# Tensor decompositions: principles and application to food sciences

Jérémy E Cohen, Rasmus Bro, Pierre Comon

# Contents

## Notations

| | |
|---|---|
| $\boldsymbol{x}$ | column vector of components $x_p$, $1 \leq p \leq P$ |
| $s$, $x$, $y$ | sources, observations, separator outputs |
| $R$ | number of sources |
| $P$ | number of sensors |
| $T$ | number of observed samples |
| $\star$ | convolution |
| $\boldsymbol{A}$ | matrix with components $A_{ij}$ |
| $\boldsymbol{A}$, $\boldsymbol{B}$ | mixing and separation matrices |
| $\boldsymbol{G}$, $\boldsymbol{W}$, $\boldsymbol{Q}$ | global, whitening, and separating unitary matrices |
| $\check{g}$ | Fourier transform of $g$ |
| $\widehat{\boldsymbol{s}}$ | estimate of quantity $\boldsymbol{s}$ |
| $p_{\boldsymbol{x}}$ | probability density of $\boldsymbol{x}$ |
| $\psi$ | joint score function |
| $\varphi_i$ | marginal score function of source $s_i$ |
| $\mathbb{E}\,\boldsymbol{x}$, $\mathbb{E}\{\boldsymbol{x}\}$ | mathematical expectation of $\boldsymbol{x}$ |
| $I\{\boldsymbol{y}\}$ or $I(p_{\boldsymbol{y}})$ | mutual information of $\boldsymbol{y}$ |
| $K\{\boldsymbol{x};\boldsymbol{y}\}$ or $K(p_{\boldsymbol{x}};p_{\boldsymbol{y}})$ | Kullback divergence between $p_{\boldsymbol{x}}$ and $p_{\boldsymbol{y}}$ |
| $H\{\boldsymbol{x}\}$ or $H\{p_{\boldsymbol{x}}\}$ | Shannon entropy of $\boldsymbol{x}$ |
| $\mathcal{L}$ | likelihood |
| $\mathcal{A}$, $\mathcal{B}$ | mixing, and separating (non linear) operators |
| $\mathrm{cum}\{x_1, \ldots, x_P\}$ | joint cumulant of variables $\{x_1, \ldots, x_P\}$ |
| $\mathrm{cum}_R\{y\}$ | marginal cumulant of order $R$ of variable $y$ |

| | |
|---|---|
| $\boldsymbol{Q}^{\mathsf{T}}$ | transposition |
| $\boldsymbol{Q}^{\mathsf{H}}$ | conjugate transposition |
| $\boldsymbol{Q}^{*}$ | complex conjugation |
| $\boldsymbol{Q}^{\dagger}$ | pseudo-inverse |
| $\Upsilon$ | contrast function |
| $\mathbb{R}$ | real field |
| $\mathbb{C}$ | complex field |
| $\widehat{\boldsymbol{A}}$ | estimator of mixing matrix |
| $\mathrm{rank}\{\boldsymbol{A}\}$ | rank of matrix $\boldsymbol{A}$ |
| $\mathrm{krank}\{\boldsymbol{A}\}$ | Kruskal's k-rank of matrix $\boldsymbol{A}$ |
| $\mathrm{diag}\{\boldsymbol{A}\}$ | vector whose components are the diagonal of matrix $\boldsymbol{A}$ |
| $\mathrm{Diag}\{\boldsymbol{a}\}$ | diagonal matrix whose entries are those of vector $\boldsymbol{a}$ |
| $\mathrm{trace}\{\boldsymbol{A}\}$ | trace of matrix $\boldsymbol{A}$ |
| $\det \boldsymbol{A}$ | determinant of matrix $\boldsymbol{A}$ |
| $\check{s}(\nu)$ | Fourier transform of process $s(t)$ |
| $\boxtimes$ | Kronecker product between matrices |
| $\odot$ | Khatri-Rao (column-wise Kronecker) product between matrices |
| $\boxdot$ | Hadamard (entry-wise) product between arrays |
| $\otimes$ | tensor product |
| $\bullet_j$ | contraction over index $j$ |
| $\mathcal{S}, \mathcal{G}$ | sets |
| $\mathrm{dom}\, f$ | domain of function $f$ |
| $\mathrm{prox}_f^{\boldsymbol{A}}(\boldsymbol{x})$ | proximity operator of function $f$ within the metric induced by $\boldsymbol{A}$ computed at $\boldsymbol{x}$ |
| $\iota_C$ | indicator function of set $C$ |
| $\mathrm{P}_C$ | projector on set $C$ |
| $\mathrm{epi}\, f$ | epigraph of function $f$ |
| $\underset{C}{\mathrm{argmin}} f$ | minimum argument of $f$ over set $C$ |
| $\underset{C}{\mathrm{argmax}}\, f$ | maximum argument of $f$ over set $C$ |
| $\sup_C f$ | supremum of $f$ over set $C$ |
| $\inf_C f$ | infimum of $f$ over set $C$ |
| $\nabla f$ | gradient of $f$ |
| $\nabla^2 f$ | Hessian of $f$ |
| $\partial f(\boldsymbol{u})$ | subdifferential set of $f$ at $\boldsymbol{u}$ |
| $f^*$ | conjugate of function $f$ |
| ${}^{\gamma}f$ | Moreau envelope of $f$ of parameter $\gamma$ |
| $\square$ | infimum-convolution |
| $\|\cdot\|_{\mathrm{F}}$ | Frobenius norm |
| $\|\|\|\cdot\|\|\|$ | spectral norm |

# 6

# Tensor decompositions: principles and application to food sciences

*J. Cohen, R. Bro and P. Comon*

## 6.1

## Introduction

Most graduate students fear the concept of tensor, as it reminds them of intricate astrophysics, material science, differential calculus or multilinear algebra formalism. Regarding the fields of signal processing and data science, while it is true that using tensor methods requires understanding at least linear algebra and convex optimization, which are both rich applied mathematics domains, the authors believe that most of this fear about tensors is unjustified in the context of engineering. Indeed, for data scientists, tensors are simply arrays that may have more than two indices, and most of the discussion in this chapter will actually be a generalization of well-understood techniques for matrices to such arrays.

Before entering the technical details of tensor algebra and tensor decomposition methods for analyzing data sets, we shall begin this chapter with a friendly introduction to tensors, so that the reader can hopefully get rid of any anxiety about tensors.

### 6.1.1

### A simplified definition

Let us start with a definition of what a tensor is, within the scope of this chapter.

DEFINITION 6.1 *A (real) tensor $\mathcal{T}$ is an element of $\mathbb{R}^{n_1 \times \cdots \times n_d}$ where $n_i$ are integers greater than or equal to 2. Integer $d$ is called the order of the tensor.*

In other words, we consider tensors as arrays containing real numbers, and a $d$th order tensor is a real array with exactly $d$ dimensions, which we call modes. This means that real matrices are second-order tensors, *i.e.* matrices are tensors with only two modes, while vectors of $\mathbb{R}^n$ are first-order tensors. A third-order tensor is depicted in Figure 6.1. As a

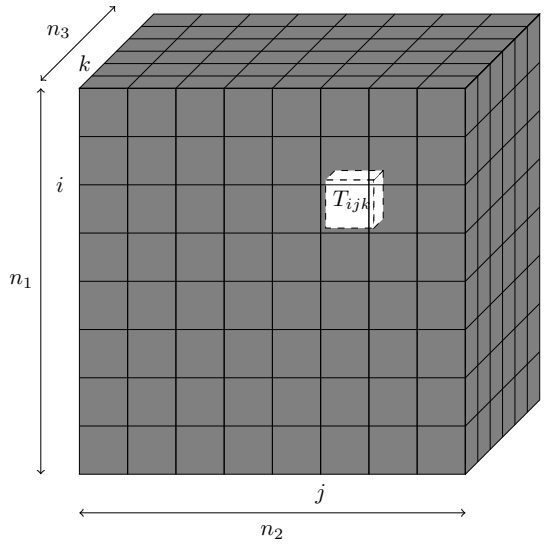



**Figure 6.1** A third-order real tensor is nothing more than a three-way array of size $n_1 \times n_2 \times n_3$, that contains real numbers $T_{i,j,k}$ at each entry $(i, j, k)$. To rephrase using a different vocabulary, this real tensor $T$ has three modes of sizes respectively $n_1$, $n_2$ and $n_3$.

side note, there exist definitions of tensors which are much more general than this one, the most general one in the mathematical sense involving monoidal category theory [1], which is far outside the scope of this chapter. However, an interested reader may turn to [2, 3, 4] for other descriptions of tensors.

The study of tensors as a data structure is necessary in many applications where such $d$-dimensional arrays emerge as efficient representations for the studied phenomenon. For instance, it may be that sensors directly output tensors, *e.g.* a video or a 3D image. It is also often the case that data are collected along several experimental variables, which become modes in a measurement tensor. This is the case with fluorescence spectroscopy; see the end of this introduction for a description of how fluorescence spectroscopy measurements lead to data being contained naturally in a tensor. Finally, it is sometimes profitable to augment a dataset to obtain a tensor. An example of such augmentation is obtained from stacked modified images that are shifted versions of an original one. The obtained data set is a collection of matrices, *i.e.* a third-order tensor.

### 6.1.2

### Separability: a key concept for tensor decomposition model

We have defined what a tensor is and how one may end up manipulating such an object. However, we have yet to define the type of information that we want to extract from

tensors.

Recalling from Definition 6.1 that modes are the various ways/dimensions of a tensor, this chapter describes some tools that exploits mode-wise information to explain the content of a tensor. The main mathematical concept that formalizes this idea of collecting mode-wise information to describe the whole tensor is "separability". By definition, a separable function $f(x, y, z)$ verifies the following equality:

$$f(x, y, z) = f_1(x)f_2(y)f_3(z) \tag{6.1}$$

for some functions $f_1, f_2, f_3$. Of course, very few functions are of this form. A separable function $f$ is therefore entirely described by a triplet of functions $f_i$, each depending only on a single variable. Therefore, they are desirable to describe multivariate patterns in an understandable, mode-wise manner. We shall see how to do that in the following sections.

The link between separable functions and tensors is the following: a tensor can be seen as an array collecting values of a sampled multivariate function. Indeed, one can always define a function $f(x, y, z)$ such that

$$T_{ijk} = f(x_i, y_j, z_k). \tag{6.2}$$

If this function $f$ is separable as defined in (6.1), then the corresponding tensor has received several names, and in particular *simple* tensor, or *decomposable* tensor in the early literature. In the remainder, we shall assume the terminology of *separable tensor*, which is more intuitive, and directly related to (6.1). This leads to the following definition:

DEFINITION 6.2 *A separable tensor is a tensor from $\mathbb{R}^{n_1 \times n_2 \times n_3}$ whose general term $\mathcal{T}$ verifies the following equality:*

$$\mathcal{T}_{i,j,k} = a_i b_j c_k \tag{6.3}$$

*where $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ are real vectors from respectively $\mathbb{R}^{n_1}, \mathbb{R}^{n_2}, \mathbb{R}^{n_3}$.*

Separable tensors are thus a formal description of what was referred in the beginning of this introduction as "patterns explaining the data across all modes". We shall see in Section 6.2.2 the definition of tensor rank; it will then be clear that separable tensors are nothing else but *rank-one* tensors. Because rank-one tensors make up for only one simple pattern, a more complex data tensor should be composed of several of them. This is the rationale under the Canonical Polyadic Decomposition (CPD), which is more formally described in Section 6.2.2. Taking this reasoning one step further, a tensor decomposition model always writes a tensor as a sum of separable terms. Therefore, tensor decomposition models always aim at expressing global information contained in a tensor using mode-wise descriptors.

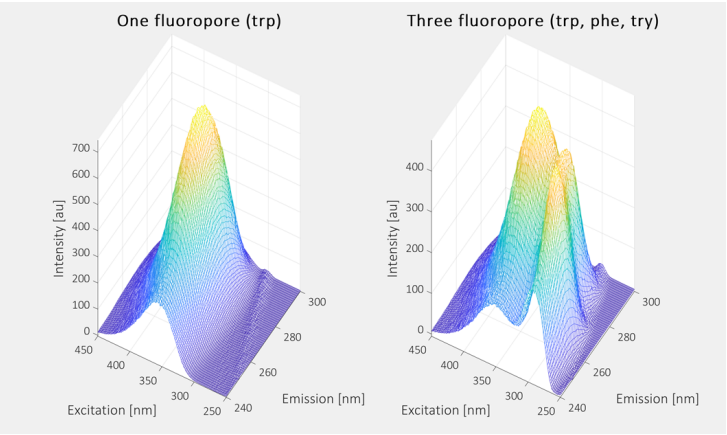



**Figure 6.2** Leftmost a fluorescence Excitation-Emission Matrix (EEM) of a single fluorophore, the amino acid tryptophan, and rightmost a sample containing three different amino acids; each with a unique contribution (peak) to the EEM. The three amino acids are tryptophan, tyrosine and phenylalanine. The leftmost matrix is separable up to noise, while the rightmost matrix is a sum of three separable matrices, also up to noise.

### 6.1.3

### Core aspects of tensor decomposition: the fluorescence spectroscopy example

To conclude this introduction, we explore how fluorescence spectroscopy measurements can lead to a tensor which has a CPD structure, and how this structure can be exploited in practice. Further details are provided in Section 6.6 dedicated to applications of tensor decompositions in food sciences.

Fluorescence spectroscopy takes advantage of the fact that in food sciences, many interesting compounds have their own fluorescent response to stimulation. To be more precise, these compounds react only in a specific range of stimulation wavelength at various intensities. Fluorophores respond to light excitation by emitting a light that has a spread spectrum, called the emission spectrum. The emitted intensity consequently depends on excitation and emission wavelengths.

Given one sample of possible several fluorescent chemicals, and stimulating the sample with a light of varying wavelengths, a matrix of data is obtained. The two modes of the matrix are the two experimental variables, *i.e.* the excitation wavelength $\lambda_{ex}$ and the emission wavelength $\lambda_{em}$, and the elements of the matrix are the measured intensity values of the light that the sample outputs. An example is provided in Figure 6.2.

Such a matrix is called a FEEM (Fluorescence Emission Excitation Matrix). Note that FEEM do not restrict to cases where only one compound is present in the solution. But in the case of a single fluorophore, the fluorescence phenomenon is separable with respect to excitation and emission wavelengths and therefore the general term of a FEEM that

contains the spectra of only one compound can be expressed as follows: $\boldsymbol{M}_{\lambda_{ex},\lambda_{em}} = a_{\lambda_{ex}} b_{\lambda_{em}}$, where $\mathbf{a}$ is the excitation spectrum, and $\mathbf{b}$ is the emission spectrum. In other words, $\boldsymbol{M}$ is a rank-one matrix, *i.e.* a rank-one tensor of order 2.

Now suppose that instead of a single sample, one has several samples of the same single chemical, but at various concentrations in the solvent. The fluorescence phenomenon being linear with respect to concentration (using a first order approximation valid at low concentrations), repeating the above reasoning on each sample yields a separable third-order tensor $\boldsymbol{\mathcal{T}}_{\lambda_{ex},\lambda_{em},k} = a_{\lambda_{ex}} b_{\lambda_{em}} c_k$, where $k$ is the sample index and $\mathbf{c}$ contains the relative concentrations with respect to the first sample. In other words, ideally a fluorophore gives a simple, rank-one tensor.

At this stage, the reader can understand why studying tensors models is crucial in fluorescence spectroscopy. Indeed, given a tensor $\boldsymbol{\mathcal{T}}$, if an informed user knows that this tensor is a collection of FEEM measured on solutions containing a single compound, then finding $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ means estimating, from the data tensor, the spectra and concentrations of the compound. Computing these parameters is typically achieved by solving an optimization problem, as discussed further in Section 6.5.

Usually, studied samples contain more than one chemical compound. The fluorescence phenomenon being additive at low concentrations, a tensor obtained by measuring a mixture is the sum of the separable tensors that would be obtained if each compound in the mixture was measured separately. Using a more technical vocabulary later defined in the chapter in Section 6.2, we shall see that a tensor of fluorescence spectroscopy measurements follows an approximate CPD model, with as many separable terms as there are compounds in the mixture. Again, identifying the parameters of this model can be done by solving an optimization problem, which leads to estimating the various spectra and relative concentrations in the mixture.

To summarize, we have seen that when studying tensors of fluorescence spectroscopy measurements, the data are naturally contained in a tensor that can be written as the sum of a small number of separable (i.e. rank-one) terms. I will be shown later that the CPD is a source separation technique which extracts on each mode a pre-defined number of sources from data stored in a tensor. Interestingly, these sources need not be independent in the statistical sense, as opposed to ICA for instance (see Section 1-3). In that sense, CPD and other tensor decomposition models are similar to Nonnegative Matrix Factorization, presented in Chapter 2 of this book.

### 6.1.4

### Structure of the chapter

In the remainder of this chapter, we first define more formally a few tensor decompositions (including the CPD), and introduce mild conditions that ensure the underlying sources can be extracted from a tensor, see Section 6.2. In Sections 6.3 and 6.4, we then introduce the concepts of constrained decomposition and coupled decompositions, which are partic-

ular tensor-based models inspired from the CP decomposition. actually compute tensor decompositions. Finally, in Section 6.6, we show in more depth how these methods can be employed to extract relevant information from various measurement techniques used in food sciences.

**Note** From this point onward, for simplicity reasons, we will only work with third-order tensors. However, all the models introduced after this point can trivially be extended to higher orders. Third-order tensors are convenient for notations, and are also more common than tensors of order four and higher in practice.

**Other introductions** Survey papers have been proposed in the literature, that focus on various aspects of tensors. If the reader wishes to have other accessible introductions to tensors before or after reading the remainder of this chapter, we can recommend these references: [4, 5, 6, 7].

## 6.2

## Tensor decompositions

### 6.2.1

### Tensor-based method, the matrix case

A convenient way to start addressing the tensor decomposition subject is to look first at the matrix case. Any matrix $\boldsymbol{M}$ of size $n_1 \times n_2$ can be written in the canonical basis as $\boldsymbol{M} = \sum_{ij} M_{ij} \boldsymbol{E}(i,j)$, where $\boldsymbol{E}(i,j)$ is the matrix having only one nonzero entry at position $(i,j)$. This is a trivial decomposition, which only shows that the linear space of $n_1 \times n_2$ matrices is of dimension $n_1 n_2$; it is not so much useful otherwise. In other words, this trivial element-wise decomposition has $n_1\, n_2$ parameters (one for each element in the sum), so it is not parsimonious and is seldom used for extracting information.

Now, looking for more parsimonious representations and bearing in mind the separability property emphasized in the introduction, we can decompose a matrix $\boldsymbol{M}$ into a sum of rank-one terms as $\boldsymbol{M} = \sum_{r=1}^{R} \sigma_r \boldsymbol{D}_r$, where $\boldsymbol{D}_r = \boldsymbol{u}_r \boldsymbol{v}_r^{\mathsf{T}}$ are unit-norm rank-one matrices, *i.e.*, vectors $\boldsymbol{u}_r$ and $\boldsymbol{v}_r$ are of unit-norm.

Stacking $\boldsymbol{u}_r$ and $\boldsymbol{v}_r$ vertically in matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ of sizes respectively $n_1 \times R$ and $n_2 \times R$, and setting $\boldsymbol{\Sigma}$ as a $R \times R$ diagonal matrix containing the values $\sigma_r$, this decomposition can be rewritten in compact form as $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{T}}$. The smallest number of rank-one terms $R$ such that all values $\sigma_r$ are non-zeros is called the rank of $M$ (which coincides with other usual definitions of matrix rank), and this decomposition is called a low-rank matrix approximation of $M$ if $R$ is smaller than $\min(n_1, n_2)$.

It can be checked out that such a decomposition exhibits now at most $R(n_1 + n_2 - 1)$ degrees of freedom (number of free parameters). Indeed, each rank-one component has $n_1$ parameters in $\boldsymbol{u}_r$, $n_2$ in $\boldsymbol{v}_r$ and one more is $\sigma_r$. However vectors $\boldsymbol{u}_r$ and $\boldsymbol{v}_r$ are normalized, removing one degree of freedom in each, meaning a rank-one component has $n_1 + n_2 - 1$ degree of freedom.

The problem is that this decomposition is not unique as soon as $R$ is strictly greater than one. *Uniqueness* is a key feature of any learning model as soon as the sought parameters are to be physically interpreted. For instance, say one wants to interpret parameter matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ as respectively collections of emission and excitation spectra in fluorescence spectroscopy. The existence of several solutions prohibits this interpretation, as only one of the possible solutions may actually correspond to the desired spectra.

To see this lack of uniqueness, consider any orthonormal $R \times R$ matrix $\boldsymbol{Q}$. Then we also have $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{Q}(\boldsymbol{V}\boldsymbol{Q})^\mathsf{T}$. This shows that other decompositions of the form $\boldsymbol{M} = \boldsymbol{U}'\boldsymbol{\Sigma}'\boldsymbol{V}'^\mathsf{T}$ hold true, if we define $\boldsymbol{V}' = \boldsymbol{V}\boldsymbol{Q}$, $\boldsymbol{\Sigma}'$ the diagonal matrix containing the norm of each column of $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{Q}$ and $\boldsymbol{U}' = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{Q}\Sigma'^{-1}$. Thus, for ensuring uniqueness, orthogonality between columns of matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ is generally imposed, which yields the Singular Value Decomposition (SVD):

$$M_{ij} = \sum_{r=1}^{R} \sigma_r \, U_{ir} V_{jr}, \text{ such that } \boldsymbol{U}^\mathsf{T}\boldsymbol{U} = \boldsymbol{I}_{n_1} \text{ and } \boldsymbol{V}^\mathsf{T}\boldsymbol{V} = \boldsymbol{I}_{n_2} \tag{6.4}$$

where $\boldsymbol{I}_{n_i}$ is the identity matrix of size $n_i \times n_i$. The SVD can also be written in a compact form: $\boldsymbol{M} = [\![\boldsymbol{\Sigma}; \boldsymbol{U}, \boldsymbol{V}]\!] := \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\mathsf{T}$, with left orthogonal matrices $U$ and $V$, and where $\boldsymbol{\Sigma}$ is diagonal $R \times R$ with positive entries $\Sigma_{rr} = \sigma_r$. As a notation convention, in the rest of the chapter we prefereably denote orthogonal matrices with letter $\boldsymbol{U}$ and $\boldsymbol{V}$, and general matrices with other letters.

