Foundations and Trends® in Machine Learning
Vol. 9, No. 4-5 (2016) 249–429

DOI: 10.1561/2200000059

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.

A. Cichocki *et al. Tensor Networks for Dimensionality Reduction and Large-Scale Optimization Part 1 Low-Rank Tensor Decompositions.* Foundations and Trends® in Machine Learning, vol. 9, no. 4-5, pp. 249–429, 2016.
DOI: 10.1561/2200000059.

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

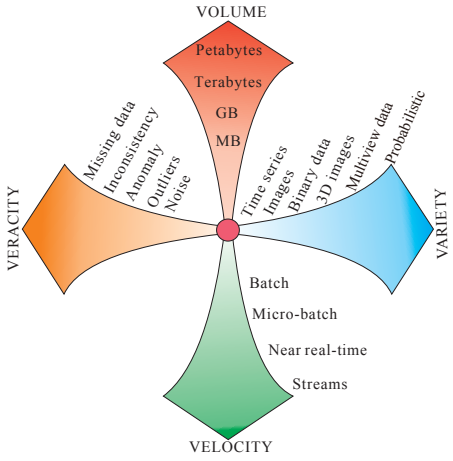
and/or inconsistent data; high Variety demands the fusion of different data types, e.g., continuous, discrete, binary, time series, images, video, text, probabilistic or multi-view. Some applications give rise to additional "V challenges", such as Visualization, Variability and Value. The Value feature 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.

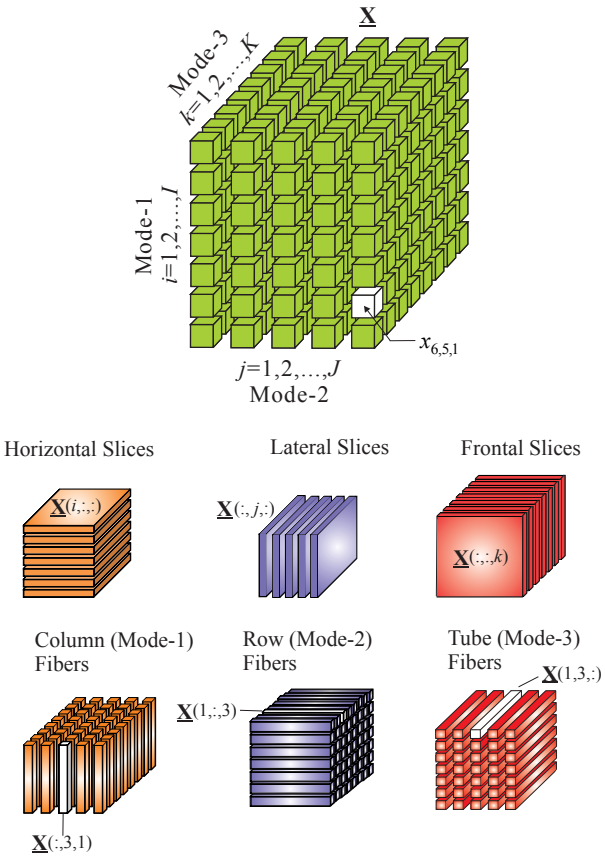


Figure 1.2: A 3rd-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$, with entries $x_{i,j,k} = \underline{\mathbf{X}}(i,j,k)$, and its subtensors: slices (middle) and fibers (bottom). All fibers are treated as column vectors.

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 $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_R] \in \mathbb{R}^{I \times R}$ are

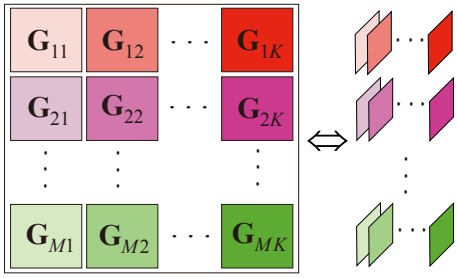


Figure 1.3: A block matrix and its representation as a 4th-order tensor, created by reshaping (or a projection) of blocks in the rows into lateral slices of 3rd-order tensors.

