposition based on a single mode blind source separation. *IEEE Signal Processing Letters*, 19 (8):523–526, 2012b.

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.

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.

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), 2016a.

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, 2016b.