Another convenient way to write the SVD is the following

$$\boldsymbol{M} = \sum_{r=1}^{R} \sigma_r \, \boldsymbol{D}_r, \tag{6.5}$$

where $\boldsymbol{D}_r$ are rank-one unit-norm matrices, which are orthogonal to each other. The SVD is *unique* if all singular values are distinct. Of course, rank-1 terms in the sum (6.4) or (6.5) can be permuted, because the sum is commutative. This induces a permutation among columns of matrices $\boldsymbol{U}$ and $\boldsymbol{V}$, which can be fixed by sorting singular values in decreasing order in matrix $\boldsymbol{\Sigma}$.

The number of degrees of freedom in the SVD is $R(n_1+n_2-R)$, which can be compared to the $R(n_1+n_2-1)$ degrees of freedom we would have if orthogonality was not imposed. More precisely, there are $n_1R - R(R+1)/2$ degrees of freedom in $\boldsymbol{U}$, $n_2R - R(R+1)/2$ in $\boldsymbol{V}$, and $R$ in $\boldsymbol{\Sigma}$. The sum of these 3 terms is $R(n_1 + n_2 - R)$. Indeed, it is sufficient to think of each column of matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ as normalized and orthogonal to all previous columns, meaning there are for each matrix $\sum_{r=1}^{R} r$ additionally fixed degrees of freedom. This reduced number of degrees of freedom provides intuition as

to why SVD is often unique while unconstrained low-rank matrix factorization is not. Note that imposing nonnegativity in matrix $\boldsymbol{\Sigma}$ permits to fix sign indeterminacies among columns of $\boldsymbol{U}$ and $\boldsymbol{V}$, but does not reduce the number of degrees of freedom.

### 6.2.2

### Canonical Polyadic Decomposition, PARAFAC/CanDecomp

Let us now define the Canonical Polyadic Decomposition (CPD) as an extension of SVD for higher-order tensors. In what follows, we will see that for third-order tensors, it is possible to drop the orthogonality constraint of the SVD and still obtain a unique decomposition. This makes the CPD appealing in practice for source separation [8], or for addressing other identification problems [9, 10, 11, 12].

Denote by $\boldsymbol{\mathcal{S}}$ a 3-way diagonal array with $\sigma_r$ as diagonal entries. For a large enough $R$, a $n_1 \times n_2 \times n_3$ tensor $\boldsymbol{\mathcal{T}}$ with entries $\mathcal{T}_{ijk}$ can always be decomposed as:

$$\mathcal{T}_{ijk} = \sum_{r=1}^{R} \sigma_r \, A_{ir} B_{jr} C_{kr} \tag{6.6}$$

where factor matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ have unit-norm columns, and where weights $\sigma_r$ may be imposed to be real positive. Indeed, just like with matrices, one may write

$$\boldsymbol{\mathcal{T}} = \sum_{r_1,r_2,r_3}^{n_1,n_2,n_3} \mathcal{T}_{r_1,r_2,r_3} \boldsymbol{\mathcal{E}}(r_1, r_2, r_3) \tag{6.7}$$

where $\mathcal{E}(r_1, r_2, r_3)_{ijk} = \delta_{i,r_1}\delta_{j,r_2}\delta_{k,r_3}$ is a tensor with zeros everywhere except a one in position $(r_1, r_2, r_3)$, and $\delta_{i,r}$ is the Kronecker symbol. Then setting $\boldsymbol{r} = (r_1, r_2, r_3)$ as a super-index, clearly any tensor $\boldsymbol{\mathcal{T}}$ admits a decomposition in separable terms (6.6) as soon as $R$ is large enough. A first naive upper bound on $R$ is therefore $n_1 n_2 n_3$. On the other hand, since the above decomposition always exists for large enough $R$, there always exists a *minimal value* of $R$, which is called the *rank* of tensor $\boldsymbol{\mathcal{T}}$. It will be denoted $\mathrm{rank}\{\boldsymbol{\mathcal{T}}\}$, as for matrices. Moreover, decomposition (6.6) with minimal number of terms $R$ is the exact *Canonical Polyadic Decomposition*(CPD) of tensor $\boldsymbol{\mathcal{T}}$. The exact CPD is therefore the decomposition of a tensor $\boldsymbol{\mathcal{T}}$ into a minimal sum of rank-one tensors. A graphical representation of the CPD is given in Figure 6.3. This can also be conveniently written as:

$$\boldsymbol{\mathcal{T}} = \sum_{r=1}^{R} \sigma_r \, \boldsymbol{\mathcal{D}}_r \tag{6.8}$$

with $\boldsymbol{\mathcal{D}}_r = \boldsymbol{a}_r \otimes \boldsymbol{b}_r \otimes \boldsymbol{c}_r$, where $\boldsymbol{a}_r$ (resp. $\boldsymbol{b}_r$ and $\boldsymbol{c}_r$) denote the unit-norm columns of matrix factor $\boldsymbol{A}$ (resp. $\boldsymbol{B}$ and $\boldsymbol{C}$), and $\otimes$ denotes the outer-product defined as

$$[\boldsymbol{a} \otimes \boldsymbol{b} \otimes \boldsymbol{c}]_{ijk} = a_i b_j c_k. \tag{6.9}$$

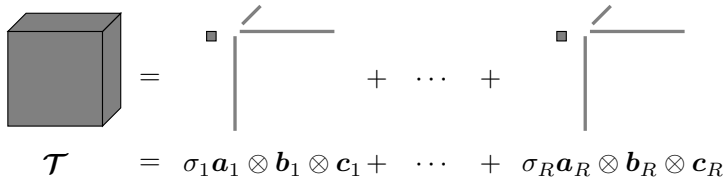



**Figure 6.3** A graphical representation of the CPD. A tensor $\boldsymbol{\mathcal{T}}$ is expressed as a minimal sum of $R$ rank-one tensors, which can themselves be expressed as outer products $\sigma_r \boldsymbol{a}_r \otimes \boldsymbol{b}_r \otimes \boldsymbol{c}_r$.

The notation used is summarized at the beginning of this book. As for the matrix SVD, we can decide to sort values $\sigma_r$ in decreasing order to fix the permutation ambiguity stemming from addition commutativity; note the similarity with (6.4).

A necessary condition for this decomposition to be unique, up to permutations and signs ambiguities, is that the number of degrees of freedom in the left hand side of equation (6.6) be at least as large as that in the right hand side. In general there are $n_1 n_2 n_3$ elements in tensor $\boldsymbol{\mathcal{T}}$, and $R(n_1 + n_2 + n_3 + 1)$ parameters in the CPD with $3R$ normalization constraints. In other words, we must have:

$$n_1 n_2 n_3 \geq R(n_1 + n_2 + n_3 - 2) \tag{6.10}$$

As soon as $R < \frac{n_1 n_2 n_3}{n_1+n_2+n_3-2}$, this necessary condition holds, and it becomes possible for the CPD to be unique. We shall subsequently see that this condition is not sufficient to ensure CPD uniqueness, but that for small enough $R$, the orthogonality constraint is generally not required to obtain a unique decomposition. This is unlike the matrix case, where uniqueness cannot be attained without additional constraints except when $R = 1$. In addition, condition (6.10) may hold true even if $R$ exceeds $\min\{n_1, n_2, n_3\}$, which is not possible under orthogonality constraints. The bound $R \leq n_1 n_2 n_3/(n_1+n_2+n_3-2)$, induced by the counting (6.10) of degrees of freedom, is studied in [13].

Strangely enough, condition (6.10) is not sufficient to guarantee uniqueness of the CPD. It has been shown that the CPD (6.8) is unique[1)] provided the rank is not too large [14, 15, 16]. In particular, uniqueness is ensured if:

$$R \leq \frac{1}{2}(\text{krank}\{\boldsymbol{A}\} + \text{krank}\{\boldsymbol{B}\} + \text{krank}\{\boldsymbol{C}\} - 2) \tag{6.11}$$

In the *sufficient condition* above, $\text{krank}\{\boldsymbol{A}\}$ denotes Kruskal's rank[2)] of matrix $\boldsymbol{A}$. Uniqueness of the CPD (6.8) can be ensured under conditions weaker than (6.11); see the recent paper [17] and references therein.

Equation (6.6) can be seen in many ways, among which two are common for practical use.

1) again, if we refer to (6.6) instead of (6.8), uniqueness is to be understood up to permutation of terms in the sum; we made the same observation for matrices in Section 6.2.1 about Eq. (6.5).
2) The Kruskal rank of a matrix is the largest number $\kappa$ such that any subset of $\kappa$ columns is full rank. Hence Kruskal's rank cannot exceed rank. For almost all matrices, Kruskal's rank is equal to rank.

- First, it is a tensor factorization model that seeks a small number $R$ of separable patterns to describe the data. This point of view is usually used for justifying the use of tensor decomposition techniques in machine learning as an exploratory technique.
- Second, it is a parameter identification technique where physically meaningful matrices $\boldsymbol{A}, \boldsymbol{B}$ and $\boldsymbol{C}$ are of interest for a further task. In other words, the CPD may be seen as a source separation technique. For instance, matrix $\boldsymbol{A}$ may stand for a spectral signature that enables the identification of the chemical compounds present in a mixture. This second approach contrasts with the first one since uniqueness here is a key feature for interpreting the results. It is mostly seen in bio-medical applications (metabolomics, neuroimaging) or in sensor arrays where the CP decomposition results from a set of physical equations.

The model is particularly useful when it coincides with the physical model of the data, because it provides meaningful solutions due to the uniqueness property.

At this stage, it is worth saying a word about terminology. Even if the CP decomposition has been introduced originally in 1927 by Hitchcock, it has been rediscovered in 1970, by Harshman and Carroll and Chang. They gave it the name of PARAFAC and CANDECOMP, respectively. To unify the terminology, Kiers [18] proposed the acronym CP, which can wisely stand for "CANDECOMP/PARAFAC", as well as for "Canonical Polyadic". In the sequel we shall refer to (6.6) and (6.8) as the exact *CP Decomposition* (CPD) when $R$ is indeed minimal and reveals *tensor rank*. If we assume a multilinear model without column normalization, then an additional scaling indeterminacy appears:

$$\mathcal{T}_{ijk} = \sum_{r=1}^{R} A_{ir} B_{jr} C_{kr}, \quad \text{denoted by} \quad \boldsymbol{\mathcal{T}} = [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!] \tag{6.12}$$

Such a decomposition without unit-norm constraint is not unique, and contains $2R$ free parameters (scaling factors). More precisely, without normalization constraints, the CPD model features $(n_1 + n_2 + n_3)R$ parameters since each rank-one component involves three vectors of sizes $n_1, n_2$ and $n_3$, and the norms of these vectors may be pulled apart without modifying the sum of rank-one terms, since

$$\forall \mu, \lambda, \nu \in \mathbb{R}, \ \mu \boldsymbol{a} \otimes \lambda \boldsymbol{b} \otimes \nu \boldsymbol{c} = \mu\lambda\nu \, (\boldsymbol{a} \otimes \boldsymbol{b} \otimes \boldsymbol{c}) \ . \tag{6.13}$$

Therefore, only $(n_1 + n_2 + n_3 - 2)R$ parameters can possibly be identified. More bibliographical pointers to the CPD may be found in [19, 4].

### 6.2.3

### Manipulation of tensors

In the following we introduce some commonly used manipulations of tensors, namely how to unfold them into matrices or vectors. These manipulations are useful in many different context, from the derivation of algorithms using linear algebra to programming.

We also explain how these unfoldings link the outer product with the Kronecker product of matrices. The Kronecker product of two matrices $\boldsymbol{A} \in \mathbb{R}^{n_1 \times n_2}$ and $\boldsymbol{B} \in \mathbb{R}^{m_1 \times m_2}$ is denoted by $\boldsymbol{A} \boxtimes \boldsymbol{B} \in \mathbb{R}^{n_1 m_1 \times n_2 m_2}$ and is defined by:

$$\boldsymbol{A} \boxtimes \boldsymbol{B} := \left[\begin{array}{c|c|c|c} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \dots & a_{1n_2}\boldsymbol{B} \\ \hline a_{21}\boldsymbol{B} & & & \\ \hline \vdots & & & \\ \hline a_{n_1 1}\boldsymbol{B} & & & a_{n_1 n_2}\boldsymbol{B} \end{array}\right]. \tag{6.14}$$

**Vectorization** There are *a priori* a combinatorial number of ways to arbitrarily transform a tensor in $\mathbb{R}^{n_1 \times n_2 \times n_3}$ into a vector in $\mathbb{R}^{n_1 n_2 n_3}$. We refer to such an operator as a vectorization. However, it is quite clear that an arbitrary, pseudo-random vectorization will destroy any nice structure that the input tensors might have. In particular, if $\boldsymbol{\mathcal{T}}$ follows a CPD, it is a reasonable ordeal to ask for the vectorized version of it to also satisfy a similar equation.

Taking this into consideration, there are still a few different ways to define a vectorization, and several co-exist in the literature. We propose to use the row-wise vectorization that exhibits a nice and simple link between the outer product and the Kronecker product as shown below. Let $\boldsymbol{\mathcal{T}}$ be a $n_1 \times n_2 \times n_3$ tensor, we define the row-wise vectorization as follows:

$$\text{vec}(\boldsymbol{\mathcal{T}})_{(i-1)n_3 n_2 + (j-1)n_3 + k} = T_{ijk} \tag{6.15}$$

See Figure 6.4 for a graphical explanation of these formulas.

Note that the above definition of the vectorization operator may not coincide with native implementation in some languages such as MATLAB, which is column-wise, but does coincide with the memory layout of other languages such as C. More details can be found in [20].

With the above definitions, we have in particular the property:

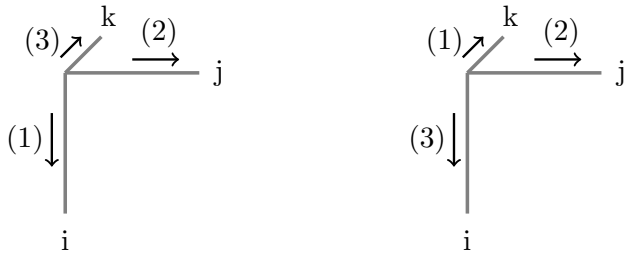
$$\text{vec}(\boldsymbol{a} \otimes \boldsymbol{b} \otimes \boldsymbol{c}) = \boldsymbol{a} \boxtimes \boldsymbol{b} \boxtimes \boldsymbol{c} \tag{6.16}$$

which is not enjoyed by most other definitions, where terms need to be permuted. Because the vectorization operation is trivially linear, we have further that

$$\text{vec}\left(\sum_{r=1}^{R} \sigma_r \boldsymbol{a}_r \otimes \boldsymbol{b}_r \otimes \boldsymbol{c}_r\right) = \sum_{r=1}^{R} \sigma_r \boldsymbol{a}_r \boxtimes \boldsymbol{b}_r \boxtimes \boldsymbol{c}_r \tag{6.17}$$

which nicely transposes the CPD to a Kronecker equation.

Furthermore, this Kronecker equation can itself be written in a more compact format, making use of the Khatri-Rao product. The Khatri-Rao product between two matrices $\boldsymbol{A} = [\boldsymbol{a}_1, \dots, \boldsymbol{a}_R]$ and $\boldsymbol{B} = [\boldsymbol{b}_1, \dots, \boldsymbol{b}_R]$ is nothing else than a column-wise Kronecker




a) columnwise vectorization b) suggested vectorization

**Figure 6.4** The suggested row-wise vectorization reads the entries of the tensor along the last index in the lexicographic order. This contrasts with the column-wise vectorization which is also often encountered.

product:

$$\boldsymbol{A} \odot \boldsymbol{B} = [\boldsymbol{a}_1 \boxtimes \boldsymbol{b}_1 \mid \ldots \mid \boldsymbol{a}_R \boxtimes \boldsymbol{b}_R] = \left[\begin{array}{c|c|c} A_{11}\boldsymbol{b}_1 & \ldots & A_{1R}\boldsymbol{b}_R \\ \hline \vdots & \ldots & \vdots \\ \hline A_{n_1 1}\boldsymbol{b}_1 & \ldots & A_{n_1 R}\boldsymbol{b}_R \end{array}\right] \tag{6.18}$$

Then it is easy to check that

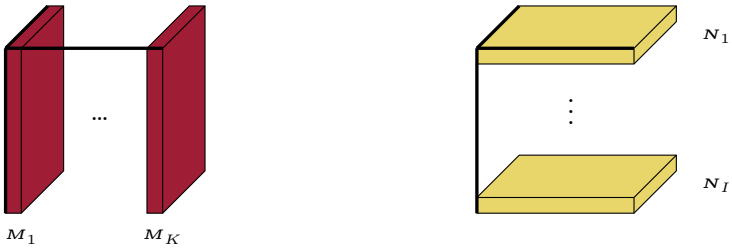
$$\operatorname{vec}\left(\sum_{r=1}^{R} \sigma_r \boldsymbol{a}_r \otimes \boldsymbol{b}_r \otimes \boldsymbol{c}_r\right) = (A \odot B \odot C)\, \boldsymbol{s} \tag{6.19}$$

with $\boldsymbol{s}$ the vector of length $R$ containing all values $\sigma_r$.

**Matricization** Similarly to vectorization, to be able to resort to known results borrowed from linear algebra, we shall sometimes need to store the elements of three-way arrays into two-way arrays (i.e. matrices). This can be done in various manners, but we shall retain three of them, namely those having as number of rows one dimension of the original tensor. The operation transforming a $d$-way array into a matrix is known as matrix unfolding, matrix flattening, or matricization [7, 21, 20]. If $\boldsymbol{\mathcal{T}}$ is a tensor of dimensions $I \times J \times K$, we shall use the following matrix unfoldings $\boldsymbol{T}^{(p)}$ along mode $p$ defined as

$$\begin{aligned} \boldsymbol{T}^{(1)} \text{ is } n_1 \times n_2 n_3 : &\quad \mathcal{T}^{(1)}_{in} = \mathcal{T}_{ijk}, \text{ with } n = k + (j-1)n_2 \\ \boldsymbol{T}^{(2)} \text{ is } n_2 \times n_3 n_1 : &\quad \mathcal{T}^{(2)}_{jp} = \mathcal{T}_{ijk}, \text{ with } p = k + (i-1)n_3 \\ \boldsymbol{T}^{(3)} \text{ is } n_3 \times n_1 n_2 : &\quad \mathcal{T}^{(3)}_{kq} = \mathcal{T}_{ijk}, \text{ with } q = j + (i-1)n_1 \end{aligned} \tag{6.20}$$

These unfoldings are illustrated in Figure 6.5. Matricizations along mode $p$ fold the $p$th dimension of the tensor on the rows of the matricized tensor. This maps the range of the tensor on the $p$th dimension to the column space of the matricized tensor. Again, similarly




$$\begin{aligned}\boldsymbol{T}^{(1)} &= [\boldsymbol{M}_1|\ldots|\boldsymbol{M}_K]\\ \boldsymbol{T}^{(2)} &= [\boldsymbol{N}_1|\ldots|\boldsymbol{N}_I]\\ \boldsymbol{T}^{(3)} &= [\boldsymbol{N}_1^T|\ldots|\boldsymbol{N}_I^T]\end{aligned}$$

**Figure 6.5** Three unfoldings of tensor $\boldsymbol{\mathcal{T}}$

to the vectorization, tensors are matricized by selecting entries along the deepest index first. This way, matricizations and vectorizations do not permute the terms from tensor produts to Kronecker products. Indeed, it holds that

$$\begin{aligned}[\boldsymbol{a}\otimes\boldsymbol{b}\otimes\boldsymbol{c}]^{(1)} &= \boldsymbol{a}\otimes\boldsymbol{b}\boxtimes\boldsymbol{c}\\ [\boldsymbol{a}\otimes\boldsymbol{b}\otimes\boldsymbol{c}]^{(2)} &= \boldsymbol{b}\otimes\boldsymbol{a}\boxtimes\boldsymbol{c}\\ [\boldsymbol{a}\otimes\boldsymbol{b}\otimes\boldsymbol{c}]^{(3)} &= \boldsymbol{c}\otimes\boldsymbol{a}\boxtimes\boldsymbol{b}\end{aligned} \tag{6.21}$$

In terms of matrix unfoldings defined in (6.20), using (6.21) and the linearity of the matricization, the CPD can be written in three different ways:

$$\begin{aligned}\boldsymbol{T}^{(1)} &= \boldsymbol{U}\boldsymbol{S}^{(1)}(\boldsymbol{V}\boxtimes\boldsymbol{W})^{\mathsf{T}} = \boldsymbol{U}\,\mathrm{diag}\{\boldsymbol{s}\}(\boldsymbol{V}\odot\boldsymbol{W})^{\mathsf{T}}\\ \boldsymbol{T}^{(2)} &= \boldsymbol{V}\boldsymbol{S}^{(2)}(\boldsymbol{U}\boxtimes\boldsymbol{W})^{\mathsf{T}} = \boldsymbol{V}\,\mathrm{diag}\{\boldsymbol{s}\}(\boldsymbol{U}\odot\boldsymbol{W})^{\mathsf{T}}\\ \boldsymbol{T}^{(3)} &= \boldsymbol{W}\boldsymbol{S}^{(3)}(\boldsymbol{U}\boxtimes\boldsymbol{V})^{\mathsf{T}} = \boldsymbol{W}\,\mathrm{diag}\{\boldsymbol{s}\}(\boldsymbol{U}\odot\boldsymbol{V})^{\mathsf{T}}\end{aligned} \tag{6.22}$$

As a side note, there exist several definitions of tensor-matrix bijective maps in the literature. What is important is to define the inverse map and related properties consistently.