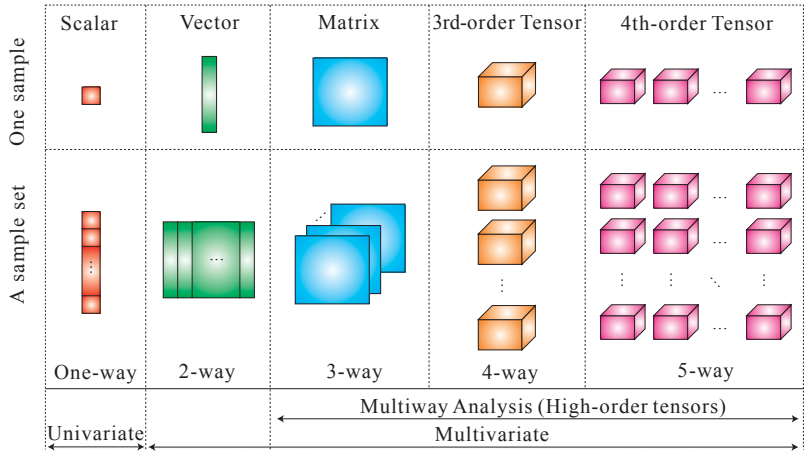
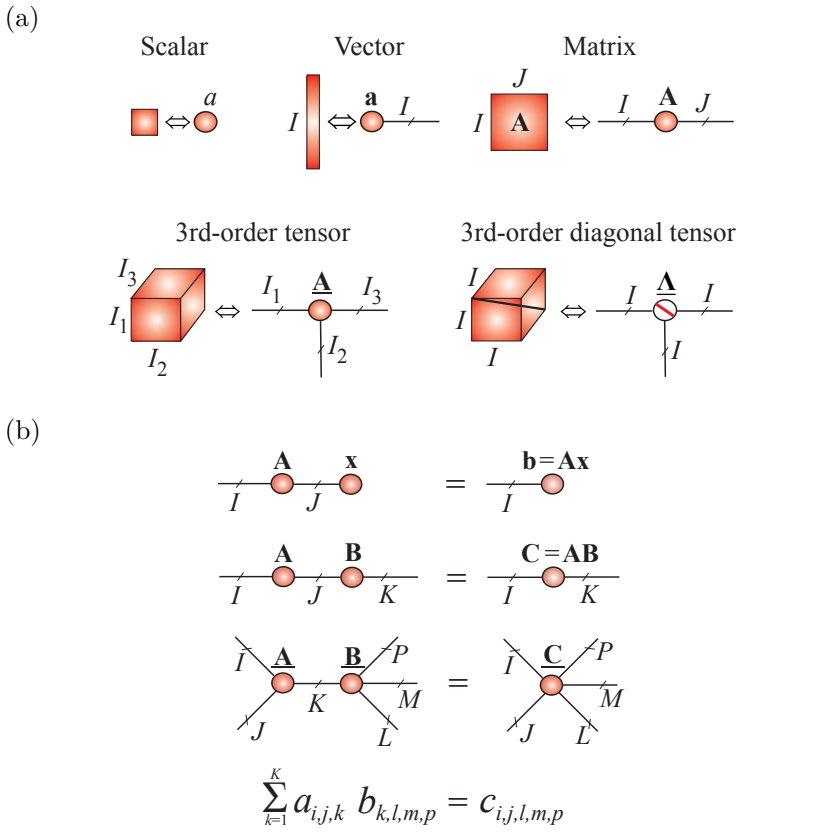


Figure 1.4: Graphical representation of multiway array (tensor) data of increasing structural complexity and "Volume" (see (Olivieri, 2008) for more detail).

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



$$\sum_{k=1}^{K} a_{i,j,k}\, b_{k,l,m,p} = c_{i,j,l,m,p}$$

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
$\underline{\mathbf{\Lambda}} \in \mathbb{R}^{R \times R \times \cdots \times R}$	Nth-order diagonal core tensor with nonzero entries λ_r on the main diagonal
$\mathbf{A}^{\mathrm{T}}$, $\mathbf{A}^{-1}$, $\mathbf{A}^{\dagger}$	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
$\otimes_L$, $\boxtimes$	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 $\vert\Psi\rangle$
row vector $\mathbf{x}^{\mathrm{T}} \in \mathbb{R}^{1 \times I}$	bra $\langle\Psi\vert$
inner product $\langle \mathbf{x}, \mathbf{x} \rangle = \mathbf{x}^{\mathrm{T}}\mathbf{x}$	$\langle\Psi\vert\Psi\rangle$
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

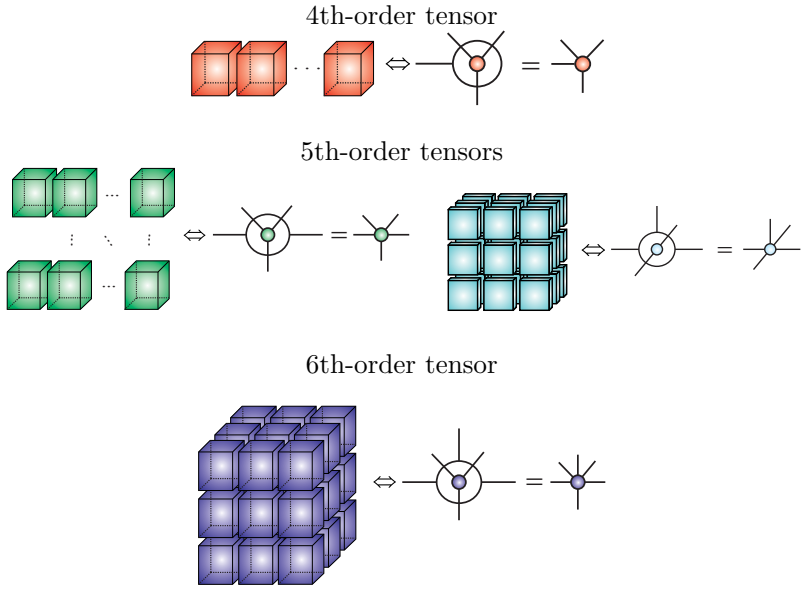


Figure 1.6: Graphical representations and symbols for higher-order block tensors. Each block represents either a 3rd-order tensor or a 2nd-order tensor. The outer circle indicates a global structure of the block tensor (e.g. a vector, a matrix, a 3rd-order block tensor), while the inner circle reflects 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