#### 6.2.3.1 Contractions and CPD

It is convenient to have at our disposal a compact notation to indicate summations on several indices. We shall assume the notation proposed in [7]:

$$\boldsymbol{\mathcal{T}} = [\![\boldsymbol{\mathcal{G}};\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}]\!], \tag{6.23}$$

meaning just that $\mathcal{T}_{ijk} = \sum_{\ell mn} A_{i\ell}B_{jm}C_{kn}\,\mathcal{G}_{\ell mn}$. Note that another notation has been proposed in [22] and would be equally meaningful: $\boldsymbol{\mathcal{T}} = (\boldsymbol{A},\boldsymbol{B},\boldsymbol{C})\cdot\boldsymbol{\mathcal{G}}$. Some authors

also write (6.23) as $\boldsymbol{\mathcal{T}} = (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C}) \cdot \boldsymbol{\mathcal{G}}$, which accounts for the fact that $\boldsymbol{\mathcal{T}}$ is the image of $\boldsymbol{\mathcal{G}}$ by the multilinear operator defined by $\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C}$ [20]. In the remainder, only notation (6.23) will be used.

In particular, if we have that $\mathcal{T}_{ijk} = \sum_r A_{ir} B_{jr} C_{kr}$, then this could be denoted as $\boldsymbol{\mathcal{T}} = [\![\boldsymbol{\mathcal{I}}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]$, where $\boldsymbol{\mathcal{I}}$ is a diagonal 3-way array with ones on its diagonal. In such a case, one can omit tensor $\boldsymbol{\mathcal{I}}$ and just write:

$$\boldsymbol{\mathcal{T}} = [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]. \tag{6.24}$$

Note that the CPD (6.6) can thus be written as $[\![\boldsymbol{\mathcal{S}}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]$, where $\boldsymbol{\mathcal{S}}$ here denotes a diagonal tensor whose entry $(q, q, q)$ equals $\sigma_q$. Moreover, by setting $\boldsymbol{A}' = [\sigma_1 \boldsymbol{a}_1, \ldots, \sigma_R \boldsymbol{a}_R]$, it holds that

$$[\![\boldsymbol{A}', \boldsymbol{B}, \boldsymbol{C}]\!] = [\![\boldsymbol{\mathcal{S}}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]. \tag{6.25}$$

In other words, the CPD may be written in several equivalent notations. In this chapter, we will use whichever format is more convenient in each situation.

### 6.2.4

### The chain rule

There exist a useful algebra result making use of the compact contraction notation defined in Section 6.2.3.1. Although it can be proven almost trivially using more general results from tensor algebra, we shall formulate it and prove it in simple terms.

PROPERTY 6.1 (CHAIN RULE) *Given matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{A}'$, $\boldsymbol{B}'$ and $\boldsymbol{C}'$ that are compatible for products pairwise, it holds that*

$$[\![[\![\boldsymbol{\mathcal{T}}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]; \boldsymbol{A}', \boldsymbol{B}', \boldsymbol{C}']\!] = [\![\boldsymbol{\mathcal{T}}; \boldsymbol{A}'\boldsymbol{A}, \boldsymbol{B}'\boldsymbol{B}, \boldsymbol{C}'\boldsymbol{C}]\!] \tag{6.26}$$

We coin this property the chain rule. To prove this result it is sufficient to show that for any mode, here arbitrarily the first one,

$$[\![[\![\boldsymbol{\mathcal{T}}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]; \boldsymbol{A}', \boldsymbol{I}, \boldsymbol{I}]\!] = [\![\boldsymbol{\mathcal{T}}; \boldsymbol{A}'\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!] \tag{6.27}$$

and then apply this partial result sequentially. This partial result is obtained by observing that

$$\begin{aligned} [\![[\![\boldsymbol{\mathcal{T}}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]; \boldsymbol{A}', \boldsymbol{I}, \boldsymbol{I}]\!]_{ijk} &= \textstyle\sum_{l'} A'_{il'} \sum_{lmn} A_{l'l} B_{jm} C_{kn} \mathcal{T}_{lmn} \\ &= \textstyle\sum_{lmn} \left(\sum_{l'} A'_{il'} A_{l'l}\right) B_{jm} C_{kn} \mathcal{T}_{lmn} \\ &= [\![\boldsymbol{\mathcal{T}}; \boldsymbol{A}'\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]_{ijk} \end{aligned} \tag{6.28}$$

and since this proof clearly holds for modes 2 and 3 as well by symmetry, we have just proved Property 6.1.

The chain rule has an important corrolary that we will use throughout the chapter:

PROPERTY 6.2 *Given invertible matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$, and two tensors $\boldsymbol{\mathcal{T}}$ and $\boldsymbol{\mathcal{G}}$, if the dimensions are compatible, then*

$$[\![\boldsymbol{\mathcal{T}}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!] = \boldsymbol{\mathcal{G}} \quad \equiv \quad \boldsymbol{\mathcal{T}} = [\![\boldsymbol{\mathcal{G}}; \boldsymbol{A}^{-1}, \boldsymbol{B}^{-1}, \boldsymbol{C}^{-1}]\!] \tag{6.29}$$

### 6.2.5 Multilinear Singular Value Decomposition

A second possibility to extend SVD to tensors is to keep orthogonality of factor matrices. In that case, a $n_1 \times n_2 \times n_3$ tensor $\boldsymbol{\mathcal{T}}$ is decomposed as:

$$\boldsymbol{\mathcal{T}} = [\![\boldsymbol{\mathcal{G}}; \boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}]\!] \tag{6.30}$$

where the so-called core tensor $\boldsymbol{\mathcal{G}}$ is of size $R_1 \times R_2 \times R_3$, smaller than $\boldsymbol{\mathcal{T}}$, that is: $R_1 \leq n_1$, $R_2 \leq n_2$, $R_3 \leq n_3$, and factor matrices $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{W}$ have orthogonal unit-norm columns. This is interesting only if at least one dimension is strictly smaller, or when the core is sparser, which means that a compression[3] has been performed. Again, it is then clear by just counting degrees of freedom that the diagonal form generally cannot be imposed[4] in $\boldsymbol{\mathcal{G}}$.

In terms of matrix unfoldings defined in (6.20), the multilinear SVD can be written in three different ways:

$$\begin{aligned} \boldsymbol{T}^{(1)} &= \boldsymbol{U}\boldsymbol{G}^{(1)}(\boldsymbol{V} \otimes \boldsymbol{W})^{\mathsf{T}} \\ \boldsymbol{T}^{(2)} &= \boldsymbol{V}\boldsymbol{G}^{(2)}(\boldsymbol{U} \otimes \boldsymbol{W})^{\mathsf{T}} \\ \boldsymbol{T}^{(3)} &= \boldsymbol{W}\boldsymbol{G}^{(3)}(\boldsymbol{U} \otimes \boldsymbol{V})^{\mathsf{T}} \end{aligned} \tag{6.31}$$

Indeed, it can be noticed that the multilinear SVD is nothing more than another decomposition of $\boldsymbol{\mathcal{T}}$ into separable terms:

$$\boldsymbol{\mathcal{T}} = \sum_{lmn}^{R_1, R_2, R_3} \mathcal{G}_{lmn} \boldsymbol{u}_l \otimes \boldsymbol{v}_m \otimes \boldsymbol{w}_n \tag{6.32}$$

with $\boldsymbol{u}_l, \boldsymbol{v}_m, \boldsymbol{w}_n$ respectively the columns of matrices $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}$. Therefore the unfoldings formulas are obtained from (6.21) by linearity. These three writings show that matrices $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{W}$ are built with the left singular vectors of matrices $\boldsymbol{T}^{(1)}$, $\boldsymbol{T}^{(2)}$ and $\boldsymbol{T}^{(3)}$ respectively. They can hence be computed by matrix SVDs. Once they are known, the core tensor can in turn be computed as

$$\boldsymbol{\mathcal{G}} = [\![\boldsymbol{\mathcal{T}}; \boldsymbol{U}^{\mathsf{T}}, \boldsymbol{V}^{\mathsf{T}}, \boldsymbol{W}^{\mathsf{T}}]\!] \tag{6.33}$$

3) For the moment, this compression is lossless. Lossy compression will be addressed in Section 6.5.3.1.

4) In fact, tensors that are orthogonally diagonalizable form a very small class, and their rank must be bounded by all their dimensions.

using (6.29) of Property 6.2. One defines the *multilinear rank* as the triplet of minimal values $(R_1, R_2, R_3)$ such that (6.30) holds exactly. Then it can be shown that

$$\max\{R_1, R_2, R_3\} \leq \text{rank}\{\boldsymbol{\mathcal{T}}\} \leq \min\{R_1R_2, R_2R_3, R_3R_1\} \tag{6.34}$$

Because the multilinear SVD is computed with the help of three SVDs, it enjoys the same uniqueness conditions. The multilinear SVD has been first suggested by Kroonenberg in 1980 [23], and further studied in 2000 by De Lathauwer in [21] under the name of High-Order SVD (HOSVD). But the premises of multilinear SVD appeared earlier with the Tucker3 decomposition, which we address now.

### 6.2.6 Tucker

Tucker proposed much earlier[24], in 1966, a multilinear decomposition similar to (6.30) but without orthogonality constraints on factor matrices:

$$\boldsymbol{\mathcal{T}} = [\![\boldsymbol{\mathcal{G}}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!] \tag{6.35}$$

The consequence of relaxing all constraints is that this decomposition – often referred to as Tucker3 – is not unique anymore, even if the size of the core is the same. The 3 in Tucker3 refers to the number of modes that are subspaced in the decomposition. The Tucker3 decomposition is also often called the Tucker format, by opposition to Tucker decomposition, because of the uniqueness issue [2]. Tucker3 decomposition formally encompasses both multilinear SVD and CPD, which appear as constrained versions. Other constraints such as nonnegativity or sparsity could be thought of and would yield other decompositions, see Section 6.3.

The particular case when one mode is not reduced in dimension, which amounts to fixing the factor on that mode to the identity matrix, say $\boldsymbol{C} = \boldsymbol{I}$, is sometimes of interest and has received the name of Tucker2 (since now strict subspaces are defined in only 2 modes). It can be denoted $\boldsymbol{\mathcal{T}} = [\![\boldsymbol{\mathcal{G}}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{I}]\!]$.

### 6.2.7 PARAFAC2

PARAFAC2 has been introduced in [25, 26]. It differs from the CPD (6.8) by the fact that a matrix factor, *e.g.* the first one, may not be the same for each matrix slice. Rather, these first-mode factors are related by an orthogonal transform:

$$\boldsymbol{\mathcal{T}} = [\![\boldsymbol{A}(k), \boldsymbol{B}, \boldsymbol{C}]\!], \quad \boldsymbol{A}(k) = \boldsymbol{P}(k)\boldsymbol{H}, \; \boldsymbol{P}(k)^{\mathsf{T}}\boldsymbol{P}(k) = \boldsymbol{I}, \tag{6.36}$$

with a slight abuse of notation. In other words, if the data tensor is preprocessed as follows

$$\forall k \leq n_1, \; \boldsymbol{\mathcal{T}}(k,:,:) \leftarrow \boldsymbol{P}(k)^T\boldsymbol{\mathcal{T}}(k,:,:) \tag{6.37}$$

then it admits the CP decomposition $[\![\boldsymbol{H}, \boldsymbol{B}, \boldsymbol{C}]\!]$. Again, this means that PARAFAC2 is not a sum of separable terms, but becomes one after a linear transformation of each slice in the tensor. More details on the Parafac2 decomposition, from the coupled decomposition perspective, are given in Block 6.4.2.

### 6.2.8

### Approximate decomposition

Up to the previous section, we have talked about *exact* decompositions. For instance, any tensor can be decomposed exactly as in (6.8). However, an exact representation is generally not suitable. Indeed, decomposition (6.8) is unique only if the tensor rank is not too large. On the other hand, (6.8) is exactly verified as long as the rank is large enough, but uniqueness may not be guaranteed. In fact, in the presence of noise, the minimal rank for (6.8) to hold may be larger than the Kruskal upper bound [27]. In addition, the underlying physical model is generally of interest only for a reasonably small value of $R$, that for instance may stand for the number of unknown sources in a source separation problem. For these reasons, a low-rank approximation is needed.

Instead of computing the exact CPD (6.6), a first idea is to minimize the objective

$$\Upsilon(\boldsymbol{\mathcal{S}}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}) = \|\boldsymbol{\mathcal{T}} - \sum_{r=1}^{R} \sigma(r)\ \boldsymbol{a}(r) \otimes \boldsymbol{b}(r) \otimes \boldsymbol{c}(r)\|_F^2 \tag{6.38}$$

for a fixed[5)] value of $R$, supposed to be smaller than the rank of $\boldsymbol{\mathcal{T}}$. Also, define the Frobenius norm for tensors as $\|T\|_F^2 = \sum_{i,j,k=1}^{n_1,n_2,n_3} T_{ijk}^2$.

Unfortunately, the low-rank tensor approximation problem is generally ill-posed for tensors (if $R > 1$ and $d > 2$) [22, 28]. More precisely, a minimizer of (6.38) may not exist. This is in contrast with matrices, for which a low-rank approximation can be easily computed by truncating the SVD [29]. Some solutions for this are discussed in the Section 6.3 relative to constrained decompositions. However, in practice, this fact is often overlooked. Section 6.5 details how to compute an approximate CPD by tentatively solving optimization problems similar to minimizing (6.38).

5) The problem of chosing the appropriate rank to obtain a meaningful approximation is application-dependent and typically very intricate. As detailed in Section 6.5.1.2, deflation strategies should not be employed in the general case.

## 6.3

## Constraints in decompositions

Although tensor decomposition techniques have proven useful in a wide range of applications, they are seldom used as a black box model. Rather, problem-specific constraints are often applied on the factors of the decomposition. There are several main reasons to apply constraints to a tensor decomposition model [19]:

- Despite the identifiability properties of tensor decompositions, the computed parameters may not fulfill key properties of the sought factors, therefore hindering interpretability of the results. Imposing constraints, for instance non-negativity, may ensure interpretable results are obtained.
- When the parameters of a tensor decomposition model are not identifiable, constraints can restore identifiability e.g. by reducing the size of the search space.
- The underlying optimization problem of low-rank approximations can be shown to have a solution in the presence of constraints, while it may not in the general case as explained in Section 6.2.8.
- Estimation performance is increased in a noisy scenario.

Below, some of the most widely used constrained tensor decomposition models are introduced. Many constrained models are however not discussed here, since their derivation is either straightforward or of relatively lesser importance in source separation applications.

Among others, the following constraints have been explored in the literature: nonnegativity [30, 31, 32], orthogonality [33, 34, 35, 36, 31, 37], smoothness [38, 39, 40, 41], unimodality [30], simplex or sum to one [19, 42], dictionary and sparsity [43, 44, 45, 46], coherence constraints to ensure existence of approximation [47]. Some of these constraints will be developed in subsequent paragraphs.

### 6.3.1

### Non-negativity

When dealing with data acquired by measuring physical properties of natural processes, such as fluorescence or reflectance measurements, one of the most widely encountered a priori information available on the model parameters is nonnegativity. Indeed, parameters related to many types of spectra or concentrations must be nonnegative by definition, and cannot be easily interpreted if they are partially negative.

Of course, nonnegativity constraints may be applied in any tensor decomposition model, but in this section we focus on the CPD, which is by far the most studied nonnegative tensor decomposition model. At the end of this section, we discuss briefly the nonnegative Tucker decomposition.

#### 6.3.1.1 Nonnegative CPD

Formally, a nonnegative CPD is derived as follows:

$$\boldsymbol{\mathcal{T}} = [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!] \quad \text{and} \quad \boldsymbol{A} \succeq 0, \boldsymbol{B} \succeq 0, \boldsymbol{C} \succeq 0 \tag{6.39}$$

where the inequality signs $\succeq$ are to be understood entry-wise. Clearly, without noise or modeling error, nonnegative factors imply nonnegative tensor data[6] $\boldsymbol{\mathcal{T}}$, but a nonnegative tensor could be written as a CPD model with negative entries in factor matrices. Therefore, it may occur that the rank of a tensor, which is the minimal number of columns in unconstrained factor matrices, is strictly smaller than the *nonnegative rank*, which is the minimal number of columns in nonnegative factor matrices and will be denoted $\text{rank}_+\{T\}$. A simple example of discrepancy[7] between rank and nonnegative rank is obtained by considering the following tensor (written slice-wise):

$$\boldsymbol{\mathcal{T}} = \left[\begin{array}{cc|cc} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{array}\right] \tag{6.40}$$

which has rank 2, but nonnegative rank 3. Indeed,

$$\boldsymbol{\mathcal{T}} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ -1 \end{bmatrix} \tag{6.41}$$

shows that $\text{rank}\{\boldsymbol{\mathcal{T}}\} \leq 2$ (while it is clear that $\text{rank}\{\boldsymbol{\mathcal{T}}\} > 1$). However it can be shown[8] that there exists no way to write $\boldsymbol{\mathcal{T}}$ as the sum of two rank-one tensors with nonnegative entries so that $\text{rank}\{\boldsymbol{\mathcal{T}}\}_+ > 2$, and

$$\boldsymbol{\mathcal{T}} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{6.42}$$

is a rank 3 nonnegative CPD of $\boldsymbol{\mathcal{T}}$.

Therefore, although the parameters of the PARAFAC model are identifiable, adding nonnegativity constraints may change the solution to the decomposition problem entirely. It has been shown recently however that in a generic case, *i.e.* choosing a nonnegative tensor at random, nonnegative rank and the usual tensor rank match [48].

As stated earlier, nonnegativity constraints are also important to make the approximation problem well posed. Indeed, nonnegativity constraints prevent components cancellation

6) In practice negative entries may appear in these tensors because of measurement error, which does not in principle prevent from fitting an approximate nonnegative CPD.

7) Note that this discrepancy also exists for nonnegative matrices [4]: nonnegative rank can be strictly larger than rank. Nonnegative matrices are used in Chapter 2.

8) In a nutshell, any rank-one term in a nonnegative decomposition of $\boldsymbol{\mathcal{T}}$ must have a zero in the second entry of one of the vector component, e.g. $A_{2r} = 0$. However the second row, column and fiber of $\boldsymbol{\mathcal{T}}$ are nonzero (they are all equal to $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$). It can be observed that at least three rank-one terms are required to place these nonzero elements.

referred to as "degeneracy" [49, 50, 51, 52] by bounding the set of admissible parameters [31]. More precisely, under nonnegativity constraints, the cost function (6.38) becomes coercive[9] and one can define a compact ball, possibly very large, within which the cost is upper-bounded. Because this cost is continuous in all parameters, it must reach its minimum value within that ball. In contrast, without nonnegativity constraints, rank-one components can cancel out while growing to infinity so that such a compact ball may not exist.

In addition, nonnegativity can guarantee uniqueness of the best low-rank approximation [32], and even the uniqueness of the CPD of the best low nonnegative rank approximate [48]. For these reasons, nonnegativity should be imposed each time it has a physical justification [31, 41, 53, 54, 55, 56, 57].

#### 6.3.1.2 **Nonnegative Tucker decomposition**

Nonnegativity constraints have also been extensively used along with the Tucker model introduced in Section 6.2.6:

$$\boldsymbol{\mathcal{T}} = [\![\boldsymbol{\mathcal{G}}; \boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}]\!] \quad \text{and} \quad \boldsymbol{U} \succeq 0, \boldsymbol{V} \succeq 0, \boldsymbol{W} \succeq 0, \boldsymbol{\mathcal{G}} \succeq 0. \tag{6.43}$$

Similarly to nonnegative matrix factorization, adding nonnegativity constraints in the Tucker model, one hoped to obtain a uniquely defined tensor decomposition [58].

The field of applying constraints on Tucker models was pioneered by Smilde and Kiers in a series of papers [59, 60, 61]. They realized that it was mostly necessary to add several constraints such as nonnegativity, forcing core elements to zero in order to obtain identified models.

But in fact, little is known on that topic. It has been shown that in the restrictive case when the dimensions $R_1, R_2, R_3$ in the nonnegative Tucker decomposition match the nonnegative ranks of the factors $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}$ *and* the nonnegative ranks of the unfoldings, the uniqueness of the nonnegative Tucker model is equivalent to the uniqueness of three nonnegative matrix factorizations of each unfolding of the tensor, which is a difficult condition to satisfy [62]. Therefore, the nonnegative Tucker decomposition does not ship with powerful uniqueness properties. Nonetheless it can still prove useful as a nonlinear dimensionality reduction technique.

A control on the sparseness of factor matrices can be introduced, *e.g.* thanks to a $\ell_1$ norm penalty, as for nonnegative matrices [63]. The advantage is that it empirically leads to a unique solution, namely the sparsest [64, 65].

If an alternating algorithm is used, a proximal term can be inserted to guarantee local convergence [66], see Section 6.5.

9) the cost grows to infinity in all directions of the parameter space.

### 6.3.2 Block Decompositions

In some applications, like fluorescence spectroscopy where components have the same concentration over all experiments, or like multipath propagation in antenna array processing, it may happen that one factor matrix in the CPD has collinear columns, which prevents CP uniqueness since one of the Kruskal's rank is then equal to 1 in (6.11). More precisely, let $\boldsymbol{\mathcal{D}}$ a tensor written as:

$$\boldsymbol{\mathcal{D}} = \boldsymbol{a} \otimes \boldsymbol{b}_1 \otimes \boldsymbol{c}_1 + \boldsymbol{a} \otimes \boldsymbol{b}_2 \otimes \boldsymbol{c}_2, \tag{6.44}$$

then $\boldsymbol{\mathcal{D}}$ can be equivalently written as a so-called Block Term Decomposition (or BTD in short) [67, 68, 69] by factorizing component $\boldsymbol{a}$:

$$\boldsymbol{\mathcal{D}} = \boldsymbol{a} \otimes (\boldsymbol{b}_1 \otimes \boldsymbol{c}_1 + \boldsymbol{b}_2 \otimes \boldsymbol{c}_2) = \boldsymbol{a} \otimes \boldsymbol{B}\boldsymbol{C}^T \tag{6.45}$$

A sum of terms similar to equation (6.45) leads to a decompostion of the form:

$$\boldsymbol{\mathcal{T}} = \sum_{r=1}^{R} \boldsymbol{a}_r \otimes \boldsymbol{B}_r \boldsymbol{C}_r^T, \tag{6.46}$$

where matrices $\boldsymbol{B}_r$ and $\boldsymbol{C}_r$ are of respective sizes $n_2 \times L_r$ and $n_3 \times L_r$ for a $n_1 \times n_2 \times n_3$ tensor $\boldsymbol{\mathcal{T}}$. Consequently, the number $R$ of terms in such a BTD can be much smaller than tensor rank. Simultaneous to the discovery of block-term decomposition, model (6.46) was investigated under the name PARALIND [67], and the two names still coexist today.

Because of the relationship between equations (6.44) and (6.45), block-term decomposition is in fact a constrained CPD with colinear columns in one factor. It is to be noted that we refer here to a specific kind of block-term decomposition [69, 9, 70], but other more involved models were introduced in [68], which do not relate directly to CPD with collinear columns in factors.

An intrinsic property of block-term decompositions is that a rotation ambiguity is introduced in each block, since for any invertible matrix $\boldsymbol{P}$, $\boldsymbol{B}\boldsymbol{C}^T = \boldsymbol{B}\boldsymbol{P}\boldsymbol{P}^{-1}\boldsymbol{C}^T$. Uniqueness of the products $\boldsymbol{B}_r\boldsymbol{C}_r^T$ as well as uniqueness of factor $\boldsymbol{A}$ have been studied in the literature, [71, 17, 72].

In spirit, constraining the block-term decomposition model could be a solution to remove the rotational ambiguity of the blocks. To that end, sparsity-constrained block-term decomposition and coupled block-term decomposition have been studied [73, 70]. The coupling is however not as flexible as in [74]. Some authors have also developed a PARAFAC2 block-term decomposition [75].

Application-wise, the block-term decomposition has been used mainly for biomedical image processing to detect epileptic seizures [76, 77].

### 6.3.3 Structured Factors

#### 6.3.3.1 Re-parameterization

As discussed above, a versatile way to impose constraints in a tensor decomposition model is to constrain the factors directly. For instance for nonnegativity constraints, factors are required to have only nonnegative entries. This can be efficiently imposed by merely parameterizing entries as squares [78]. It is also possible to impose more complicated constraints via parameterization.

First, a parametric model may be used to describe one or several factors. For instance, factor $\boldsymbol{A}$ in a CPD may be a sinusoidal function so that $A_{ir} = \sin(2i\pi\rho + r\phi)$ and the new parameter set becomes $\rho, \phi$ [10]. Such parameterizations have been studied in the literature in the context of array processing, where factors are well represented by exponential maps [9, 11]. Damped exponentials also have been used to model factor matrices [12]. Exponential decay also appears in early literature of Chemometrics [79].

Second, a basis of representation may be provided in a matrix format, and the constrained factors are then represented by coefficients in a new feature space. For instance, both nonnegativity and smoothness may be imposed on factor matrix $\boldsymbol{A}$ by fixing a family of B-splines $\boldsymbol{D}$ of size $n_1 \times R_1$ for some small integer $R_1$ as described in [38], and impose that $\boldsymbol{A} = \boldsymbol{D}\boldsymbol{A}_c$. B-splines are piece-wise polynomial functions that are zero valued outside a given interval. Therefore, imposing $\boldsymbol{A}$ to live in the (nonnegative) span of $\boldsymbol{D}$ necessarily implies that it is nonnegative, and smooth as a sum of polynomials. Such splines bases are widely used in psychometrics, where factors in the CPD are heavily constrained by user prior knowledge [80].

Interestingly, if such a matrix $\boldsymbol{D}$ is provided as *a priori* information, and that $R_1$ is smaller than $R$, then provided the columns of $\boldsymbol{D}$ are free, a compression similar to the Tucker compression discussed in Section 6.5.3.1 can be done using the QR decomposition of matrix $\boldsymbol{D}$ [38].

#### 6.3.3.2 Dictionary constraints

An important issue in source separation is the identification, or labeling, of the outputs. This can be done for instance by comparing the output factors with a library, also called a dictionary, of reference factors, like reference emission and excitation spectra of some well-known chemical compounds. However, if all chemical compounds are known in advance, it is also possible to exploit this dictionary inside the source separation algorithm to produce labeled outputs and improve identification accuracy. This dictionary may be very large and very redundant, since the source may be characterized by a family of correlated reference spectra.

Formally, given a dictionary $\boldsymbol{D}$, one wants to select columns of, say, factor $\boldsymbol{A}$ in the columns of $\boldsymbol{D}$, so that $\boldsymbol{A} = \boldsymbol{D}(:, \mathcal{K})$ for an index set $\mathcal{K}$ of size $R$. Such a combina-

torial formulation was introduced in [46], but a former formulation using row-sparsity constraints can be found in [45]. Along with improving identification performances, dictionary constraints also make the low-rank approximation well-posed, similarly to non-negativity constraints.

Two related problems remain open. First, when no library is provided a priori, how can a dictionary be learned from a set of tensors? Second, if a provided dictionary is not exactly adapted to the data at hand, what distance would best describe the discrepancy between the constrained factor matrix and the provided dictionary?

Note that important work has also been done about *tensor dictionary learning*, mostly focussing on learning a dictionary from a matrix data set that has a tensor structure. The problem of dictionary learning is, as of now, unrelated to what has been presented above and has yet found no application in chemometrics that we know of, but an interested reader can refer to [43, 44, 81].

## 6.4

## Coupled decompositions

One of the reasons why tensors have gained importance in signal processing is that the complexity and variety of sensors has skyrocketed in the recent years. On the other hand, most well-studied data analysis tools, for instance Principal Component Analysis, Factor Analysis, linear regression and sparse regression, are designed for two-way arrays. From a practical point of view, this means that only a relationship between two experimental parameters (e.g. time, wavelength) can be inferred. Nowadays multiple such parameters are involved in the measurement process, and one possible way to deal with this fact is to build matrices of data by stacking such parameters, thus overlooking the real intricate relationship between all sets of experimental parameters. PARAFAC, and other previously described models, does mine relationships between all experimental parameters through a collection of separable patterns.

An interesting way to understand the CP decomposition is to cast it as a simultaneous low-rank matrix decomposition of a collection of matrices with equality constraint on the mixing matrices as detailed in Section 6.4.1 below. This is also in line with how Richard Harshman developed the PARAFAC model based on the principle of parallel proportional profiles [82]. Thus the CPD is a reasonable tool for data fusion, i.e. the joint analysis of multiple data sets. However, it is clear that previously presented models may not be used to tackle any data fusion problem, since the separability assumption may be too strong to describe stacked heterogeneous data sets. Even if that was not the case, data sizes may be completely different due to varying sampling rates among sensors and experiments, and stacking would then not be feasible naively.

This section introduces a general framework for designing tensor decomposition models in a broad context of data fusion. Regression models between multiple blocks of variables

will not be addressed here. Our main goal here is not to collect all existing tensor models that account for some specific types of fusion, but rather to give a taste of how to implement peculiar knowledge on the relationship between sources into a decomposition model.

Note that because of the wide range of problems that can be coined as data fusion, there exist numerous *a priori* unrelated models in chemometrics alone designed to address either multimodality or subject variability that we will not address in this chapter.

### 6.4.1

### Exact coupled decomposition, a first approach

Let us first illustrate these concepts on a simple data fusion model, namely the exact coupled decomposition. Suppose data from $N$ sensors are collected in the form of $N$ tensors $\boldsymbol{\mathcal{T}}_n$ of order 3, with one shared experimental parameter. The exact coupled decomposition model supposes that each tensor shares at least one factor with all the others. If exactly one is shared, then for all $n$ in $[1, N]$,

$$\boldsymbol{\mathcal{T}}_n = [\![\boldsymbol{A}, \boldsymbol{B}_n, \boldsymbol{C}_n]\!] + \boldsymbol{\mathcal{E}}_n, \tag{6.47}$$

where $\boldsymbol{\mathcal{E}}_n$ is a noise tensor, and $\boldsymbol{A}$ is the shared factor. Note that we have assumed, to simplify, that the shared experimental parameter's sampling rate is the same for each data set, which is not necessarily true in practice [74].

Although coupled tensor factorization stems from well established concepts such as canonical-correlation analysis [83] and although data fusion with tensors was previously studied in chemometrics [84, 85], exact coupled decomposition was formally introduced much later [6, 74]. Exact coupled decomposition has been used in many application domains such as metabolomics [86] or recommender systems [87].

Exact coupled decomposition is not only a useful data mining tool, but is also the link between matrix factorization models and tensor decomposition. The PARAFAC model may indeed be cast as an exactly coupled matrix factorization model: let $\boldsymbol{M}_k$ be a collection of $n_3$ matrices of size $n_1 \times n_2$ such that $\boldsymbol{M}_k = \boldsymbol{A}\boldsymbol{D}_k\boldsymbol{B}^\mathsf{T}$ where $\boldsymbol{A}$, $\boldsymbol{B}$ are respectively $n_1 \times R$ and $n_2 \times R$ matrices, and $\boldsymbol{D}_k$ is a $R \times R$ diagonal matrix of weights. Then the tensor $\boldsymbol{\mathcal{T}}$ obtained by stacking matrices $\boldsymbol{M}_k$ along a third mode, *i.e.*

$$\boldsymbol{\mathcal{T}}(:,:,k) = \boldsymbol{A}\boldsymbol{D}_k\boldsymbol{B}^T \tag{6.48}$$

follows a PARAFAC model

$$\boldsymbol{\mathcal{T}} = [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!] \tag{6.49}$$

where $\boldsymbol{C} = [\mathrm{diag}(\boldsymbol{D}_1), \ldots, \mathrm{diag}(\boldsymbol{D}_K))]$. This can be easily checked by looking at the expanded formula

$$\mathcal{T}_{ijk} = \sum_{r=1}^{R} A_{ir} D_{krr} B_{jr} = \sum_{r=1}^{R} A_{ir} B_{jr} C_{kr} \tag{6.50}$$

In other words, computing the CPD of a three-way array is equivalent to finding the common factorization of a collection of matrices with individual component weights, or again to compute the exact coupled decomposition of matrices where all factors are coupled.

Writing an approximation problem for exact coupled decomposition is straightforward (but its solution may raise difficulties, as pointed out in Section 6.2.8). In the presence of i.i.d. Gaussian noise of variance $\sigma_n$ on each entry of the tensor $\boldsymbol{\mathcal{T}}_n$, the data distribution $p(\boldsymbol{\mathcal{T}}_n|\boldsymbol{A}, \boldsymbol{B}_n, \boldsymbol{C}_n)$ is Gaussian and therefore the maximum likelihood estimator of all factors is given by the following optimization problem:

$$\underset{\boldsymbol{A},\boldsymbol{B}_n,\boldsymbol{C}_n}{\operatorname{argmax}} \log p(\boldsymbol{\mathcal{T}}_n|\boldsymbol{A}, \boldsymbol{B}_n, \boldsymbol{C}_n) = \underset{\boldsymbol{A},\boldsymbol{B}_n,\boldsymbol{C}_n}{\operatorname{argmin}} \sum_{n=1}^{N} \frac{1}{\sigma_n^2} \|\boldsymbol{\mathcal{T}}_n - [\![\boldsymbol{A}, \boldsymbol{B}_n, \boldsymbol{C}_n]\!]\|_F^2. \tag{6.51}$$

Many different optimization algorithms may be used to compute the exact coupled model, which are derived from either alternating least squares or all-at-once descent algorithms introduced in Section 6.5.1.1. Important contributions to the identifiability of the parameters of the exact coupled tensor decomposition model have been made by Sørensen *et. al.* [71].

This exact coupling model is however nowhere near satisfying in most practical scenarios. In fact, realistic problems are more complicated and may feature:

- different tensor sizes on the shared mode,
- more complex variation slice-wise of the coupled factors,
- a stochastic coupling relationship.

To extend the ideas presented above and allow for customization of coupled multiway decomposition models, a more flexible framework than exact coupled decomposition is therefore needed.

### 6.4.2

### A general framework for data fusion in tensor decompositions

As illustrated above, most data fusion methods for source separation make the assumption that a subset of parameters are linked. Describing how these parameters are linked is therefore the cornerstone of designing data fusion models. Understanding how to design tensor data fusion models is essential for linking various existing tensor decomposition models together. As an example, Box 6.4.2 shows how the PARAFAC2 model described in Section 6.2.7 can be written as a coupled matrix factorization model, shedding light on the link between PARAFAC and PARAFAC2.

**Block 1: PARAFAC2 as a flexible tensor coupled decomposition**

Suppose a collection of $n_3$ matrices $\boldsymbol{M}_k$ of sizes $n_1 \times n_2$ is to be jointly factorized using a coupled model, using a single coupled mode. Using above notations, this can be formalized as such:

$$\boldsymbol{M}_k = \boldsymbol{A}\boldsymbol{D}_k\boldsymbol{B}_k^{\mathsf{T}} \tag{6.52}$$

which can be rearranged into

$$\boldsymbol{M} = \left[\begin{array}{c|c|c|c} \boldsymbol{M}_1 & \ldots & \boldsymbol{M}_{n_3} \end{array}\right] = \boldsymbol{A}\left[\begin{array}{c|c|c|c} \boldsymbol{D}_1\boldsymbol{B}_1 & \ldots & \boldsymbol{D}_{n_3}\boldsymbol{B}_{n_3} \end{array}\right] = \boldsymbol{A}\bar{\boldsymbol{B}}^{\mathsf{T}} \tag{6.53}$$

by stacking "horizontally" matrices $\boldsymbol{M}_k$ (resp. matrices $\boldsymbol{D}_k\boldsymbol{B}_k$) into a large $n_1 \times n_2 n_3$ matrix $\boldsymbol{M}$ (resp. a large $R \times JK$ matrix $\bar{\boldsymbol{B}}^{\mathsf{T}}$). Therefore, if matrices $\boldsymbol{B}_k$ share no relationship, the coupled model is simply equivalent to a large low-rank matrix factorization. On the other hand, equality between matrices $\boldsymbol{B}_k$ would yield the PARAFAC model as shown in equation (6.48). An intermediate constraint may therefore be sought, so that factors $\boldsymbol{B}_k$ are all related but not equal. The PARAFAC2 model suggests to fix the inner-products $\boldsymbol{B}_k^T\boldsymbol{B}_k$ across $k$. This yields the following flexible matrix coupling model:

$$\boldsymbol{M}_k = \boldsymbol{A}\boldsymbol{D}_k\boldsymbol{B}_k^{\mathsf{T}} \quad \text{and} \quad \boldsymbol{B}_k = \boldsymbol{P}_k\boldsymbol{E} \tag{6.54}$$

where matrices $\boldsymbol{P}_k$ are $J \times R$ left-orthogonal matrices, i.e. $\boldsymbol{P}_k^{\mathsf{T}}\boldsymbol{P}_k = \boldsymbol{I}_R$, and $\boldsymbol{E}$ is a common $R \times R$ Grammian matrix (A Grammian matrix is a symmetric matrix containing all pairwise scalar products between several vectors.). In other words, $\boldsymbol{M}_k\boldsymbol{P}_k = \boldsymbol{A}\boldsymbol{D}_k\boldsymbol{E}^{\mathsf{T}}$, which is nothing more than a CPD.

A convenient way to formalize more general relationships between parameter sets in multiway array decompositions is to resort to a Bayesian probabilistic formulation. In a Bayesian framework, decomposing multiple tensors $\{\boldsymbol{\mathcal{T}}_n\}_{n\leq N}$ means finding the parameters $\{\boldsymbol{\theta}_n\}_{n\leq N}$ so that the probability $p(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N, \boldsymbol{\mathcal{T}}_1, \ldots, \boldsymbol{\mathcal{T}}_N)$ is maximized over $\{\boldsymbol{\theta}_n\}_{n\leq N}$. As shown in [74], to rewrite this criterion in a useful form, the following hypothesis is required:

**H1: Conditional independence of the data.** The data arrays $\boldsymbol{\mathcal{T}}_n$ are statistically independent when conditioned by their decomposition parameters $\boldsymbol{\theta}_n$. This means that knowing the factors of a decomposition for $\boldsymbol{\mathcal{T}}_n$, this tensor can be fully reconstructed, without using the other data sets.

Hypothesis **H1** is a technical assumption, and can be assumed to be true in most practical data fusion problems. Moreover, to provide a good Bayesian model for the coupled data sets, it is also necessary to know the joint densities of the coupled parameters $p(\theta_1, \ldots, \theta_N)$, as well as the likelihood functions $p(\boldsymbol{\mathcal{T}}_n|\theta_n)$, which contain the decomposition model for each data set. Under **H1** and given the likelihoods and the joint probability of the coupled parameters, the logarithm of the Maximum A Posteriori estimator is given

by

$$\underset{\theta_n}{\operatorname{argmax}} \sum_{n=1}^{N} \log\left(p(\boldsymbol{\mathcal{T}}_n|\theta_n)\right) + \log\left(p(\theta_1,\ldots,\theta_N)\right) \tag{6.55}$$

which can be used as the cost function in an optimization problem, see Section 6.5. In the vast majority of coupled decomposition problems, some blocks of parameters should be coupled, and some should not. For the latter, it is possible to marginalize their contribution to the joint distribution, so that only the joint distribution of the coupled parameters is used in (6.55).

Let us instantiate (6.55) with a simple demonstrative example. If each $\boldsymbol{\mathcal{T}}_n$ follows a CPD model with Gaussian Noise of i.i.d. noise of variance $\sigma_n^2$, and if column-wise normalized factors $\boldsymbol{C}_n$ are coupled through the following model:

$$\boldsymbol{C}_n = \boldsymbol{C}^* + \boldsymbol{\Gamma}_n \tag{6.56}$$

for a latent variable matrix $\boldsymbol{C}^*$, and a zero-mean Gaussian noise $\boldsymbol{\Gamma}_n$ whose entries have a variance ${\sigma_{C_n}}^2$, then equation (6.55) becomes

$$\underset{\boldsymbol{A}_n,\boldsymbol{B}_n,\boldsymbol{C}_n,\boldsymbol{C}^*}{\operatorname{argmin}} \sum_{n=1}^{N} \frac{1}{\sigma_n^2} \left\| \boldsymbol{\mathcal{T}}_n - [\![\boldsymbol{A}_n,\boldsymbol{B}_n,\boldsymbol{C}_n]\!] \right\|_F^2 + \frac{1}{{\sigma_{C_n}}^2} \|\boldsymbol{C}_n - \boldsymbol{C}^*\|_F^2 \tag{6.57}$$

where $\boldsymbol{C}_n$ are normalized column-wise.

Note that a latent shared factor matrix $\boldsymbol{C}^*$ is added to the set of parameters. This was done by supposing the joint probability of the coupled parameters is known conditionally to a latent variable $\theta^*$, for which a non-informative prior is used. Then using the Bayes law, equation (6.57) is obtained from (6.55). Also note that the normalization is important, since all factors $\boldsymbol{C}_n$ should relate to one matrix $\boldsymbol{C}^*$ with a given error measured by $\sigma_{C_n}$, and this coupling relationship is not invariant by scaling.

This flexible exact coupling example is simply meant for illustrating how a coupled decomposition model can be designed. In some practical cases, the probabilistic framework may be dropped and instead, and a deterministic coupling model can be used. All the examples of data fusion models described in the next subsection are indeed based on a deterministic description of the relationship between a subset of the decomposition variables. However, it should be borne in mind that, using a probabilistic framework, the modeling possibilities are much wider.

### 6.4.3

### Examples of coupled decomposition models

In what follows, we introduce coupled decomposition models that can be encountered in chemometrics. For most of them, solving the underlying optimization problem is non

trivial, but an interested reader can refer to the original publications for more details on this subject. To this list, one should add the PARAFAC2 model described in Section 6.2.7 and in Box 6.4.2.

#### 6.4.3.1 Advanced Coupled Matrix Tensor Factorization

Advanced coupled matrix tensor factorization [88] was designed to adapt exactly coupled decomposition (6.47) for situations when only a portion of the components of the coupled factor $\boldsymbol{A}$ are shared across the data sets (thus the "advanced" adjective).

A naive way to design a so-called partially coupled decomposition model is to fix manually the subset of coupled columns of factors $\boldsymbol{A}_n$, such that

$$\boldsymbol{A}_n = [\boldsymbol{A} \mid \boldsymbol{A}_n^{nc}] \tag{6.58}$$

where matrix $\boldsymbol{A}$ contains the shared components, matrices $\boldsymbol{A}_n^{nc}$ the uncoupled ones, and the concatenation is horizontal. However, such a formulation suggests that the number of coupled components is known in advance, which may not be the case. Moreover, a specific ordering of the components is imposed which can bring some permutation problems in the decomposition algorithm.

Advanced coupled matrix tensor factorization proceeds differently. It makes use of a sparsity constraint on the norms of the components (recall formulation (6.25)), to impose this partial coupling, solving the following optimization problem[10]:

$$\begin{aligned}\underset{\boldsymbol{A},\boldsymbol{B}_n,\boldsymbol{C}_n,\boldsymbol{\mathcal{S}}_n}{\operatorname{argmin}} \; & \|\boldsymbol{\mathcal{T}}_n - [\![\boldsymbol{\mathcal{S}}_n; \boldsymbol{A}, \boldsymbol{B}_n, \boldsymbol{C}_n]\!]\|_F^2 + \lambda \sum_{n=1}^{n_3} \|\text{Diag}(\boldsymbol{\mathcal{S}}_n)\|_1 \\ & + \alpha \sum_{r=1}^{R} \left[ (\|\boldsymbol{a}_r\|_2^2 - 1)^2 + \sum_{n=1}^{n_3} (\|\boldsymbol{b}_{nr}\|_2^2 - 1)^2 + (\|\boldsymbol{c}_{nr}\|_2^2 - 1)^2 \right]\end{aligned} \tag{6.59}$$

where $\boldsymbol{\mathcal{S}}_n$ are diagonal tensors containing the values $\sigma_{nr}$ of the components intensities for each tensor $\boldsymbol{\mathcal{T}}_n$ with $n \leq n_3$, while $\lambda$ and $\alpha$ are hyperparameters tuned by the user. By forcing normalization of the factor matrices, the components amplitude stored in $\boldsymbol{\mathcal{S}}_n$ truly reflect the importance of component $r$ in data block $n$. If a component $\boldsymbol{a}_r$ in the shared matrix $\boldsymbol{A}$ should not be used in data block $\boldsymbol{\mathcal{T}}_n$, then the score of that component in the decomposition of this data block $\sigma_{nr}$ may be set to zero. This motivates the use of a sparsity inducing metric such as the $\ell_1$ norm on the entries of the diagonal tensors $\boldsymbol{\mathcal{S}}_n$.