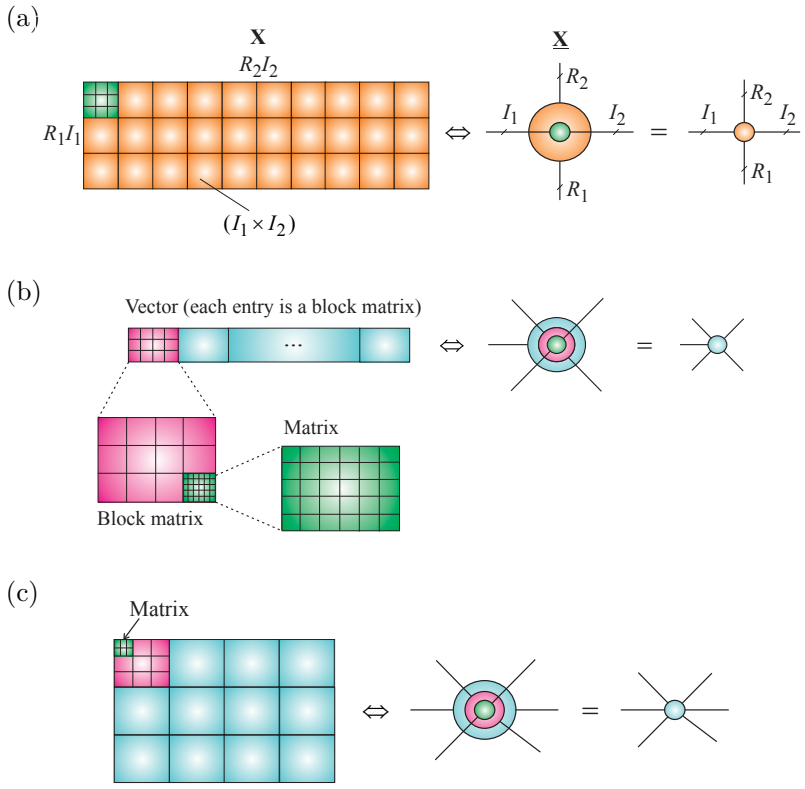


Figure 1.7: Hierarchical matrix structures and their symbolic representation as tensors. (a) A 4th-order tensor representation for a block matrix $\mathbf{X} \in \mathbb{R}^{R_1 I_1 \times R_2 I_2}$ (a matrix of matrices), which comprises block matrices $\mathbf{X}_{r_1, r_2} \in \mathbb{R}^{I_1 \times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

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

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

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

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

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

  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

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

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

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

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

  with the equivalent compact matrix representation

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

  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

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

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

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

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

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

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

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.

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 $\mathbf{c}_j = \mathbf{a}_j \otimes_L \mathbf{b}_j \in \mathbb{R}^{IK}$

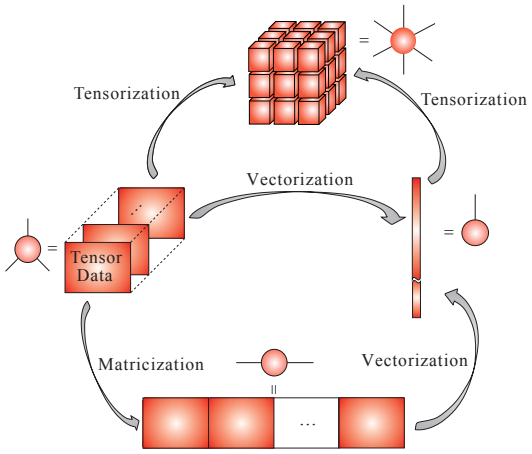


Figure 2.1: Tensor reshaping operations: Matricization, vectorization and tensorization. Matricization refers to converting a tensor into a matrix, vectorization to converting a tensor or a matrix into a vector, while tensorization refers to converting a vector, a matrix or a low-order tensor into a 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 vec$(\cdot)$ or 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, $\text{vec}(\mathbf{a} \circ \mathbf{b}) = \mathbf{b} \otimes \mathbf{a} = \mathbf{a} \otimes_L \mathbf{b}$.