Advanced coupled matrix tensor factorization has been used successfully in metabolomics and brain imaging [89, 90].

On the theoretical side, the uniqueness of partially exactly coupled decompositions has been studied in the case of matrix-tensor coupled decompositions [91]. Partial coupling is shown to reduce rotational ambiguities in the matrix decomposition without completely

10) the original publication gave a slightly different optimization problem featuring a relaxation of the $\ell_1$ norm.

negating it. Furthermore, the presence of constraints, in particular nonnegativity constraints, may further improve the identification properties of the partially coupled matrix-tensor models.

#### 6.4.3.2 Shift PARAFAC and others

When dealing with several data sets acquired in similar experimental conditions, it is natural to assume that in the presence of time measurements, some delay is to be accounted for across the various data sets. But such delays may actually depend on the source index, if some variability occurs in the behavior of each source along the various measurements. Modeling and estimating this delay is partially what the PARAFAC2 model described above does, but in a fairly general manner.

In 2003, Harshman [92] introduced a more specific modeling of component shifts in a collection of coupled low rank data matrices. This model, coined as Shift-PARAFAC, may be cast in the coupled decomposition framework as follows:

$$[\boldsymbol{C}_n]_{k,r} = [\boldsymbol{C}^*]_{k+\tau_n,r} \tag{6.60}$$

supposing the coupled factor is $\boldsymbol{C}$ and the amount of shift is proportional to the sampling rate. It is also possible to design an arbitrary shift amount, *i.e.* not an increment of the indices, by resorting to interpolation between the latent factor and its shifted instances. The computation of the Shift PARAFAC model can be done rather efficiently by resorting to the Fourier transformation of the data [93], since a shift in time domain becomes a product in Fourier domain. The Shift-PARAFAC model has been used mainly to decompose FMRI data [93], so as to account for variations in the activation profiles of brain sources.

It is worth noting that a few other similar relationships have been explored in the literature. In particular, distortions due to time contraction or dilatation are discussed in the Warped Factor Analysis model [94, 95]. Also, both Shift-PARAFAC and Warped Factor Analysis differ from data alignment approaches like Ico-shift [96, 97] which preprocess the data to remove any delay among the related data slices. Indeed, if only the data slices are shifted, then implicitly all the components are supposed to have the same delays.

#### 6.4.3.3 GSVD

The Generalized Singular Value Decomposition model has been proposed by Van Loan [98] as a generalization of the SVD to more than a single matrix. It was one of the first attempts at defining a joint diagonalization technique. Its main usage in source separation has been for genomics, where GSVD has been applied notably by Alter *et. al.* to discriminate cancerous DNA from sane DNA [99].

GSVD resembles an exactly (i.e. without noise) coupled decomposition of two matrices:

$$\begin{aligned} \boldsymbol{M}_1 &= \boldsymbol{U}_1 \boldsymbol{\Sigma}_1 \boldsymbol{V}^T \\ \boldsymbol{M}_2 &= \boldsymbol{U}_2 \boldsymbol{\Sigma}_2 \boldsymbol{V}^T \end{aligned} \tag{6.61}$$

but with orthogonality constraints imposed on the non-coupled matrices $\boldsymbol{U}_n$. Note that without such constraints, computing the exact coupled decomposition of a collection of matrices amounts to a single matrix factorization of the stacked matrices $\boldsymbol{M}_n$, which does not admit an essentially unique solution.

On the other hand, the parameters of the GSVD are identifiable, and a closed-form algorithm is available to compute it when the data are not corrupted by noise. For these reasons, GSVD can be used as an exploratory model, in a similar spirit as Principal Component Analysis, rather than being cast as a physical modeling of the two data sets.

Notably, GSVD was also extended to deal with multiple data matrices [100], and with two coupled third-order tensors [101]. In both cases, the relationship with coupled models is not as straightforward.

## 6.5 Algorithms

This section aims at giving a short overview of simple and well-understood optimization algorithms that are known to work for computing tensor decomposition models. As a warning to already informed readers however, research on optimization techniques for tensor decompositions is extremely prolific, and surveying all the available methods while discussing their pros and cons would require another book in itself. Therefore only our understanding of mainstream approaches are described below. On the other hand, to readers who simply want to make use of well-designed toolboxes for source separation problems can turn to the following programs:

- Tensor toolbox[11]: this open-source toolbox features a particular care to processing and storaging large sparse tensors, and implements several basic tensor routines. It can therefore also be used as a backend onto which building one's own code.
- N-way toolbox[12]: a simple yet comprehensive open-source toolbox implementing the Alternating Least Squares to compute both nonnegative and unconstrained CP, as well other related regression models and the PARAFAC2 decomposition.
- Tensorly[13]: a tensor decomposition collaborative open-source toolbox in Python that mimics the scikit-learn syntax. It supports the use of several backends such as numpy or pytorch, and is geared towards machine learning applications.
- Tensorlab[14]: a well maintained open-source toolbox, which can identify many of the well-known tensor models, including CPD, block-term decomposition, exact and flex-

11) https://www.tensortoolbox.org/
12) http://www.models.life.ku.dk/nwaytoolbox
13) http://tensorly.org/
14) https://www.tensorlab.net/

ible joint decompositions as well as imposing various constraints on the factors. It is based on a nonlinear least squares solver and develops its own syntax.

- PLS-Toolbox[15]: A very comprehensive, open-source (and commercial) toolbox for general chemometric modelling. It includes tools for CPD, Tucker and PARAFAC2 including various constrained versions. It also includes older direct methods for CPD modelling based on generalized eigenvalue decomposition such as the generalized rank annihilation method.

Again, many other toolboxes exist. Several lists updated regularly are available online[16]. Some original codes can also be found on authors' home pages[17], but are generally not part of a toolbox.

### 6.5.1

### Unconstrained tensor decomposition

Among all tensor decomposition models, the most studied by far in terms of optimization strategies is the CPD. In the unconstrained case, both iterative and direct algorithms have been designed in the literature, leading to a wide variety of possible algorithms to choose from. Iterative methods are however the most common choice for approximate decompositions, and are therefore the main focus of this section.

#### 6.5.1.1 Iterative algorithms for approximate CPD

Let $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, and consider the following cost function[18]:

$$\Upsilon(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}) = \|\boldsymbol{\mathcal{T}} - [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]\|_F^2. \tag{6.62}$$

The minimization of objective (6.62) is the problem we end up with, if we want to find the maximum likelihood estimates of factor matrices $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$, when the data follow a rank-$R$ PARAFAC model corrupted by an additive isotropic Gaussian noise. Equivalently, the solution to (6.62) is the best rank-$R$ approximation of $\boldsymbol{\mathcal{T}}$ when it exists.

Since no closed form solution for the minimum of (6.62) is known in the general case, iterative methods rely on the following strategy:

- Provide an initial guess for factors $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$.
- Fix a subset (possibly empty) of the parameters and update the others.
- Stop when convergence is reached.

15) `www.eigenvector.com`
16) `www.tensorworld.org/toolboxes/`, `https://tensornetwork.org/software/`
17) See for instance the tensor package at `www.gipsa-lab.grenoble-inp.fr/ pierre.comon/TensorPackage/tensorPackage.html`.
18) This section is written for third-order tensors without loss of generality

The main difference between various iterative methods is therefore the choice of fixed parameters and the update strategy.

**Gradient computation** Most iterative strategies make use of the gradient of (6.62) for the update rule. The gradient of (6.62) is as follows:

$$\begin{array}{l} \frac{\partial \Upsilon}{2\partial \boldsymbol{A}} = -\boldsymbol{T}_{(1)}\left(\boldsymbol{B}\odot\boldsymbol{C}\right) + \boldsymbol{A}\left(\boldsymbol{B}^T\boldsymbol{B}\boxdot\boldsymbol{C}^T\boldsymbol{C}\right) \\ \frac{\partial \Upsilon}{2\partial \boldsymbol{B}} = -\boldsymbol{T}_{(2)}\left(\boldsymbol{A}\odot\boldsymbol{C}\right) + \boldsymbol{B}\left(\boldsymbol{A}^T\boldsymbol{A}\boxdot\boldsymbol{C}^T\boldsymbol{C}\right) \\ \frac{\partial \Upsilon}{2\partial \boldsymbol{C}} = -\boldsymbol{T}_{(3)}\left(\boldsymbol{A}\odot\boldsymbol{B}\right) + \boldsymbol{C}\left(\boldsymbol{A}^T\boldsymbol{A}\boxdot\boldsymbol{B}^T\boldsymbol{B}\right) \end{array} \tag{6.63}$$

To derive these gradients easily, one may resort to the matricized versions of the PARAFAC model. Indeed, since the Frobenius norm acts entry-wise, the cost (6.62) is equivalent rewritten as

$$\Upsilon(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}) = \|\boldsymbol{T}_{(1)} - \boldsymbol{A}(\boldsymbol{B}\odot\boldsymbol{C})^T\|_F^2. \tag{6.64}$$

The gradient of this cost function with respect to $\boldsymbol{A}$ is given by

$$\left(-\boldsymbol{T}_{(1)} + \boldsymbol{A}\left(\boldsymbol{B}\odot\boldsymbol{C}\right)^T\right)\left(\boldsymbol{B}\odot\boldsymbol{C}\right). \tag{6.65}$$

At this stage, with some formula manipulation, one may not that for any indexes $r$ and $q$ in $[1, R]$,

$$\Big[(\boldsymbol{B}\odot\boldsymbol{C})^T(\boldsymbol{B}\odot\boldsymbol{C})\Big]_{rq} = (\boldsymbol{b}_r\boxtimes\boldsymbol{c}_r)^T(\boldsymbol{b}_q\boxtimes\boldsymbol{c}_q) = (\boldsymbol{b}_r^T\boldsymbol{b}_q)(\boldsymbol{c}_r^T\boldsymbol{c}_q). \tag{6.66}$$

Therefore the second term in the gradient simplifies into $\boldsymbol{B}^T\boldsymbol{B}\boxdot\boldsymbol{C}^T\boldsymbol{C}$, which has a much lower computational complexity for small $R$ since it is computed by multiplying an $n_i \times R$ matrix with its transpose, plus a few $R \times R$ element-wise products.

Then the bottleneck in the gradient computation is the matrix product $\boldsymbol{T}_{(1)}\left(\boldsymbol{B}\odot\boldsymbol{C}\right)$, sometimes called Matricized Tensor Times Khatri Rao Product (MTTKRP), which has a naive complexity of $\mathcal{O}(Rn_1n_2n_3)$ since it is computed as the matrix product of a $n_1 \times n_2n_3$ matrix with a $n_2n_3 \times R$ matrix. The fast implementation of this costly product, and in general of tensor contractions, is the topic of many recent researches in high performance computing [102, 103, 104, 105, 106, 107].

**Alternating least squares** The workhorse algorithm for identifying the PARAFAC model is the Alternating Least Squares algorithm. It is very easy to implement, and although several other algorithms are more reliable, ALS still performs reasonably well in some cases. In its most simple form, ALS also features no parameter tuning. However, it fails to deliver in difficult scenarios, such as in the presence of near-colinear dependency among the factors's columns or rank under- (or over-) estimation. Notably, ALS is a particular case of the nonlinear block Gauss-Seidel method for solving nonlinear systems. It can be adapted to tackle very large data sets [108] and coupled decompositions [74].

The core principle of ALS is to minimize (6.62) with respect to each factor while the others remain fixed. Since PARAFAC is a multilinear model, it is linear with respect to each such block of parameters, and therefore the optimal solution with respect to only one block is known in closed form. Skipping the technical conditions for the existence of such a closed form solution, one may simply set the gradients (6.63) to zero to obtain the sequential update rules for the ALS. For instance, the estimate $\widehat{\boldsymbol{A}}$ of matrix $\boldsymbol{A}$ is given by

$$\widehat{\boldsymbol{A}} = \left(\boldsymbol{T}_{(1)}\left(\boldsymbol{B} \odot \boldsymbol{C}\right)\right)\left(\boldsymbol{B}^T\boldsymbol{B} \boxdot \boldsymbol{C}^T\boldsymbol{C}\right)^{-1}. \tag{6.67}$$

A pseudo-code for ALS is given in Algorithm 1. The stopping conditions can be a fixed number of iterations, the relative decrease of $\Upsilon$ across successive iterations reaching a threshold, or any arbitrary condition that fits the needs of a particular application. Also, the inverse in the factor updates does not need to be explicitly computed. Rather, the least squares update can be computed by solving the linear system obtained by setting the gradients in (6.63) to zero.

**Algorithm 1** A squeleton of the Altenating Least Squares algorithm

**Input:** Data tensor $\boldsymbol{\mathcal{T}}$, Initial values $\boldsymbol{A}^{(0)}, \boldsymbol{B}^{(0)}, \boldsymbol{C}^{(0)}$
Set $k = 0$
**while** stopping condition is not met **do**

$$\boldsymbol{A}^{(k+1)} = \left(\boldsymbol{T}_{(1)}\left(\boldsymbol{B}^{(k)} \odot \boldsymbol{C}^{(k)}\right)\right)\left(\boldsymbol{B}^{(k)^T}\boldsymbol{B}^{(k)} \boxdot \boldsymbol{C}^{(k)^T}\boldsymbol{C}^{(k)}\right)^{-1}$$

$$\boldsymbol{B}^{(k+1)} = \left(\boldsymbol{T}_{(2)}\left(\boldsymbol{A}^{(k+1)} \odot \boldsymbol{C}^{(k)}\right)\right)\left(\boldsymbol{A}^{(k+1)^T}\boldsymbol{A}^{(k+1)} \boxdot \boldsymbol{C}^{(k)^T}\boldsymbol{C}^{(k)}\right)^{-1}$$

$$\boldsymbol{C}^{(k+1)} = \left(\boldsymbol{T}_{(3)}\left(\boldsymbol{A}^{(k+1)} \odot \boldsymbol{B}^{(k+1)}\right)\right)\left(\boldsymbol{A}^{(k+1)^T}\boldsymbol{A}^{(k+1)} \boxdot \boldsymbol{B}^{(k+1)^T}\boldsymbol{B}^{(k+1)}\right)^{-1}$$

Increment k = k+1
**end while**
**Output:** Final estimates $\boldsymbol{A}^{(k)}, \boldsymbol{B}^{(k)}, \boldsymbol{C}^{(k)}$

Importantly, ALS iterates of the objective function always converges since the cost function decreases at each iteration and is bounded by below. However, this does not garantee that factors $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$ converge; if they do, the obtained solution is not either guaranteed to be a local minimum. For this to be true in the framework of local convergence[19], some reasonable technical conditions on the Hessian matrix of $\Upsilon$ should be met [109], which sadly can hardly be checked in practice. Global convergence to a local minimum is also subject to theoretical technical conditions [110]. Further, in some pathological cases, the convergence speed of ALS can be sub-linear [110], a fact observed in difficult decomposition problems [111].

An important tweak on ALS, that often improves its convergence speed drastically, is to extrapolate factor estimates using current and previous estimates. This extrapolation

19) local convergence means convergence if the starting point of the algorithm is in the neighborhood some local minimum.

procedure is standard in optimization [112], and is often called 'Line Search' in the tensor community. The practical speed up makes it a nice feature of a good ALS implementation [19, 111, 113, 114]. Note that extrapolation may lead to increasing the cost function at some iterations, and the so-called restart strategy that discard steps increasing the cost is the key to obtain an efficient acceleration in some of the works mentioned above [113, 114].

**First and second order descent algorithms** All-at-once gradient-based methods are also a great choice for computing an unconstrained PARAFAC model. First, for first-order methods (based solely on the gradient for finding a descent direction), the complexity per iteration is the same as the ALS. Second, all the knowledge on descent methods applied to non-convex problems may be put to profit, in particular through convergence results or stochastic approaches for handling very large data sets [115]. Third, missing data can be dealt with using a mask of weights on the data, which is not feasible using the ALS algorithm. Since large data sets with a lot of missing data are common in machine learning, all-at-once descent algorithms have notably been preponderant in this context [116, 117].

There is no particular practical difficulty to computing a PARAFAC decomposition using well-known descent algorithms such as gradient descent, non-linear conjugate gradient descent [118] or Gauss-Newton once gradients (6.63) have been computed. However, some parameters such as the step size need to be tuned, which makes the ALS a simpler choice for novice users. On the other hand, for high precision works or difficult scenarios, resorting to second-order methods promoted to solve nonlinear least squares problems, like the Levenberg Marquardt algorithm [111] can prove rewarding. Indeed, the Jacobian matrix has a particular structure that can be used to speed-up the – otherwise time demanding – Hessian computation. Moreover, second-order methods have guaranteed local convergence at quadratic speed.

**Normalization** Normalizing the columns in the CPD is not always mandatory in practice, but it has two advantages: (i) it avoids scaling indeterminacies, and (ii) it helps avoiding very small/large values in the factors, thus improving numerical stability as well as providing interpretable results.

#### 6.5.1.2 Deflation and N-PLS

Up to now, we supposed that the number of terms $R$ in the decomposition is known. In practice this is rarely the case, but finding the optimal $R$ has proven to be a difficult problem, mainly because contrarily to matrices, best fitting models with adjacent rank values may have no relationship. The procedure consisting of computing $R$ successive rank-1 approximations with the goal of obtaining a rank-$R$ approximation is often referred to as *deflation*. This idea works for matrices, by subtracting the best rank-1 approximation at each iteration. But the reader should pay attention to the fact that this does not work for tensors, since subtracting the best rank-1 approximation generally does not reduce its rank [119, 120, 121]. Also note that it does not work either for matrices in $\mathbb{R}^+$ because

of lack of stability by subtraction. Therefore, a naive strategy is to test several ranks and pick the one that works the best, but other methods have been proposed for specific applications [122, 123, 124].

Another way to perform deflation to compute approximate CPD is by means of the N-PLS. Let us first describe what Partial Least Squares (PLS) is.

Given a $n \times p$ matrix $\boldsymbol{X}$ and a $n \times 1$ data vector $\boldsymbol{y}$, the goal of PLS is to find the part $\hat{\boldsymbol{y}}$ of $\boldsymbol{y}$ that is related to $\boldsymbol{X}$ in the form $\hat{\boldsymbol{y}} = \boldsymbol{X}\,\boldsymbol{b}$. The classical solution to finding this regression vector is well-known and given by $\boldsymbol{b} = (\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{y}$. However, it is not desired to compute this expression: first, it can be computationally prohibitive, and second, matrix $\boldsymbol{X}$ can be ill-conditioned (e.g. if columns are close to collinear).

Partial least Squares regression (PLS) aims at computing an approximation of $\boldsymbol{b}$ and $\hat{\boldsymbol{y}}$ by delivering at each iteration $k$ the loading that provides a score vector with the highest possible cross-covariance with $\boldsymbol{y}$ (or the residual part of $\boldsymbol{y}$). It is robust with respect to ill-conditioning.

It turns out that the PLS iterations perform nothing else but the minimization of the objective $\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{b}\|^2$ by the conjugate gradient algorithm[20] [125]. In fact, PLS yields at each iteration $k$ the projection of $\boldsymbol{b}$ onto the subspace spanned by the $k$ dominant eigenvectors of matrix $(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})$. Hence, it can be stopped before convergence, and can output at any time an approximation of the best regression.

The multi-linear partial least squares regression, also known as $N$-way PLS regression, is an extension of the two-way PLS regression [126]. It is based on sequentially extracting rank-one tensors from a given tensor $\boldsymbol{\mathcal{T}}$ and a given vector $\boldsymbol{y}$. The first rank one tensor has the property that the mode one component vector has maximal covariance with the vector similar sized, $\boldsymbol{y}$, to be predicted. Subsequently $\boldsymbol{\mathcal{T}}$ and $\boldsymbol{y}$ are orthogonalized with the mode one vector, and a new rank-one tensor is determined from the residual data. The process is repeated as long as new components improve the predictions, which is usually determined through cross-validation or similar tests. Note that in the algorithm described in [126], the rank-one approximation of a tensor is computed via two rank-one approximations of matrices, which is possible by breaking the role symmetry of the three modes.

#### 6.5.1.3 Exact decomposition methods

Another line of research on decomposition algorithms is to find exact decomposition algorithms, such that the remaining error in equation (6.62) is zero. Of course one can resort to the iterative methods described above. But in the noiseless case, there is no need to introduce a statistical framework, and therefore algebraic methods, that are not theoretically robust to noise, can provide fast and reliable solutions. Because in source separation, such exact decomposition problems have a relatively smaller importance, we shall here only provide a few references that an interested reader can refer to, *e.g.* [127, 128, 129, 130], or

20) This has been proved in exact arithmetic, i.e. if there are no rounding errors.

in the world of chemistry [131, 132, 133, 134]. In particular, in the absence of noise, only two matrix slices need to be used to compute a matrix factor of the CPD [135, 136].

### 6.5.2 Constrained tensor decomposition

Although unconstrained tensor decomposition algorithms are today quite well understood, constrained tensor decomposition algorithms on the other hand have been an important research theme over the last decades. It is practically impossible to summarize all the works on this topic in one section. For instance, the sole case of nonnegative matrix factorization, which is a second order constrained tensor factorization problem, has been discussed in the entire Chapter **??**. Therefore, in this section, we shall simply sketch the research directions that have been pursued.

#### 6.5.2.1 Constrained Least Squares

A first approach to constrained tensor decomposition is to modify the ALS algorithm presented in Section 6.5.1.1 to impose the constraints on the decomposition factors. Since ALS relies on solving least squares problems alternatively, this approach boils down to solving constrained least squares problems.

For nonnegativity constraints, various efficient algorithms are available. The first nonnegative least squares algorithm appeared in [137, 138], and made use of an active set strategy. It has been adapted to compute the nonnegative CPD in [139]. More recent techniques employ exact block coordinate descent, using the observation that nonnegative least squares can be solved exactly by clipping to zero for one-dimensional data [140]. These approaches have been respectively used by Bro *et.al.* and Phan *et.al.* for computing nonnegative CPD, with a computational load comparable to the unconstrained CPD [139, 41].