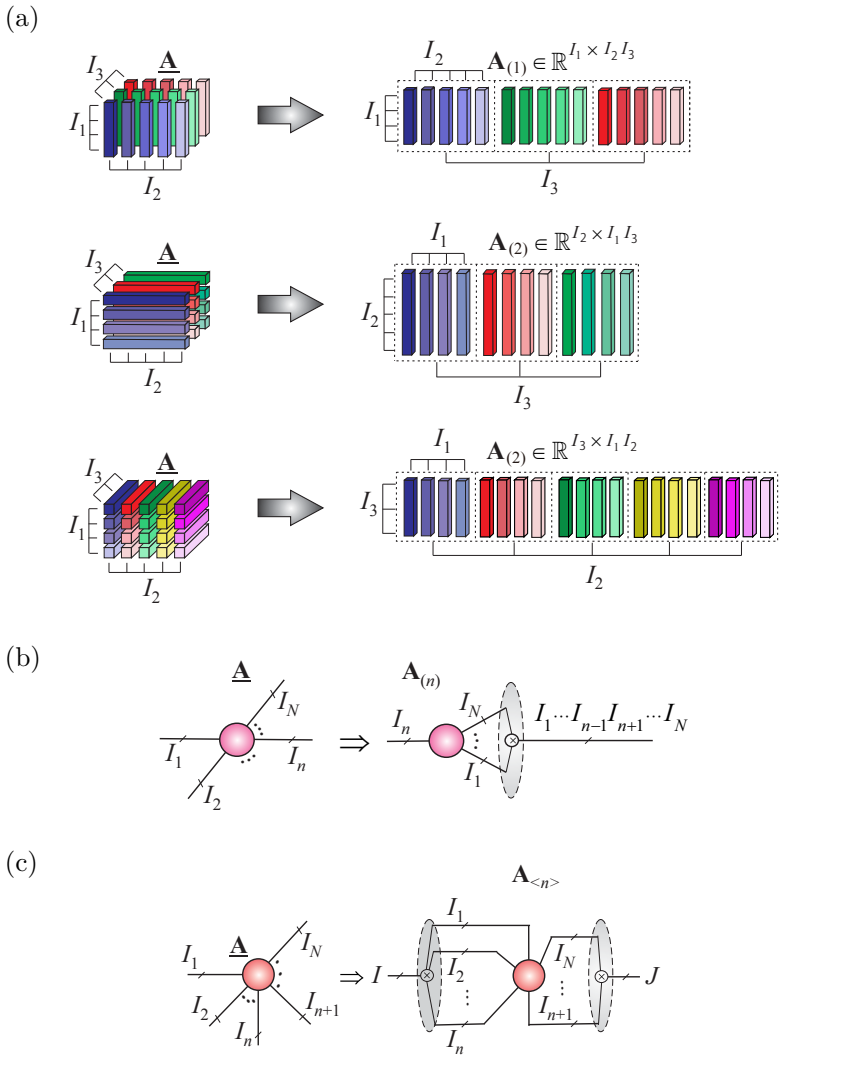


Figure 2.2: Matricization (flattening, unfolding) used in tensor reshaping. (a) Mode-1, mode-2, and mode-3 matricizations of a 3rd-order tensor, from the top to the bottom panel. (b) Tensor network diagram for the mode-n matricization of an Nth-order tensor, $\underline{\mathbf{A}} \in \mathbb{R}^{I_1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

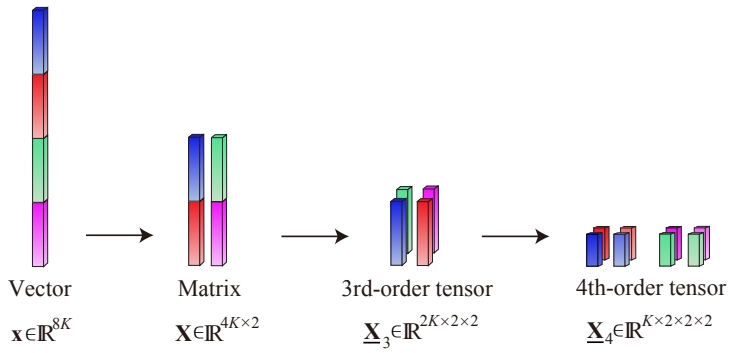


Figure 2.3: Tensorization of a vector into a matrix, 3rd-order tensor and 4th-order tensor.

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

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

As special cases we immediately have (see Figure 2.2)

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

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.

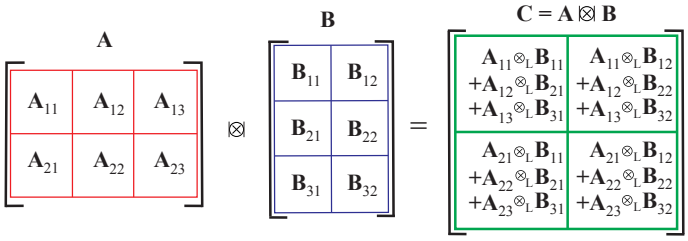


Figure 2.4: Illustration of the strong Kronecker product of two block matrices, $\mathbf{A} = [\mathbf{A}_{r_1,r_2}] \in \mathbb{R}^{R_1 I_1 \times R_2 J_1}$ and $\mathbf{B} = [\mathbf{B}_{r_2,r_3}] \in \mathbb{R}^{R_2 I_2 \times R_3 J_2}$, which is defined as a block matrix $\mathbf{C} = \mathbf{A} \boxtimes \mathbf{B} \in \mathbb{R}^{R_1 I_1 I_2 \times R_3 J_1 J_2}$, 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)

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

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

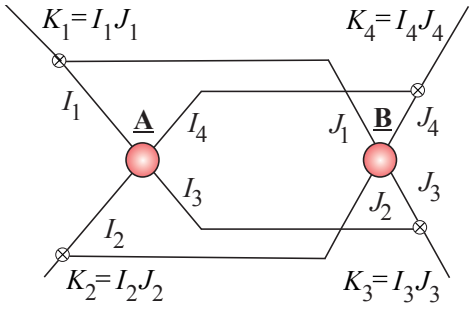


Figure 2.5: The left Kronecker product of two 4th-order tensors, $\underline{\mathbf{A}}$ and $\underline{\mathbf{B}}$, yields a 4th-order tensor, $\underline{\mathbf{C}} = \underline{\mathbf{A}} \otimes_L \underline{\mathbf{B}} \in \mathbb{R}^{I_1 J_1 \times \cdots \times I_4 J_4}$, with entries $c_{k_1,k_2,k_3,k_4} = a_{i_1,\ldots,i_4}\, b_{j_1,\ldots,j_4}$, 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]

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

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

[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 $1 \leqslant k_n \leqslant I_n, \quad \forall n$, $\underline{\mathbf{C}}(k_1, \ldots, k_N) = \underline{\mathbf{B}}(k_1 - I_1, \ldots, k_N - I_N)$ if $I_n < k_n \leqslant I_n + J_n, \quad \forall n$, 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 $I_m = J_m, \quad \forall m \neq n$ 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)