To tackle a wider set of constraints, such as sparsity, it was suggested in [117] to solve each constrained least squares problems successively for each factor using a splitting of variables and a primal dual algorithm, namely the Alternating Direction Method of Multipliers (ADMM). The advantage of ADMM in this context, on top of the variety of is problems it can tackle, is parallelism.

#### 6.5.2.2 Projected gradient and all-at-once proximal methods

Similarly to the unconstrained case, constrained tensor decompositions can be tackled using variants of gradient descent for constrained problems. The goal here is not to give a full description of constrained convex optimization, but in a nutshell, constrained first order methods are based on projection operators. Namely, after a gradient step has been performed, the parameters are updated by projecting them onto the set of constraints. Another solution is to add a penalization term in the objective function to promote the constraint. Projected gradient and penalized approaches can be studied and improved

under the hood of proximal operators, see Chapter 3 for more details on these optimization techniques and Chapter 2 for their application in the context of NMF.

Various first order methods have been used to compute the CPD, including penalized gradient [141] or proximal gradient [142]. Second order methods, relying on an estimation of the curvature of the cost function using second order derivatives, have been extensively used in conjunction with projection or penalization to be used in the Tensorlab toolbox [143].

#### 6.5.2.3 Parametric approaches

In the particular case of nonnegativity, instead of explicitly imposing the constraint on the factors, several authors have suggested to parameterize the variables, for instance as squares, so that the nonnegativity constraints are implicitly imposed; see *e.g.* [141, 78]. Actually, parametric approaches are a convenient way to handle structured factors in tensor decompositions, both with respect to formalism and optimization [144]. Notably, this kind of approach is used in several packages including the TensorPackage and Tensorlab. See page 30 for links. See also [57] for a unit-norm parameterization.

### 6.5.3 Handling large data sets

All the algorithms we presented above make the implicit assumption that the data set can fit into the computer memory, so that any data point can be accessed easily. However, when dealing with very large tensors, this may not be the case. Historically, a compression method coined as the Tucker compression, introduced below, served as both an acceleration method and a storage technique. It may however not be computable in reasonable time.

To cope with very large data sets, several strategies have been explored in the literature, such as sketching or randomized sampling [108]. However, this rapidly evolving topic is out of the scope of this chapter.

#### 6.5.3.1 Multilinear SVD Compression

Given a large tensor $\mathcal{T}$ of size $n_1 \times n_2 \times n_3$ following an unconstrained unknown PARAFAC model of small rank $R$, computing the CP decomposition may prove quite computationally time consumming. On the other hand, since the tensor is explained by a relatively small number of parameters, in fact by $R(n_1 + n_2 + n_3 - 2)$ parameters, it should be possible to reduce the data set to a more essential one, that can be stored and manipulated instead of the whole tensor.

Finding tensor representations for efficient storage or fast computation of decomposition models is actually a very active field of research, with representations such as the hierarchical decomposition or the tensor train format, see [145] and references therein. However

in the context of source separation, and in particular in chemometrics, the most widely used representation method for storage and fast decompositions is the so-called Tucker compression or multilinear SVD compression, which is described in Section 6.2.5. Thus in what follows, we only describe this usual compression method, keeping in mind that newer approaches vastly widen the following discussion.

The idea behind multilinear SVD compression is to use the information that the rank of the tensor approximating the data is small with respect to its dimensions. Then because multilinear ranks, *i.e.* the ranks of the unfoldings, are always smaller or equal to the tensor rank, the approximate tensor must have small multilinear ranks as well. Therefore, using a truncated singular value decomposition of each unfolding as a way to compute approximate multilinear SVD[21)], a basis for each mode is obtained which can be used to project the data tensor onto a feature space of lower dimensions.

Formally, if a CPD $\boldsymbol{\mathcal{T}} = [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]$ is sought, first a multilinear SVD $\boldsymbol{\mathcal{T}} = [\![\boldsymbol{\mathcal{G}}; \boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}]\!]$ is computed where $\boldsymbol{U}, \boldsymbol{V}$ and $\boldsymbol{W}$ are left orthogonal matrices of respective sizes $n_1 \times R_1$, $n_2 \times R_2$ and $n_3 \times R_3$, and the compressed dimensions $R_i$ are larger than or equal to $R$. Then, using Property 6.2,

$$\boldsymbol{\mathcal{G}} = [\![\boldsymbol{\mathcal{T}}; \boldsymbol{U}^T\boldsymbol{A}, \boldsymbol{V}^T\boldsymbol{B}, \boldsymbol{W}^T\boldsymbol{C}]\!] := [\![\boldsymbol{A}_c, \boldsymbol{B}_c, \boldsymbol{C}_c]\!], \tag{6.68}$$

which is nothing more than a CPD of the smaller $R_1 \times R_2 \times R_3$ tensor $\boldsymbol{\mathcal{G}}$. Once that CPD is computed, the original CPD of the larger tensor $\boldsymbol{\mathcal{T}}$ can be recovered by $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{A}_c$ and similarly on other modes.

In practice, given a large tensor $\boldsymbol{\mathcal{T}}$ for which multiple approximate PARAFAC models of various ranks are to be computed, it is sufficient to compute the multilinear SVD of $\boldsymbol{\mathcal{T}}$ with reasonably small multilinear ranks, which outputs matrices $\boldsymbol{U}, \boldsymbol{V}$ and $\boldsymbol{W}$. Then after computing the compressed core tensor $\widehat{\boldsymbol{\mathcal{G}}}$ once, $\widehat{\boldsymbol{\mathcal{G}}}$ becomes the new data set, to be decomposed using any PARAFAC model with compressed factors $\boldsymbol{A}_c$, $\boldsymbol{B}_c$ and $\boldsymbol{C}_c$ of small sizes $R_i \times R$. Tensor $\boldsymbol{\mathcal{T}}$ can also be stored with small loss using its multilinear SVD compression, while using a PARAFAC model often leads to a more lossy compression.

As a side note, very few works study efficient compression and acceleration techniques in the presence of constraints. In our opinion, this topic is a promising line of research. Early works have been proposed for nonnegative CPD [55].

**Structured decompositions** Another strategy to accelerate tensor decomposition algorithms is to simply write tensor $\boldsymbol{\mathcal{T}}$ as a structured tensor using any tensor decomposition model, for instance $\boldsymbol{\mathcal{T}} = [\![\boldsymbol{\mathcal{G}}; \boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}]\!]$. Using such structure leads to faster computations and lowers the memory requirements just like the Tucker compression. For instance, using again MLSVD, gradient (6.63) with respect to $\boldsymbol{A}$ is written as

$$\frac{\partial \Upsilon}{2\partial \boldsymbol{A}} = -\boldsymbol{U}\boldsymbol{G}^{(1)}\left(\boldsymbol{V}^T\boldsymbol{B} \odot \boldsymbol{W}^T\boldsymbol{C}\right) + \boldsymbol{A}\left(\boldsymbol{B}^T\boldsymbol{B} \boxdot \boldsymbol{C}^T\boldsymbol{C}\right) \tag{6.69}$$

21) As explained in [21], the solution obtained by SVDs would not be optimal in a noisy setting. Nevertheless, this truncation procedure is generally broadly sufficient as a preprocessing before computing the exact CPD.

and the data-factors product, which is the bottleneck, has now a reduced complexity if $\boldsymbol{U}$ has fewer columns than rows.

The computation speed-up is similar but smaller than using Tucker compression, however the structured approximation technique extends trivially to any constrained decomposition of $\boldsymbol{\mathcal{T}}$ which makes it attractive in practice [144].

**Other cost functions for fitting the CPD** As a last remark, it often occurs that the discrepancy between the data tensor $\boldsymbol{\mathcal{T}}$ and the CPD is not efficiently measured by the Frobenius norm. In fact, a wide variety of distances may be used to fit a CPD, which may be obtained by taking the log-likelihood of the data distribution.

Despite the large choice of distance, there is a trick to easily obtain the gradient of a cost function written as

$$f \circ g(\boldsymbol{\mathcal{T}}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}) = f(\boldsymbol{\mathcal{T}} - [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]) \tag{6.70}$$

Indeed, the following chain rule may be used [146]:

$$\begin{aligned}
\tfrac{\partial f\circ g}{\partial \boldsymbol{A}}(\boldsymbol{\mathcal{T}}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}) &= -\nabla_f(\boldsymbol{\mathcal{T}} - [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!])\,(\boldsymbol{B} \odot \boldsymbol{C}) \\
\tfrac{\partial f\circ g}{\partial \boldsymbol{B}}(\boldsymbol{\mathcal{T}}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}) &= -\nabla_f(\boldsymbol{\mathcal{T}} - [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!])\,(\boldsymbol{A} \odot \boldsymbol{C}) \\
\tfrac{\partial f\circ g}{\partial \boldsymbol{C}}(\boldsymbol{\mathcal{T}}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}) &= -\nabla_f(\boldsymbol{\mathcal{T}} - [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!])\,(\boldsymbol{A} \odot \boldsymbol{B})
\end{aligned} \tag{6.71}$$

which is nothing more than the usual composition chain rule $(f \circ g)'(x) = f'(g(x))g'(x)$ extended to vector valued functions. Note that for $f$ set to the Frobenius norm, equation (6.71) recovers the gradients shown in (6.63) since $\nabla_{\|\ \|_F^2}(x) = 2x$.

More techniques can be found in the literature [146] which tackle the gradient computation for more general cost functions to fit the CPD.

## 6.6

## Applications

### 6.6.1

### Preprocessing

Before analyzing data sets, it is often necessary or beneficial to preprocess data. This goes for multi-way data as well. And essentially, the preprocessing is not much different from normal matrix data. The reader is therefore referred to the literature for classical preprocessing such as scatter correction of infrared spectral data [147], baselining of Raman data [148], removal of Raman and Rayleigh scattering effects before analyzing fluorescence [56], normalization of e.g. omics data [149], etc. One aspect though merits some special attention.

Centering and scaling are perhaps the most often used preprocessing methods both for matrix and tensor data. In matrix data analysis there are certain traditional approaches for centering and scaling and those approaches actually help making sure that the preprocessing achieves what is expceted. In tensor analysis, it is slightly more complicated mainly because there are few traditions. Richard Harshman has written an excellent description of the common pitfalls in centering and scaling [150, 151]. Centering often serves two separate and independent purposes; to remove offsets in data and to make sure that the components are centered e.g. for subsequent regression problems. Not all types of centering will achieve these two goals. Imagine as an example, that a tensor follows a three-component CPD model plus an offset. Such data cannot be modelled by a three-component CPD model directly. Rather, a four-component model would be able to model the data. Upon centering, it is expected that the rank four data will now be rank three meaning that the offset information has been removed. Subtracting e.g. the overall average of the data would not have that effect [151]. It can be shown that only centering **across** one mode will be able to remove offsets. Centering across one mode means that the average of each column/row/tube is subtracted from that column/row/tube. Any other centering will introduce artefacts in the data that must then also be modelled. Likewise for scaling. Tensor data has to be scaled **within** a mode. That means that each slab of a three-way array has to be scaled by the same scalar. As for centering, scaling differently than within a mode will increase the rank artificially.

### 6.6.2 Fluorescence

In fluorescence excitation emission spectroscopy, each sample is excited at $K$ excitation wavelengths and the emission subsequently measured at $J$ emission wavelengths. Hence, for $I$ samples an $I \times J \times K$ tensor $\boldsymbol{\mathcal{T}}$ is obtained. If the samples contain, say $R$, chemical compounds that fluoresce, then the rank of the tensor should be $R$ under ideal conditions up to the noise of the measurements. That is, if the sample is fairly dilute and does not contain an excessive number of other chromophores that absorbs significantly [152, 19]. In practice, such data may contain artefacts that need to be handled before a chemically meaningful CPD model can be fitted. If the absorbance of the sample is too high, there may be inner filter effects that distort the signal. There are several methods available for correcting for this either explicitly or implicitly [153, 154, 155].

In addition to inner filter effects, it is common that FEEMs will have significant variation caused by Raman and Rayleigh scattering [154]. The Raman scattering is often of moderate size and for many applications, it can be removed by simply subtracting an FEEM of the solvent from each FEEM. The Rayleigh scattering (Figure 6.6) cannot be handled this way so usually those areas are removed by replacing the measurements with missing values or interpolating [156, 155].

The sample shown in Figure 6.7 comes from a dataset of 27 samples all containing varying

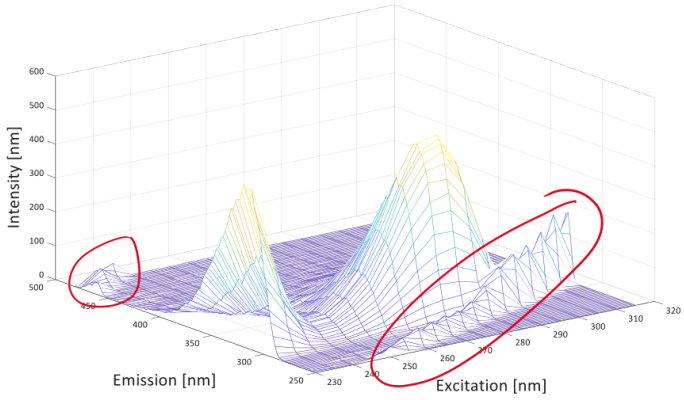



**Figure 6.6** A fluorescence excitation emission matrix (FEEM). Areas marked with red lines represent Rayleigh scattering.

concentrations of the four fluorophores Hydroquinone, Tryptophan, Phenylalanine, and Dopa. Since there are four chemical compounds, it is expected that a four-component CPD model would provide an adequate model of the data.

Indeed, a four-component CPD model has a so-called core consistency of 88% indicating a valid model [122]. However, several aspects seem suspicious. First of all, it seems that there may be problems with local minima. Refitting the model ten times, the fit varies between three distinct values: 99.8% variation explained, 96.6% and 95.5%. Only the best fitting of those qualifies for being the actual CPD model so the others have to be disregarded. Normally, local minima are not a huge problem for datasets that follow the CPD models well, but in this case where there is both a large amount of missing data and some outliers present, the algorithm apparently struggles. Investigating residuals and parameters, four outlying samples are identified and removed. The main reason for the outlying behavior is that the concentrations are quite high. Upon removing the four samples, a four-component model has a core consistency of 100%. Normally, it is advised to use models with the highest number of components with a sufficiently high core consistency [122].

It was investigated if the model was more stable and robust when using nonnegativity constraints. Some of the estimated fluorescence spectra in the unconstrained model were slightly negative. Not enough to be a significant issue, but oftentimes, imposing *nonnegativity* can also stabilize the model with respect to numerical problems. Indeed, a four-component model with nonnegativity on all parameters did not show any local minima and had a perfect core consistency. Furthermore, the estimated emission and excitation spectra looked very similar to what would be expected from prior knowledge. The five-component model has a low core consistency and some of the emission components come out identical which is not chemically meaningful. Hence, the four-component model seems a good candidate.

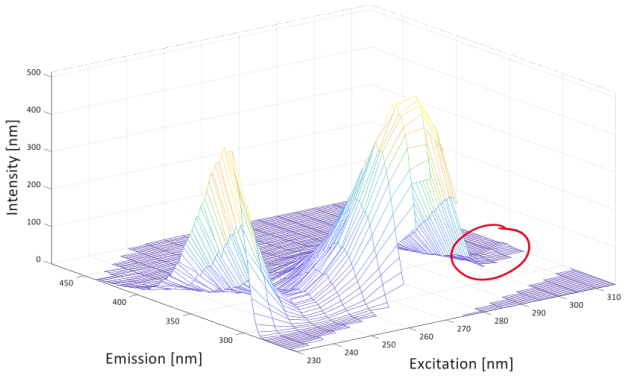



**Figure 6.7** The same FEEM as in Figure 1 after removal of Rayleigh scattering. Some traces of Raman scattering are visible.

To verify the model, we perform a splithalf analysis where the data is split in two parts in the sample mode [157]. A four-component CPD model is fitted to the first 13 samples and independently to the last 14 samples. If the model is correctly specified the components should be the same in the two models. In Figure 6.8, the results of the two models are shown together with the overall model. As can be seen, the four estimated emission (top) and excitation (bottom) spectra are almost exactly identical even though they are estimated from different sample sets. This is a very convincing diagnostic for assessing the validity of the model

As a final illustration of the ability to uncover the underlying chemistry, the scores are plotted in Figure 6.9. Each score is plotted against the known actual concentration of the corresponding chemical in each sample. As can be seen, the model is capable of recovering the concentrations up to a scaling; hence estimating the relative concentration of each compound.

### 6.6.3

### Chromatography

Gas Chromatograpy with Mass Spectrometric detection (GC-MS) is a very common tool in analytical chemistry e.g. for measuring hormones in food products, flavor compounds in wines or proteins in blood. In simpler cases, there is little need for much data analysis as the whole purpose of the chromatography is to ensure that different chemical constituents come out at different times. However as shown in Figure 6.10, sometimes the peaks of different chemicals are overlapping. Ideally, each chemical would be a baseline-resolved Gaussian curve but when compounds overlap, the traditional approaches for handling the data often fail.

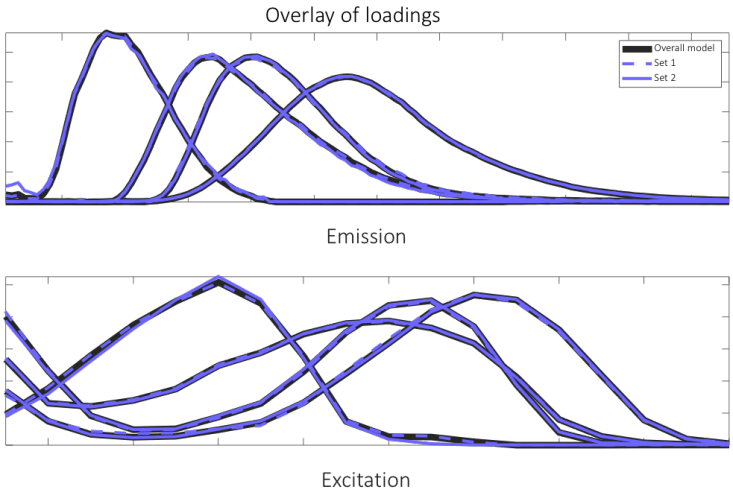



**Figure 6.8** The results of split-half analysis.please fix

For ideal chromatographic data, fitting a CPD model would allow to resolve overlapping data. Each CPD component would consist of a component in the elution mode giving the elution profile and in the spectral mode giving the pure mass spectrum of each analyte. The sample mode would then give the relative concentration of each chemical in each sample/experiment [158, 159]. However, the CPD model requires that the elution profile of each chemical compound keeps the same shape across different samples. This is almost never the case in chromatography. Due to retention time shifts, the elution profile will change slightly from sample to sample. This is also evident in Figure 6.10, where the peak at approximately time 21.7 minutes varies. Further, there are a number of minor peaks around 21.8-22 minutes and it is difficult to discern exactly how many.

The PARAFAC2 model has been shown repeatedly to provide a good model for chromatographic data and fitting the model to interval indicated in Figure 6.10; it turns out that there are as many as seven components needed for describing the data. In Figure 6.11, the elution mode components are shown. There are 44 samples in the dataset, hence there are 44 versions of each elution profile.

### 6.6.4

### Other applications

Tensor analysis has a long history in chemistry and there are many diverse fields of applications as also evidenced in older reviews [160]. The applications can be divided into typical groups. The first group is consisting of applications where hard modeling such as Beer's law is used to identify chemical information like pure spectra and concentrations.

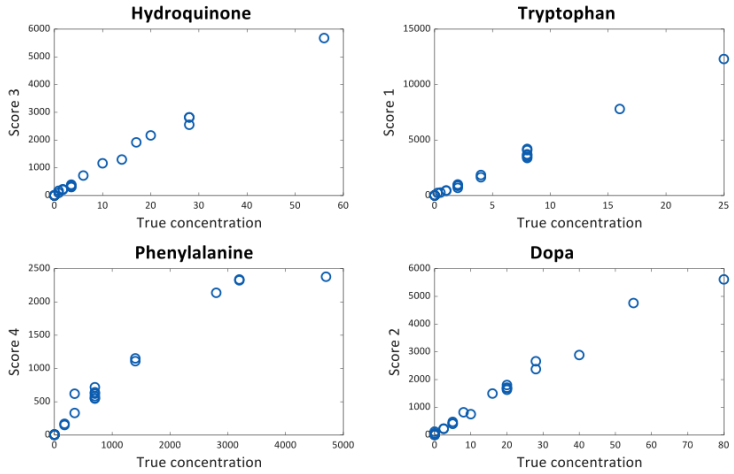



**Figure 6.9** The four score vectors of a CPD model plotted against the corresponding actual concentrations.

This can be used for untargeted approaches where many chemicals are being estimated at the same time [161] or in targeted approaches where one or a few compounds needs to be quantified [162]. The models used are mostly CPD and PARAFAC2 but also sometimes alternatives such as restricted Tucker3 models [163] or methods based on rank annihilation [164, 165, 166, 167, 168]. Especially CPD is useful e.g. for high-resolution nuclear magnetic resonance [169, 170, 171, 172, 173] as well as low-resolution magnetic resonance [174, 175]. Traditionally, CPD and variants have also been popular within electroencephalography [176, 177, 178]. For more exploratory purposes, it is common to use the Tucker3 model often followed by some types of rotations of either the core or the component matrices [33, 59]. Examples often come from environmental analysis [179, 180, 181] but variants of CPD are also used [182, 183]. Sensory profiling is a common approach for understanding human perception e.g. in food analysis. The traditional sensory profiling data is a three-way structure consisting of a number of assessors assessing a number of items with respect to a number of attributes. The data can be analyzed with both CPD and Tucker3, but the Tucker2 model is often preferred because the extended core array allows meaningful interactions between components [184]. In batch process monitoring or multivariate statistical process monitoring in general, the aim is to understand and operate production processes. In early days, both Tucker, CPD and even PARAFAC2 models were investigated [185, 186]. Nowadays though, the three-way data is often unfolded and analyzed as matrix data to better handle the complex dynamics that such data have. A third type of problem that occurs quite often in the chemical sciences is regression, which is also referred to as multivariate calibration. The classical problem is to replace a tedious and costly reference method with a prediction based on some more easily available data. The most popular algorithm for this is multi-way

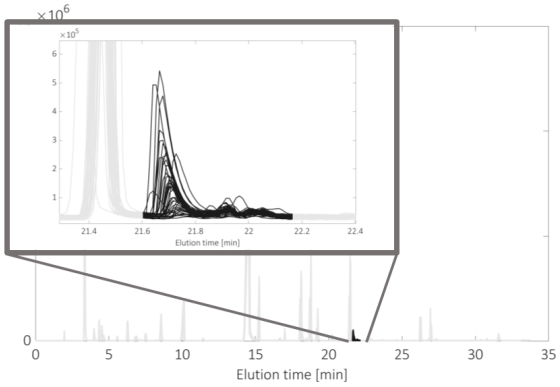



**Figure 6.10** Example of a set of samples measured by GC-MS. The mass spectrum is summed at each time point so that the measurements of each sample becomes a vector called a TIC – Total Ion Current chromatogram. In the dark part, a time interval with overlapping peaks is shown.