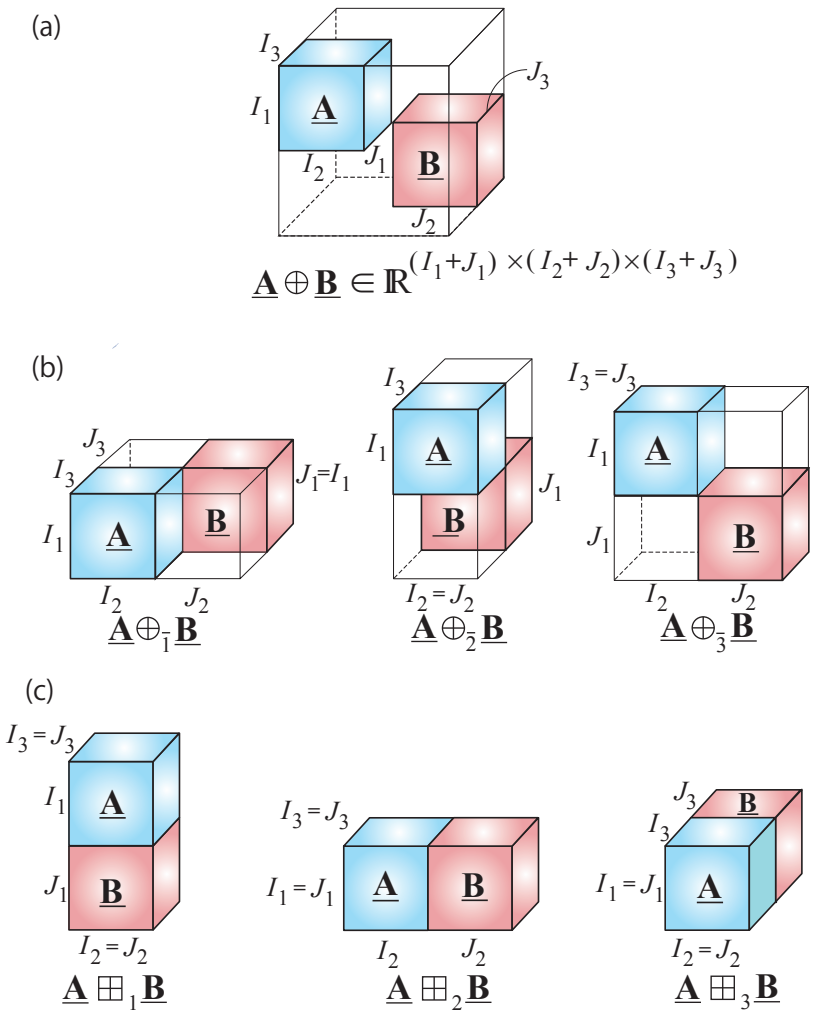


Figure 2.6: Illustration of the direct sum, partial direct sum and concatenation operators of two 3rd-order tensors. (a) Direct sum. (b) Partial (mode-1, mode-2, and mode-3) direct sum. (c) Concatenations along mode-1,2,3.

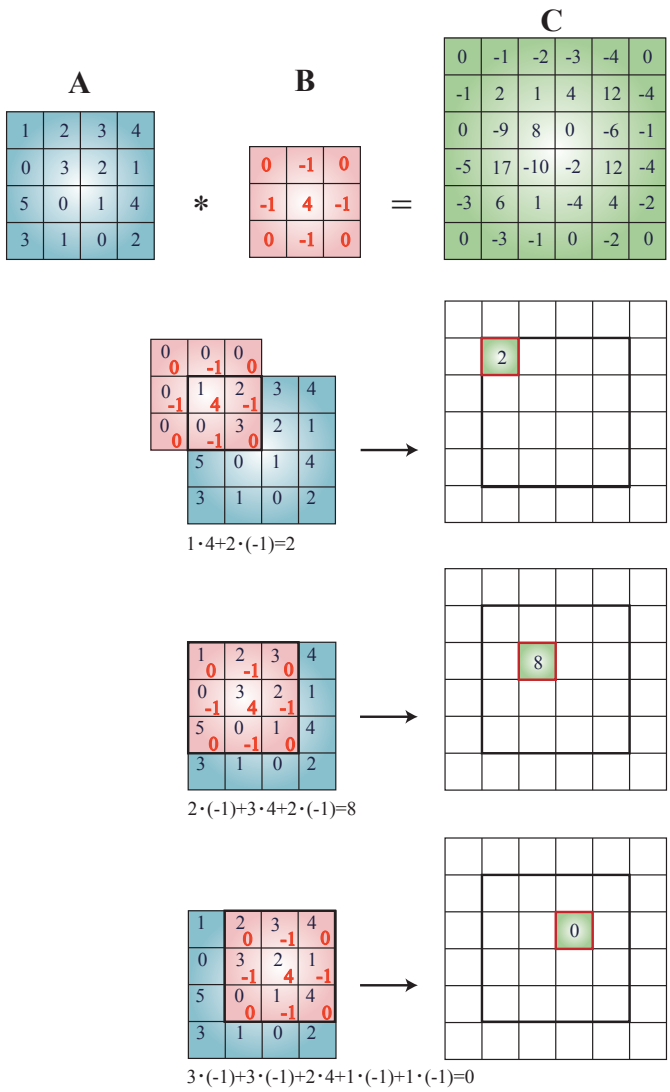


Figure 2.7: Illustration of the 2D convolution operator, performed through a sliding window operation along both the horizontal and vertical index.

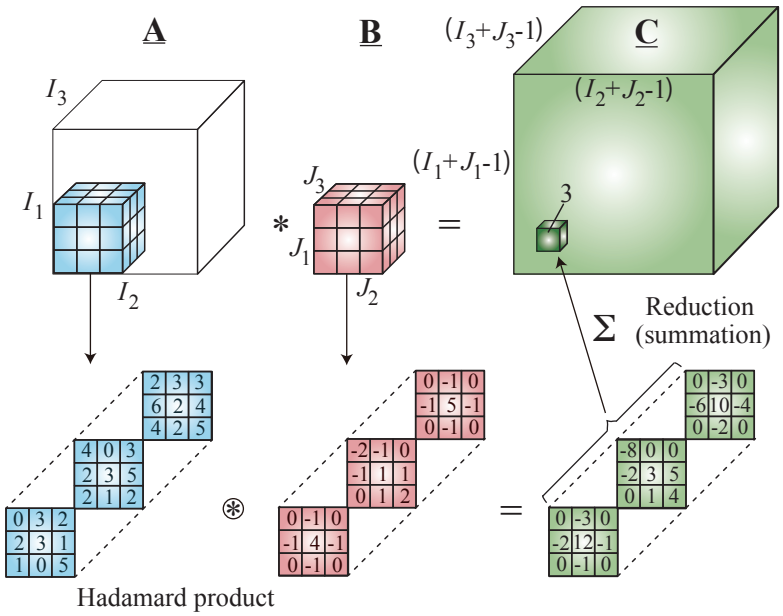


Figure 2.8: Illustration of the 3D convolution operator, performed through a sliding window operation along all three indices.

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

Note that for 1st-order tensors (vectors), the tensor product reduces to the standard outer product of two nonzero vectors, $\mathbf{a} \in \mathbb{R}^I$ and $\mathbf{b} \in \mathbb{R}^J$, which yields a rank-1 matrix, $\mathbf{X} = \mathbf{a} \circ \mathbf{b} = \mathbf{a}\mathbf{b}^{\mathrm{T}} \in \mathbb{R}^{I \times J}$. 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 $x_{i_1, i_2, \ldots, i_N} = b^{(1)}_{i_1} b^{(2)}_{i_2} \cdots b^{(N)}_{i_N}$.

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

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

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

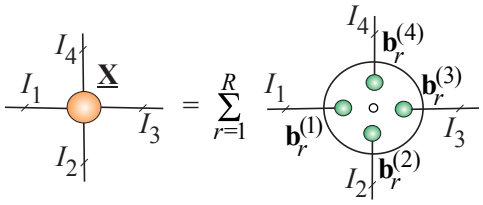


Figure 2.9: The CP decomposition for a 4th-order tensor $\underline{\mathbf{X}}$ of rank R. Observe that the rank-1 subtensors are formed through the outer products of the vectors $\mathbf{b}_r^{(1)}, \ldots, \mathbf{b}_r^{(4)}$, $r = 1, \ldots, R$.

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

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

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

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

with entries

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

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, $\underline{\mathbf{G}} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$,

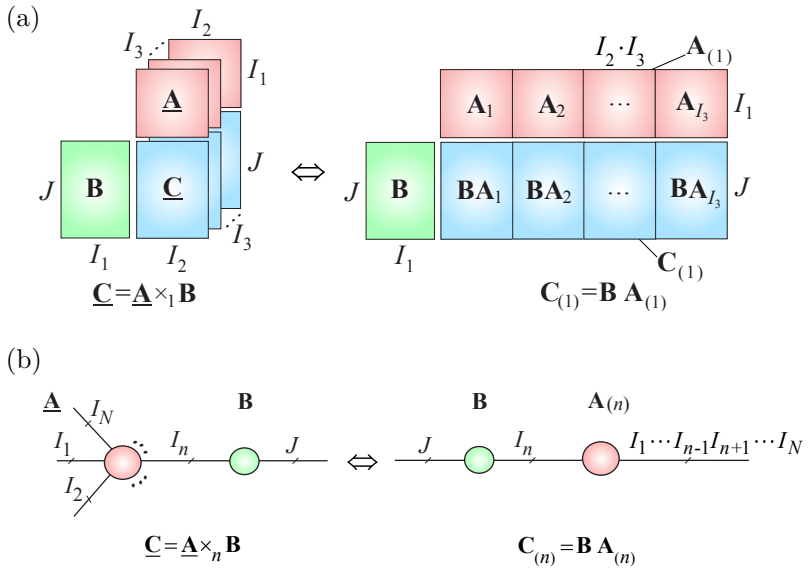


Figure 2.10: Illustration of the multilinear mode-n product, also known as the TTM (Tensor-Times-Matrix) product, performed in the tensor format (left) and the matrix format (right). (a) Mode-1 product of a 3rd-order tensor, $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, and a factor (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))

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

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

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

with entries

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

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))

$$\underline{\mathbf{C}} = \underline{\mathbf{A}} \times_n^m \underline{\mathbf{B}}, \tag{2.16}$$