*partial least squares regression* [187], which has been used for a multitude of problems [188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 171, 198, 199]. An interesting alternative is the method SCREAM that combines the regression with the more flexible PARAFAC2 model [200].

## References


1 M. L., S. (2013) *Categories for the working mathematician*, vol. 5, Springer Science & Business Media.

2 Hackbusch, W. (2012) *Tensor Spaces and Numerical Tensor Calculus*, Series in Computational Mathematics, Springer, Berlin, Heidelberg.

3 Landsberg, J.M. (2012) *Tensors: Geometry and Applications*, *Graduate Studies in Mathematics*, vol. 128, AMS publ.

4 Comon, P. (2014) Tensors: a brief introduction. *IEEE Sig. Proc. Magazine*, **31** (3), 44–53. Special issue on BSS.

5 Bro, R. (2006) Review of multiway analysis in chemistry – 2000-2005. *Critical reviews in Analytical Chemistry*, **36**, 279–293.

6 Acar, E. and Yener, B. (2009) Unsupervised multiway data analysis: A literature survey. *IEEE transactions on knowledge and data engineering*, **21** (1), 6–20.

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

8 Comon, P. and Jutten, C. (eds) (2010) *Handbook of Blind Source Separation, Independent Component Analysis and Applications*, Academic Press, Oxford UK, Burlington USA. ISBN: 978-0-12-374726-6.

9 De Lathauwer, L. (2011) Blind separation of exponential polynomials and the decomposition of a tensor in rank-(lr,lr,1) terms. *SIAM J. Matrix Anal. Appl.*, **32** (4), 1451–1474.

10 Domanov, I. and De Lathauwer, L. (2016) Generic uniqueness of a structured matrix factorization and applications in blind source

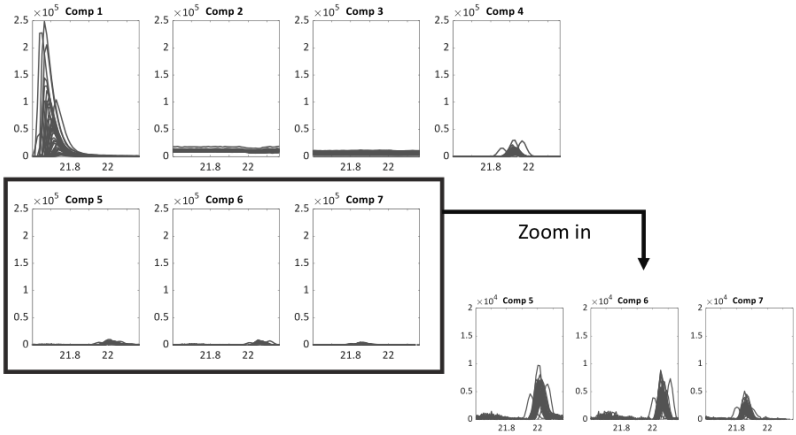



**Figure 6.11** Estimated elution profiles from a seven component PARAFAC2 model. Component 2 and 3 are describing baseline variation whereas the remaining five components are describing different chemical compounds.


separation. *IEEE Journal of Selected Topics in Signal Processing*, **10** (4), 701–711.

**11** Raimondi, F., Farias, R.C., Michel, O., and Comon, P. (2017) Wideband multiple diversity tensor array processing. *IEEE Trans. Sig. Proc.*, **65**, 5334–5346. Hal-01350549.

**12** Sahnoun, S., Comon, P., and Usevich, K. (2017) Multidimensional ESPRIT for damped and undamped signals: Algorithm, computations and perturbation analysis. *IEEE Trans. Sig. Proc.*, **65** (22), 5897–5910. Hal-01360438.

**13** Chiantini, L., Ottaviani, G., and Vannieuwenhoven, N. (2014) An algorithm for generic and low-rank specific identifiability of complex tensors. *SIAM J. matrix Ana. Appl.*, **35** (4), 1265–1287.

**14** Kruskal, J.B. (1977) Three-way arrays: Rank and uniqueness of trilinear decompositions. *Linear Algebra and Applications*, **18**, 95–138.

**15** Sidiropoulos, N.D. and Bro, R. (2000) On the uniqueness of multilinear decomposition of N-way arrays. *Jour. Chemo.*, **14**, 229–239.

**16** Stegeman, A. and Sidiropoulos, N. (2007) On Kruskal's uniqueness condition for the CP decomposition. *Lin. Alg. Appl.*, **420** (2-3), 540–552.

**17** Domanov, I. and Lathauwer, L.D. (2017) Canonical polyadic decomposition of third-order tensors: relaxed uniqueness conditions and algebraic algorithm. *Linear Algebra Appl.*, **513**, 342–375.

**18** Kiers, H.A.L. (2000) Towards a standardized notation and terminology in multiway analysis. *J. Chemometrics*, pp. 105–122.

**19** Bro, R. (1997) Parafac, tutorial and applications. *Chemom. Intel. Lab. Syst.*, **38**, 149–171.

**20** Cohen, J.E. (2015) About notations in multiway array processing. *arXiv:1511.01306*.

**21** Lathauwer, L.D., Moor, B.D., and Vandewalle, J. (2000) A multilinear singular value decomposition. *SIAM J. Matrix Ana. Appl.*, **21** (4), 1253–1278.

**22** Silva, V.D. and Lim, L.H. (2008) Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM J. Matrix Analysis Appl.*, **30** (3), 1084–1127.

**23** Kroonenberg, P.M. and Leeuw, J.D. (1980) Principal component analysis of three-mode data. *Psychometrika*, **45**, 69–97.

**24** Tucker, L.R. (1966) Some mathematical notes for three-mode factor analysis. *Psychometrika*, **31**, 279–311.

**25** Harshman, R.A. (1972) PARAFAC2: Mathematical and technical notes. *UCLA working papers in phonetics*, **22** (3044), 122 215.

**26** Kiers, H.A.L., ten Berge, J.M.F., and Bro, R.

(1999) PARAFAC2: Part I. a direct fitting algorithm. *J. Chemometrics*, **13**, 275–294.

**27** Comon, P., Berge, J.M.F.T., DeLathauwer, L., and Castaing, J. (2009) Generic and typical ranks of multi-way arrays. *Linear Algebra Appl.*, **430** (11–12), 2997–3007. Hal-00410058.

**28** Lim, L.H. and Comon, P. (2014) Blind multilinear identification. *IEEE Trans. Inf. Theory*, **60** (2), 1260–1280.

**29** Golub, G.H. and Loan, C.F.V. (1989) *Matrix computations*, The John Hopkins University Press.

**30** Bro, R. and Sidiropoulos, N.D. (1998) Least squares algorithms under unimodality and non-negativity constraints. *J. Chemometrics*, **12**, 223–247.

**31** Lim, L.H. and Comon, P. (2009) Nonnegative approximations of nonnegative tensors. *Jour. Chemometrics*, **23**, 432–441.

**32** Qi, Y., Comon, P., and Lim, L.H. (2016) Uniqueness of nonnegative tensor approximations. *IEEE Trans. Inf. Theory*, **62** (4), 2170–2183. ArXiv:1410.8129.

**33** Kiers, H.A.L. (1992) Tuckals core rotations and constrained Tuckals modelling. *Statistica Applicata*, **4** (4), 659–667.

**34** Comon, P. (1994) Tensor diagonalization, a useful tool in signal processing, in *IFAC-SYSID, 10th IFAC Symposium on System Identification*, vol. 1 (eds M. Blanke and T. Soderstrom), Copenhagen, Denmark, vol. 1, pp. 77–82.

**35** Martin, C.D.M. and van Loan, C. (2008) A Jacobi-type method for computing orthogonal tensor decompositions. *SIAM J. Matrix Anal. Appl.*, **30** (3), 1219–1232.

**36** Krijnen, W.P., Dijkstra, T.K., and Stegeman, A. (2008) On the non-existence of optimal solutions and the occurrence of degeneracy in the Candecomp/Parafac model. *Psychometrika*, **73** (3), 431–439.

**37** Uschmajew, A. (2010) Well-posedness of convex maximization problems on stiefel manifolds and orthogonal tensor product approximations. *Numerische Mathematik*, **115**, 309–331.

**38** Timmerman, M.E. and Kiers, H.A.L. (2002) Three-way component analysis with smoothness constraints. *Computational Statistics Data Analysis*, **440**, 447–470.

**39** Chen, Z., Chichocki, A., and Rutkowski, T.M. (2006) Constrained non-negative matrix factorization method for EEG alalysis in early detection of Alzheimer disease, in *ICASSP*, vol. V, Toulouse, vol. V, pp. 893–896.

**40** Berry, M.W., Browne, M., Langville, A.N., Pauca, V.P., and Plemmons, R.J. (2007) Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics Data Analysis*, **52** (1), 155–173.

**41** Cichocki, A., Zdunek, R., Phan, A.H., and Amari, S.I. (2009) *Nonnegative Matrix and Tensor Factorization*, Wiley.

**42** Veganzones, M., Cohen, J.E., Farias, R.C., Chanussot, J., and Comon, P. (2016) Nonnegative tensor CP decomposition of hyperspectral data. *IEEE Trans. Geoscience and Remote Sensing*, **54** (5), 2577–2588.

**43** Roemer, F., Galdo, G.D., and Haardt, M. (2014) Tensor-based algorithms for learning multidimensional separable dictionaries, in *ICASSP*, pp. 3963–3967.

**44** Boyer, R. and Haardt, M. (2016) Noisy compressive sampling based on block-sparse tensors: Performance limits and beamforming techniques. *IEEE Trans. Signal Processing*, **64** (23), 6075–6088.

**45** Sahnoun, S., Djermoune, E.H., Brie, D., and Comon, P. (2017) A simultaneous sparse approximation method for multidimensional harmonic retrieval. *Signal Processing, Elsevier*, **137**, 36–48.

**46** Cohen, J.E. and Gillis, N. (2018) Dictionary-based tensor canonical polyadic decomposition. *IEEE Trans. Signal Processing*, **66** (7), 1876–1889.

**47** Sahnoun, S. and Comon, P. (2015) Joint source estimation and localization. *IEEE Trans. Sig. Proc.*, **63** (10), 2485–2495.

**48** Qi, Y., Comon, P., and Lim, L.H. (2016) Semialgebraic geometry of nonnegative tensor rank. *SIAM J. Matrix Ana. Appl.*, **37** (4), 1556–1580. Hal-01763832.

**49** Mitchell, B.C. and Burdick, D.S. (1994) Slowly converging Parafac sequences: Swamps and two-factor degeneracies. *Jour. Chemometrics*, **8**, 155–168.

**50** Paatero, P. (2000) Construction and analysis of degenerate Parafac models. *Jour. Chemometrics*, **14**, 285–299.

**51** Rajih, M., Comon, P., and Harshman, R. (2008) Enhanced line search : A novel method to accelerate Parafac. *SIAM Journal on Matrix Analysis Appl.*, **30** (3), 1148–1171.

**52** Stegeman, A. (2006) Degeneracy in Candecomp/Parafac explained for $p \times p \times 2$ arrays of rank $p + 1$ or higher. *Psychometrika*, **71** (3), 483–501.

**53** Cichocki, A., Zdunek, R., and Amari, S.I. (2008) Nonnegative matrix and tensor factorization. *IEEE Sig. Proc. Mag.*, pp. 142–145.

**54** Zhou, G., Cichocki, A., Zhao, Q., and Xie, S. (2014) Nonnegative matrix and tensor factorizations : An algorithmic perspective. *Signal processing Magazine*, **31** (3), 54–65.

**55** Cohen, J.E., Cabral-Farias, R., and Comon, P. (2015) Fast decomposition of large nonnegative tensors. *IEEE Sig. Proc. Letters*, **22** (7), 862–866.

**56** Royer, J.P., Thirion-Moreau, N., Comon, P., Redon, R., and Mounier, S. (2015) A regularized nonnegative canonical polyadic decomposition algorithm with preprocessing for 3D fluorescence spectroscopy. *Jour. Chemometrics*, **29** (4), 253–265.

**57** Cabral-Farias, R., Comon, P., and Redon, R. (2014) Data mining by nonnegative tensor approximation, in *IEEE MLSP*, Reims, France. Hal-01077801.

**58** Phan, A.H. and Chichocki, A. (2008) Fast and efficient algorithms for nonnegative tucker decomposition, in *Int. Symp. Neural networks*, vol. LNCS 5264, Springer, Berlin, vol. LNCS 5264, pp. 772–782.

**59** Kiers, H.A.L. (1998) Recent developments in three-mode factor analysis: Constrained three-mode factor analysis and core rotations, in *Data Science, Classification, and Related Methods* (eds C.Hayashi and K.Yajima), Springer, Studies in Classification, Data Analysis, and Knowledge Organization, pp. 563–574.

**60** ten Berge, J.M.F. and Smilde, A.K. (2002) Non-triviality and identification of a constrained tucker3 analysis. *Journal of Chemometrics*, **16** (12), 609–612.

**61** Tomasi, G. and Bro, R. (2009) Multilinear models: Iterative methods, in *Comprehensive Chemometrics: Chemical and Biochemical Data Analysis*, vol. 2 (eds R. S.Brown and B.Walczak), Elsevier, chap. 22, pp. 411–451.

**62** Zhou, G., Cichocki, A., Zhao, Q., and Xie, S. (2015) Efficient nonnegative tucker decompositions: Algorithms and uniqueness. *IEEE Transactions on Image Processing*, **24** (12), 4990–5003.

**63** Hoyer, P.O. (2004) Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, **5**, 1457–1469.

**64** Morup, M., Hansen, L.K., and Arnfred, S.M. (2008) Algorithms for sparse nonnegative tucker decompositions. *Neural Computation*, **20** (8), 2112–2131.

**65** Anandkumar, A., Hsu, D., and Kakade, M.J.S. (2015) When are overcomplete topic models identifiable? uniqueness of tensor tucker decompositions with structured sparsity. *Jour. Machine Learning Research*, **16**, 2643–2694.

**66** Xu, Y. (2015) Alternating proximal gradient method for sparse nonnegative Tucker decomposition. *Mathematical Programming Computation*, **7** (1), 39–70. Arxiv:1302.2559.

**67** Bro, R., Harshman, R.A., Sidiropoulos, N.D., and Lundy, M.E. (2009) Modeling multi-way data with linearly dependent loadings. *J. Chemo.*, **23** (7-8), 324–340.

**68** De Lathauwer, L. (2008) Decompositions of a higher-order tensor in block terms–Part II: Definitions and uniqueness. *SIAM Journal on Matrix Analysis and Applications*, **30** (3), 1033–1066.

**69** Guo, X., Miron, S., Brie, D., and Stegeman, A. (2012) Uni-mode and partial uniqueness conditions for Candecomp/Parafac of three-way arrays with linearly dependent loadings. *SIAM J. Matrix Analysis Appl.*, **33**, 111–129.

**70** Sorensen, M., Domanov, I., and De Lathauwer, L. (2015) Coupled canonical polyadic decompositions and (coupled) decompositions in multilinear rank-(lr,n, lr,n, 1) terms — part ii: Algorithms. *SIAM J. Matrix Anal. Appl.*, **36** (3), 1015–1045.

**71** Sorensen, M. and De Lathauwer, L. (2015) Coupled canonical polyadic decompositions and (coupled) decompositions in multilinear rank-(lr,n, lr,n, 1) terms — part i: uniqueness. *SIAM J. Matrix Anal. Appl.*, **36** (2), 496–522.

**72** Lahat, D. and Jutten, C. (2018) A new link between joint blind source separation using second order statistics and the canonical polyadic decomposition, in *14th Int. Conf. on Latent Variable Analysis and Signal Separation (LVA-ICA)*, Springer, Univ. of Surrey, Guildford, UK.

**73** Caland, F., Miron, S., Brie, D., and Mustin, C. (2012) A blind sparse approach for

estimating constraint matrices in Paralind data models, in *20th EUSIPCO*, Eurasip, Bucharest, pp. 839–843.

**74** Cabral Farias, R., Cohen, J.E., and Comon, P. (2016) Exploring multimodal data fusion through joint decompositions with flexible couplings. *IEEE Trans. Sig. Proc.*, **64** (18), 4830–4844.

**75** Chatzichristos, C., Kofidis, E., and Theodoridis, S. (2017) PARAFAC2 and its block term decomposition analog for blind fMRI source unmixing, in *EUSIPCO*, Kos island, pp. 2081–2085.

**76** Hunyadi, B., Camps, D., Sorber, L., Paesschen, W.V., Vos, M.D., Huffel, S.V., and Lathauwer, L.D. (2014) Block term decomposition for modelling epileptic seizures. *EURASIP Journal on Advances in Signal Processing*, **139**, 1–19.

**77** Aldana, Y.R., Hunyadi, B., Reyes, E.J.M., Rodriguez, V.R., and Van Huffel, S. (2018) Nonconvulsive epileptic seizure detection in scalp eeg using multiway data analysis. *IEEE Journal of Biomedical and Health Informatics*.

**78** Coloigner, J., Karfoul, A., Albera, L., and Comon, P. (2014) Line search and trust region strategies for canonical decomposition of semi-nonnegative semi-symmetric 3rd order tensors. *Linear Algebra Appl.*, **450**, 334–374.

**79** Windig, W. and Antalek, B. (1997) Direct exponential curve resolution algorithm (decra). *Chemometrics and Intelligent Laboratory Systems*, **37** (2), 241–254.

**80** Helwig, N.E. (2017) Estimating latent trends in multivariate longitudinal data via parafac2 with functional and structural constraints. *Biometrical Journal*, **59** (4), 783–803.

**81** Dantas, C.F., Cohen, J.E., and Gribonval, R. (2019) Learning tensor-structured dictionaries with application to hyperspectral image denoising, in *2019 27th European Signal Processing Conference (EUSIPCO)*, IEEE, pp. 1–5.

**82** Cattell, R.B. (1944) "parallel proportional profiles" and other principles for determining the choice of factors by rotation. *Psychometrika*, **9** (4), 267–283.

**83** Hotelling, H. (1936) Relations between two sets of variates. *Biometrika*, **28**, 321–377.

**84** Smilde, A.K. and Kiers, H.A.L. (1999) Multiway covariates regression models. *J. Chemometrics*, **13** (1), 31–48.

**85** Smilde, A.K., Westerhuis, J.A., and de Jong, S. (2003) A framework for sequential multiblock component methods. *J. Chemometrics*, **17** (6), 323–337.

**86** Acar, E., Bro, R., and Smilde, A.K. (2015) Data fusion in metabolomics using coupled matrix and tensor factorizations. *Proceedings of the IEEE*, **103** (9), 1602–1620.

**87** Acar, E., Kolda, T.G., and Dunlavy, D.M. (2011) All-at-once optimization for coupled matrix and tensor factorizations. *CoRR*, **abs/1105.3422**.

**88** Acar, E., Papalexakis, E.E., Gürdeniz, G., Rasmussen, M.A., Lawaetz, A.J., Nilsson, M., and Bro, R. (2014) Structure-revealing data fusion. *BMC bioinformatics*, **15** (1), 239.

**89** Acar, E., Bro, R., and Smilde, A.K. (2015) Data fusion in metabolomics using coupled matrix and tensor factorizations. *Proceedings of the IEEE*, **103** (9), 1602–1620.

**90** Acar, E., Levin-Schwartz, Y., Calhoun, V.D., and Adali, T. (2017) Acmtf for fusion of multi-modal neuroimaging data and identification of biomarkers, in *Signal Processing Conference (EUSIPCO), 2017 25th European*, IEEE, pp. 643–647.

**91** De Lathauwer, L. and Kofidis, E. (2018) Coupled matrix-tensor factorizations—the case of partially shared factors, in *Proc. of the Asilomar Conference on Signals, Systems and Computers*, accepted.

**92** Harshman, R.A., Hong, S., and Lundy, M.E. (2003) Shifted factor analysis—part i: Models and properties. *Journal of chemometrics*, **17** (7), 363–378.

**93** Mørup, M., Hansen, L.K., Arnfred, S.M., Lim, L.H., and Madsen, K.H. (2008) Shift-invariant multilinear decomposition of neuroimaging data. *NeuroImage*, **42** (4), 1439–1450.

**94** Hong, S. (2005) Warped image factor analysis, in *1st IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing, 2005.*, pp. 121–124.

**95** Cohen, J.E., Cabral Farias, R., and Rivet, B. (2018) Curve registered coupled low rank factorization, in *14th LVA/ICA conference*, *LNCS*, vol. 10891, Springer, Univ. of Surrey, Guildford, UK, *LNCS*, vol. 10891.

**96** Tomasi, G., Van Den Berg, F., and Andersson, C. (2004) Correlation optimized warping and dynamic time warping as preprocessing methods for chromatographic data. *Journal of*

*Chemometrics*, **18** (5), 231–241.

**97** Savorani, F., Tomasi, G., and Engelsen, S.B. (2010) icoshift: A versatile tool for the rapid alignment of 1d nmr spectra. *Journal of Magnetic Resonance*, **202** (2), 190–202.

**98** Van Loan, C.F. (1976) Generalizing the singular value decomposition. *SIAM Journal on Numerical Analysis*, **13** (1), 76–83.