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

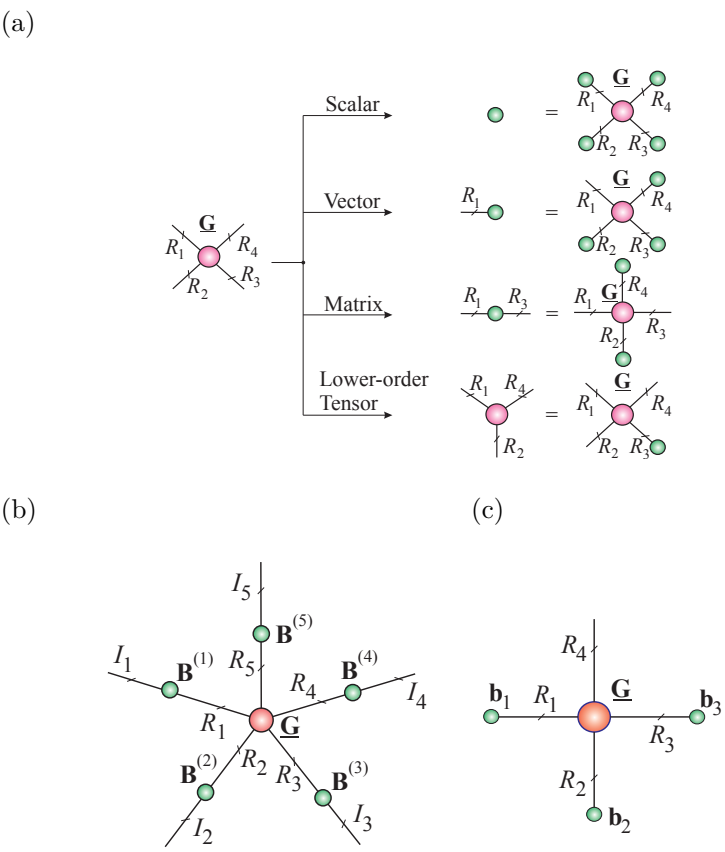


Figure 2.11: Multilinear tensor products in a compact tensor network notation. (a) Transforming and/or compressing a 4th-order tensor, $\underline{\mathbf{G}} \in \mathbb{R}^{R_1 \times R_2 \times R_3 \times R_4}$, into a scalar, vector, matrix and 3rd-order tensor, by multilinear products of the tensor 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 $\mathbf{c} = \underline{\mathbf{G}} \bar{\times}_1 \mathbf{b}_1 \bar{\times}_2 \mathbf{b}_2 \bar{\times}_3 \mathbf{b}_3 \in \mathbb{R}^{R_4}$.

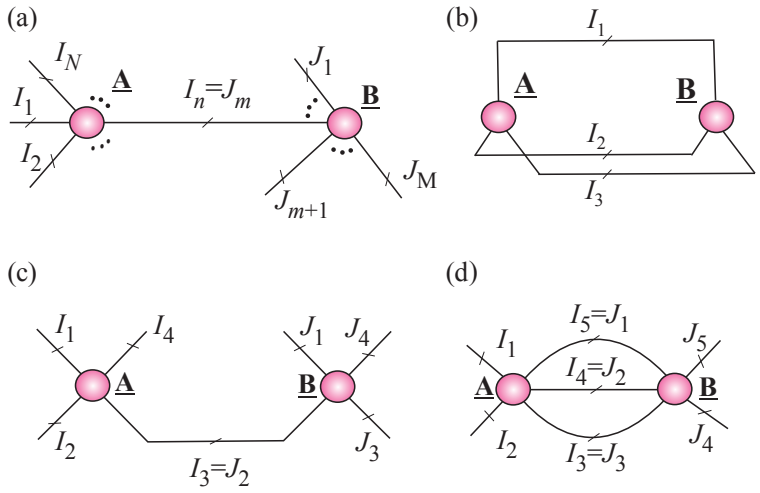


Figure 2.12: Examples of contractions of two tensors. (a) Multilinear product of two tensors is denoted by $\underline{\mathbf{A}} \times_n^m \underline{\mathbf{B}}$. (b) Inner product of two 3rd-order tensors yields a scalar $c = \langle \underline{\mathbf{A}}, \underline{\mathbf{B}} \rangle = \underline{\mathbf{A}} \times_{1,2,3}^{1,2,3} \underline{\mathbf{B}} = \underline{\mathbf{A}} \bar{\times} \underline{\mathbf{B}} = \sum_{i_1,i_2,i_3} a_{i_1,i_2,i_3} \, b_{i_1,i_2,i_3}$. (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

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

In this notation, the multiplications of matrices and vectors can be written as, $\mathbf{A} \times_2^1 \mathbf{B} = \mathbf{A} \times^1 \mathbf{B} = \mathbf{AB}$, $\mathbf{A} \times_2^2 \mathbf{B} = \mathbf{AB}^{\mathrm{T}}$, $\mathbf{A} \times_{1,2}^{1,2} \mathbf{B} = \mathbf{A} \bar{\times} \mathbf{B} = \langle \mathbf{A}, \mathbf{B} \rangle$, 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 $\mathbf{A}_i \in \mathbb{R}^{R \times R}$ $(i = 1, 2, \ldots, I)$, that is, (see bottom of Figure 2.13)

$$\mathbf{a} = [\mathrm{tr}(\mathbf{A}_1), \ldots, \mathrm{tr}(\mathbf{A}_i), \ldots, \mathrm{tr}(\mathbf{A}_I)]^{\mathrm{T}}. \tag{2.19}$$

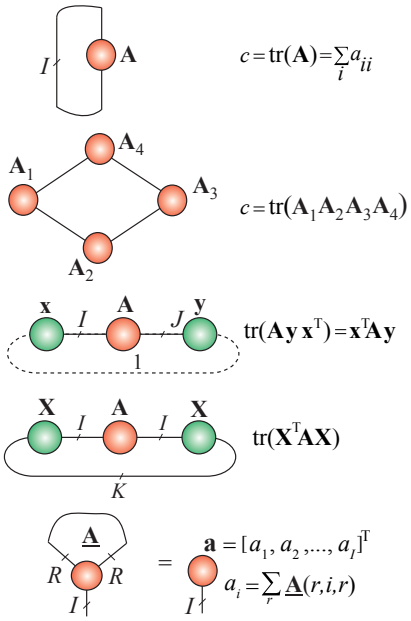


Figure 2.13: Tensor network notation for the traces of matrices (panels 1-4 from the top), and a (partial) tensor trace (tensor self-contraction) of a 3rd-order tensor (bottom panel). Note that graphical representations of the trace of matrices intuitively explain the permutation property of trace operator, 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 $\widetilde{\underline{\mathbf{A}}} = \text{Tr}(\underline{\mathbf{A}}) \in$

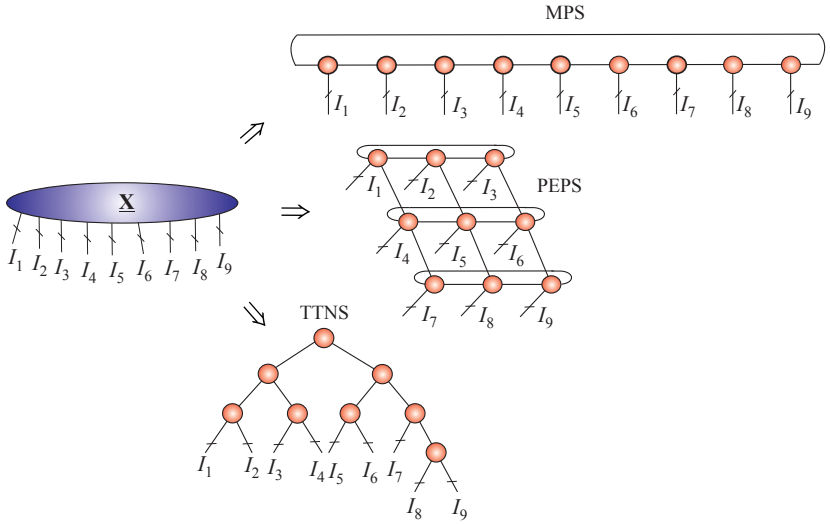


Figure 2.14: Illustration of the decomposition of a 9th-order tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_9}$, into different forms of tensor networks (TNs). In general, the objective is to decompose a very high-order tensor into sparsely (weakly) connected low-order 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

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

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).

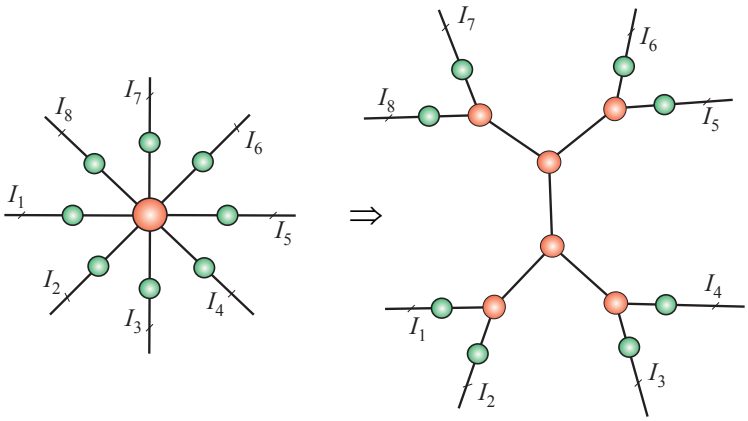


Figure 2.15: The standard Tucker decomposition of an 8th-order tensor into a core tensor (red circle) and eight factor matrices (green circles), and its transformation into an equivalent Hierarchical Tucker (HT) model using interconnected smaller size 3rd-order core tensors and the same factor matrices.

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,..., 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

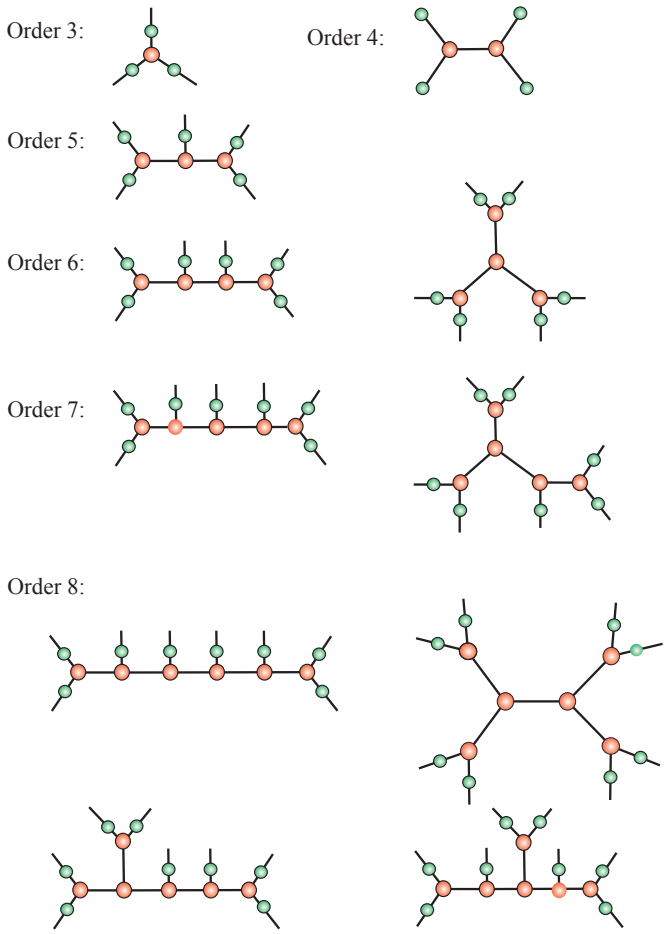


Figure 2.16: Examples of HT/TT models (formats) for distributed Tucker decompositions with 3rd-order cores, for different orders of data tensors. Green circles denote factor matrices (which can be absorbed by core tensors), while red circles indicate cores. Observe that the representations are not unique.

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