**99** Alter, O., Brown, P.O., and Botstein, D. (2000) Singular value decomposition for genome-wide expression data processing and modeling. *Proceedings of the National Academy of Sciences*, **97** (18), 10 101–10 106.

**100** Ponnapalli, S.P., Saunders, M.A., Van Loan, C.F., and Alter, O. (2011) A higher-order generalized singular value decomposition for comparison of global mrna expression from multiple organisms. *PloS one*, **6** (12), e28 072.

**101** Sankaranarayanan, P., Schomay, T.E., Aiello, K.A., and Alter, O. (2015) Tensor gsvd of patient-and platform-matched tumor and normal dna copy-number profiles uncovers chromosome arm-wide patterns of tumor-exclusive platform-consistent alterations encoding for cell transformation and predicting ovarian cancer survival. *PloS one*, **10** (4), e0121 396.

**102** Smith, S. and Karypis, G. (2015) Tensor-matrix products with a compressed sparse tensor, in *Proceedings of the 5th Workshop on Irregular Applications: Architectures and Algorithms*, pp. 1–7.

**103** Nelson, T., Rivera, A., Balaprakash, P., Hall, M., Hovland, P.D., Jessup, E., and Norris, B. (2015) Generating efficient tensor contractions for gpus, in *2015 44th International Conference on Parallel Processing*, IEEE, pp. 969–978.

**104** Shi, Y., Niranjan, U.N., Anandkumar, A., and Cecka, C. (2016) Tensor contractions with extended blas kernels on cpu and gpu, in *2016 IEEE 23rd International Conference on High Performance Computing (HiPC)*, IEEE, pp. 193–202.

**105** Abdelfattah, A., Baboulin, M., Dobrev, V., Dongarra, J., Earl, C., Falcou, J., Haidar, A., Karlin, I., Kolev, T., Masliah, I. *et al.* (2016) High-performance tensor contractions for gpus. *Procedia Computer Science*, **80**, 108–118.

**106** Smith, D. and Gray, J. (2018) opt_einsum-a python package for optimizing contraction order for einsum-like expressions. *Journal of Open Source Software*, **3** (26), 753.

**107** Springer, P. and Bientinesi, P. (2018) Design of a high-performance gemm-like tensor–tensor multiplication. *ACM Transactions on Mathematical Software (TOMS)*, **44** (3), 1–29.

**108** Sidiropoulos, N., Papalexakis, E.E., and Faloutsos, C. (2014) Parallel randomly compressed cubes. *IEEE Sig. Proc. Magazine*, **31** (5), 57–70. Special issue on Big data.

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

**110** Espig, M., Hackbusch, W., and Khachatryan, A. (2015) On the convergence of alternating least squares optimisation in tensor format representations. *arXiv preprint arXiv:1506.00062*.

**111** Comon, P., Luciani, X., and De Almeida, A.L.F. (2009) Tensor decompositions, alternating least squares and other tales. *Jour. Chemometrics*, **23** (7-8), 393–405.

**112** Nesterov, Y. (1983) A method of solving a convex programming problem with convergence rate o (1/k2). *Soviet Mathematics Doklady*, **27** (2), 372–376.

**113** Mitchell, D., Ye, N., and De Sterck, H. (2018) Nesterov Acceleration of Alternating Least Squares for Canonical Tensor Decomposition. *arXiv e-prints*, p. arXiv:1810.05846.

**114** Ang, A.M.S., Cohen, J.E., Gillis, N. *et al.* (2020) Extrapolated alternating algorithms for approximate canonical polyadic decomposition, in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, pp. 3147–3151.

**115** Schmidt, M., Le Roux, N., and Bach, F. (2017) Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, **162** (1-2), 83–112.

**116** Acar, E., Dunlavy, D.M., Kolda, T.G., and Mørup, M. (2010) Scalable tensor factorizations with missing data, in *Proceedings of the 2010 SIAM international conference on data mining*, SIAM, pp. 701–712.

**117** K.Huang, Sidiropoulos, N., and Liavas, A.P. (2016) A flexible and efficient algorithmic

framework for constrained matrix and tensor factorization. *IEEE Trans. Signal proc.*, **64** (19), 5052–5065.

**118** Paatero, P. (1999) The multilinear engine: A table-driven, least squares program for solving multilinear problems, including the n-way parallel factor analysis model. *J.Comput. Graph. Stat.*, **8** (4), 854–888.

**119** Stegeman, A. and Comon, P. (2009) Subtracting a best rank-1 approximation does not necessarily decrease tensor rank, in *EUSIPCO'09*, Glasgow, Scotland. Hal-00435877.

**120** Comon, P. (2009) Tensors vs matrices, usefulness and unexpected properties, in *15th IEEE Workshop on Statistical Signal Processing (SSP'09)*, Cardiff, UK, pp. 781–788. Keynote. hal-00417258.

**121** Stegeman, A. and Comon, P. (2010) Subtracting a best rank-1 approximation does not necessarily decrease tensor rank. *Linear Algebra Appl.*, **433** (7), 1276–1300. Hal-00512275.

**122** Bro, R. and Kiers, H.A.L. (2003) A new efficient method for determining the number of components in Parafac models. *Journal of Chemometrics*, **17** (5), 274–286.

**123** da Costa, J.P.C., Haardt, M., and Romer, F. (2008) Robust methods based on the HOSVD for estimating the model order in PARAFAC models, in *5th IEEE Sensor Array and Multichannel Signal Processing Workshop*, pp. 510–514.

**124** Han, X., Albera, L., Kachenoura, A., Senhadji, L., and Shu, H. (2017) Low rank canonical polyadic decomposition of tensors based on group sparsity, in *25th European Signal Processing Conference, EUSIPCO 2017, Kos, Greece, August 28 - September 2, 2017*, pp. 668–672.

**125** Wold, S., Ruhe, A., Wold, H., and Dunn, W.J. (1984) The collinearity problem in linear regression, the partial least squares (PLS) approach to generalized inverses. *SIAM J. Sci. Stat. Comput.*, **5** (3), 735–743.

**126** Smilde, A., Bro, R., and Geladi, P. (2004) *Multi-Way Analysis*, Wiley, Chichester UK.

**127** Luciani, X. and Albera, L. (2011) Semi-algebraic canonical decomposition of multi-way arrays and joint eigenvalue decomposition, in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, IEEE, pp. 4104–4107.

**128** De Lathauwer, L., De Moor, B., and Vandewalle, J. (2004) Computation of the canonical decomposition by means of a simultaneous generalized schur decomposition. *SIAM journal on Matrix Analysis and Applications*, **26** (2), 295–327.

**129** De Lathauwer, L. (2006) A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization. *SIAM journal on Matrix Analysis and Applications*, **28** (3), 642–666.

**130** Domanov, I. and Lathauwer, L.D. (2014) Canonical polyadic decomposition of third-order tensors: Reduction to generalized eigenvalue decomposition. *SIAM Journal on Matrix Analysis and Applications*, **35** (2), 636–660.

**131** Sanchez, E., Ramos, L.S., and Kowalski, B.R. (1987) Generalized rank annihilation method : I. application to liquid chromatography—diode array ultraviolet detection data. *Journal of Chromatography A*, **385**, 151–164.

**132** Ramos, L.S., Sanchez, E., and Kowalski, B.R. (1987) Generalized rank annihilation method : II. analysis of bimodal chromatographic data. *Journal of Chromatography A*, **385**, 165–180.

**133** Booksh, K.S., Lin, Z., Wang, Z., and Kowalski, B.R. (1994) Extension of trilinear decomposition method with an application to the flow probe sensor. *Analytical Chemistry*, **66**, 2561–2569.

**134** Smilde, A.K., Tauler, R., Saurina, J., and Bro, R. (1999) Calibration methods for complex second-order data. *Analytica Chimica Acta*, **398**, 237–251.

**135** Leurgans, S. and Ross, R.T. (1992) Multilinear models: Applications in spectroscopy. *Statistical Sciences*, **17** (3), 289–319.

**136** Leurgans, S., Ross, R.T., and Abel, R.B. (1993) A decomposition for three-way arrays. *SIAM Jour. Matrix Anal. Appl.*, **14** (4), 1064–1083.

**137** Haskell, K.H. and Hanson, R.J. (1981) An algorithm for linear least squares problems with equality and nonnegativity constraints. *Mathematical Programming*, **21**, 98–118.

**138** Lawson, C.L. and Hanson, R.J. (1995) *Solving least squares problems*, vol. 15, Siam.

**139** Bro, R. and Jong, S.D. (1997) A fast

non-negativity-constrained least squares algorithm. *J. Chemometrics*, **11** (5), 393–401.

**140** Gillis, N. and Glineur, F. (2012) Accelerated multiplicative updates and hierarchical als algorithms for nonnegative matrix factorization. *Neural computation*, **24** (4), 1085–1105.

**141** Paatero, P. (1997) A weighted non-negative least squares algorithm for three-way 'Parafac' factor analysis. *Chemometrics and Intelligent Laboratory Systems*, **38** (2), 223–242.

**142** Fu, X., Gao, C., Wai, H.T., and Huang, K. (2019) Block-randomized stochastic proximal gradient for low-rank tensor factorization. *arXiv preprint arXiv:1901.05529*.

**143** Vervliet, N., Debals, O., Sorber, L., Van Barel, M., and De Lathauwer, L. (2016), Tensorlab 3.0. Available online, `https://www.tensorlab.net`.

**144** Vervliet, N., De Lathauwer, L., and Cocchi, M. (2018) Numerical optimization based algorithms for data fusion. *Data Fusion Methodology and Applications, M. Cocchi, Ed. Elsevier*, pp. 693–697.

**145** Grasedyck, L., Kressner, D., and Tobler, C. (2013) A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*, **36** (1), 53–78.

**146** Hong, D., Kolda, T.G., and Duersch, J.A. (2018) *Generalized Canonical Polyadic Tensor Decomposition*. ArXiv:1808.07452.

**147** Martens, H. and Stark, E. (1991) Extended multiplicative signal correction and spectral interference subtraction: new preprocessing methods for near infrared spectroscopy. *J. Pharma. Biomed. Anal.*, **9** (8), 625–635.

**148** Heraud, P., Wood, B.R., Beardall, J., and McNaughton, D. (2006) Effects of pre-processing of raman spectra on in vivo classification of nutrient status of microalgal cells. *J. Chemometrics*, **20** (5), 193–197.

**149** Goodacre, R., Broadhurst, D., Smilde, A.K. *et al.* (2007) Proposed minimum reporting standards for data analysis in metabolomics. *Metabolomics*, **3** (3), 231–241.

**150** Harshman, R.A. and Lundy, M.E. (1984) Data preprocessing and the extended Parafac model, in *Research Methods for Multimode Data Analysis* (eds J.H. H. G. Law C. W. Snyder and R.P. McDonald), Praeger, New York, pp. 216–284. HarsL84:rmmda centering scaling preprocessing.

**151** Bro, R. and Smilde, A.K. (2003) Centering and scaling in component analysis. *J. Chemometrics*, **17** (1), 16–33.

**152** Lee, C., Kim, K., and Ross, R.T. (1991) Trilinear analysis for the resolution of overlapping fluoresecence spectra. *Korean Biochem. J.*, **24**, 374–379.

**153** Kubista, M., Sjoback, R., Eriksson, S., and Albinsson, B. (1994) Experimental correction for the inner-filter effect in fluorescence spectra. *Analyst*, **119**, 417–419.

**154** Lakowicz, J.R. (1999) *Principles of Fluorescence Spectroscopy*, Kluwer Academic, New York.

**155** Cohen, J.E., Comon, P., and Luciani, X. (2016) Correcting inner filter effects, a non multilinear tensor decomposition method. *Chemometrics Intel. Lab. Syst.*, **150**, 29–40.

**156** Elcoroaristizabal, S., Bro, R., Garcia, J.A., and Alonso, L. (2015) Parafac models of fluorescence data with scattering: A comparative study. *Chemo. Intel. Lab. Syst.*, **142**, 124–130.

**157** Harshman, R.A. and DeSarbo, W.S. (1984) An application of parafac to a small sample problem, demonstrating preprocessing, orthogonality constraints, and split-half diagnostic techniques. *Research Methods for Multimode Data Analysis p*, pp. 602–642.

**158** Amigo, J.M., Skov, T., Coello, J., Maspoch, S., and Bro, R. (2008) Solving gc-ms problems with PARAFAC2. *Trac-Trends in Analytical Chemistry*, **27** (8), 714–725.

**159** Amigo, J.M., Skov, T., and Bro, R. (2010) Chromathography: Solving chromatographic issues with mathematical models and intuitive graphics. *Chemical Reviews*, **110**, 4582–4605.

**160** Bro, R., J. Workman, J., Mobley, P., and Kowalski, B.R. (1997) Review of chemometrics applied to spectroscopy: 1985-1995, part iii: Multi-way analysis. *Applied Spectroscopy Reviews*, **32**, 237–261.

**161** Khakimov, B., Amigo, J.M., Bak, S., and Engelsen, S.B. (2012) Plant metabolomics: resolution and quantification of elusive peaks in liquid chromatography–mass spectrometry profiles of complex plant extracts using multi-way decomposition methods. *Journal of Chromatography A*, **1266**, 84–94.

**162** Garcia, I., Sarabia, L., Ortiz, M.C., and Aldama, J.M. (2004) Three-way models and detection capability of a gas

chromatography-mass spectrometry method for the determination of clenbuterol in several biological matrices: the 2002/657/ec european decision. *Analytica Chimica Acta*, **515** (1), 55–63.

**163** Smilde, A.K., Wang, Y., and Kowalski, B.R. (1994) Theory of medium-rank second-order calibration with restricted tucker models. *Journal of Chemometrics*, **8**, 21–36.

**164** Hayashi, C. and Hayashi, F. (1982) A new algorithm to solve Parafac-model. *Behaviormetrika*, **11**, 49–60.

**165** Wilson, B.E., Sanchez, E., and Kowalski, B.R. (1989) An improved algorithm for the generalized rank annihilation method. *Journal of Chemometrics*, **3**, 493–498.

**166** Gerritsen, M.J.P., Tanis, H., Vandeginste, B.G.M., and Kateman, G. (1992) Generalized rank annihilation factor analysis, iterative target transformation factor analysis, and residual bilinearization for the quantitative analysis of data from liquid chromatography with photodiode array detection. *Analytical Chemistry*, **64**, 2042–2056.

**167** Faber, N.M., Buydens, L.M.C., and Kateman, G. (1994) Generalized rank annihilation method. i: derivation of eigenvalue problems. *Journal of Chemometrics*, **8**, 147–154.

**168** Faber, N.M., Buydens, L.M.C., and Kateman, G. (1994) Generalized rank annihilation method. ii: bias and variance in the estimated eigenvalues. *Journal of Chemometrics*, **8**, 181–203.

**169** Orekhov, V.Y., Ibraghimov, I.V., and Billeter, M. (2001) MUNIN: A new approach to multi-dimensional NMR spectra interpretation. *Journal of Biomolecular NMR*, **20** (1), 49–60.

**170** Gutmanas, A., Jarvoll, P., Orekhov, V.Y., and Billeter, M. (2002) Three-way decomposition of a complete 3D N-15-NOESY-HSQC. *Journal of Biomolecular NMR*, **24** (3), 191–201.

**171** Dyrby, M., Petersen, M., Whittaker, A.D. *et al.* (2005) Analysis of lipoproteins using 2D diffusion-edited nmr spectroscopy and multi-way chemometrics. *Analytica Chimica Acta*, **531**, 209–216.

**172** Jansen, J.J., Bro, R., Hoefsloot, H.C.J., van den Berg, F., Westerhuis, J.A., and Smilde, A.K. (2008) PARAFASCA: ASCA combined with Parafac for the analysis of metabolic fingerprinting data. *Journal of Chemometrics*, **22**, 114–121.

**173** Bro, R., Viereck, N., Toft, M. *et al.* (2010) Mathematical chromatography solves the cocktail party effect in mixtures using 2D spectra and Parafac. *Trends in Analytical Chemistry*, **29** (4), 281–284.

**174** Engelsen, S.B. and Bro, R. (2003) Powerslicing. *Journal of Magnetic Resonance*, **163** (1), 192–197.

**175** Engelsen, S.B., Pedersen, H.T., and Bro, R. (2006) Direct exponential curve resolution by slicing, in *Modern Magnetic Resonance* (ed. G.A. Webb), Springer, Berlin, Heidelberg, New York, pp. 1823–1830.

**176** Mørup, M., Hansen, L.K., Herrmann, C.S., Parnas, J., and Arnfred, S.M. (2006) Parallel factor analysis as an exploratory tool for wavelet transformed event-related EEG. *Neuroimage*, **29** (3), 938–947.

**177** Acar, E., Aykut-Bingol, C., Bingol, H., Bro, R., and Yener, B. (2007) Multiway analysis of epilepsy tensors. *Bioinformatics*, **23** (13), 10–18.

**178** Becker, H., Albera, L., Comon, P. *et al.* (2014) EEG extended source localization: Tensor-based vs conventional methods. *NeuroImage*, **96**, 143–157.

**179** Basford, K.E., Kroonenberg, P.M., Cooper, M., and Hammer, G.L. (1996) Three-mode analytical methods for crop improvement programs, in *Plant adaptation and crop improvement*, Int. Rice Research Institute, chap. 14, pp. 291–305.

**180** Stanimirova, I., Walczak, B., Massart, D.L., Simeonov, V., Saby, C.A., and Crescenzo, E.D. (2004) Statis, a three-way method for data analysis. application to environmental data. *Chemometrics and Intelligent Laboratory Systems*, **73** (2), 219–233.

**181** Pere-Trepat, E., Ginebreda, A., and Tauler, R. (2007) Comparison of different multiway methods for the analysis of geographical metal distributions in fish, sediments and river waters in catalonia. *Chemometrics and Intelligent Laboratory Systems*, **88** (1), 69–83.

**182** Paatero, P. (1996) A weighted nonnegative least squares algorithm for three-way parafac factor analysis, in *2nd Int. Chemo. Int. Conf. (INCINC'96)*.

**183** Hopke, P., Xie, Y., and Paatero, P. (1999) Mixed multiway analysis of airborne particle composition data. *Journal of Chemometrics*, **13**, 343–352.

**184** Nœs, T. and Kowalski, B.R. (1989) Predicting sensory profiles from external instrumental measurements. *Food Quality and Preference*, **1** (4-5), 135–147.

**185** Louwerse, D.J. and Smilde, A.K. (2000) Multivariate statistical process control of batch processes based on three-way models. *Chemical Engineering Science*, **55** (7), 1225–1235.

**186** Wise, B.M., Gallagher, N.B., and Martin, E.B. (2001) Application of Parafac2 to fault detection and diagnosis in semiconductor etch. *Journal of Chemometrics*, **15** (4), 285–298.

**187** Bro, R. (1996) Multiway calibration, multilinear pls. *Jour. Chemometrics*, **10**, 47–61.

**188** Wittrup, C. (2000) Comparison of chemometric methods for classification of fungal extracts based on rapid fluorescence spectroscopy. *Journal of Chemometrics*, **14** (5-6), 765–776.

**189** Zampronio, C.G., Gurden, S.P., Moraes, L.A.B., Eberlin, M.N., Smilde, A.K., and Poppi, R.J. (2002) Direct sampling tandem mass spectrometry (ms/ms) and multiway calibration for isomer quantitation. *Analyst*, **127** (8), 1054–1060.

**190** Nilsson, J., de Jong, S., and Smilde, A.K. (1997) Multiway calibration in 3D QSAR. *Journal of Chemometrics*, **11**, 511–524.

**191** Heimdal, H., Bro, R., Larsen, L.M., and Poll, L. (1997) Prediction of polyphenol oxidase activity in model solutions containing various combinations of chlorogenic acid, (-)-epicatechin, o2, CO2, temperature and pH by multiway analysis. *Journal of Agricultural and Food Chemistry*, **45** (7), 2399–2406.

**192** Coello, J., Maspoch, S., and Villegas, N. (2000) Simultaneous kinetic-spectrophotometric determination of levodopa and benserazide by bi- and three-way partial least squares calibration. *Talanta*, **53** (3), 627–637.

**193** Hasegawa, K., Arakawa, M., and Funatsu, K. (1999) 3D-QSAR study of insecticidal neonicotinoid compounds based on 3-way partial least squares model. *Chemometrics and Intelligent Laboratory Systems*, **47**, 33–40.

**194** de la Pena, A.M., Mansilla, A.E., Gomez, D.G., Olivieri, A.C., and Goicoechea, H.C. (2003) Interference-free analysis using three-way fluorescence data and the parallel factor model. determination of fluoroquinolone antibiotics in human serum. *Analytical Chemistry*, **75** (11), 2640–2646.

**195** Tang, K.L. and Li, T.H. (2003) Comparison of different partial least-squares methods in quantitative structure-activity relationships. *Analytica Chimica Acta*, **476** (1), 85–92.

**196** Ni, Y.N., Huang, C.F., and Kokot, S. (2004) Application of multivariate calibration and artificial neural networks to simultaneous kinetic-spectrophotometric determination of carbamate pesticides. *Chemometrics and Intelligent Laboratory Systems*, **71** (2), 177–193.

**197** Bergant, K. and Kajfez-Bogataj, L. (2005) N-pls regression as empirical downscaling tool in climate change studies. *Theoretical And Applied Climatology*, **81** (1-2), 11–23.

**198** Durante, C., Cocchi, M., Grandi, M., Marchetti, A., and Bro, R. (2006) Application of N-PLS to gas chromatographic and sensory data of traditional balsamic vinegars of modena. *Chemometrics and Intelligent Laboratory Systems*, **83**, 54–65.

**199** Chow, E., Ebrahimi, D., Gooding, J.J., and Hibbert, D.B. (2006) Application of N-PLS calibration to the simultaneous determination of Cu2+, Cd2+ and Pb2+ using peptide modified electrochemical sensors. *Analyst*, **131** (9), 1051–1057.

**200** Marini, F. and Bro, R. (2013) SCREAM: A novel method for multi-way regression problems with shifts and shape changes in one mode. *Chemometrics and Intelligent Laboratory Systems*, **129**, 64–75.