NonNegative 3-Way tensor Factorzation taking into account Possible Missing Data

Jean-Philip Royer, Nadège Thirion-Moreau, Pierre Comon

NONNEGATIVE 3-WAY TENSOR FACTORIZATION TAKING INTO ACCOUNT POSSIBLE MISSING DATA

Jean-Philip Royer [1,3] *Nadège Thirion-Moreau* [3] *Pierre Comon* [2]

[1] I3S, CNRS UMR7271, 2000 route des Lucioles, BP.121, F-06903, Sophia Antipolis, jph.royer@gmail.com
[2] GIPSA-Lab, CNRS UMR5216, Grenoble Campus BP.46 F-38402 St Martin d'Heres Cedex, p.comon@ieee.org
[3] ISITV, LSIS, CNRS UMR7296, Université du Sud Toulon Var, F-83957, La Garde Cédex, France, Tel/Fax: +33 4 9414 2456/2671 or 2538 thirion@univ-tln.fr

ABSTRACT

This paper deals with the problem of incomplete data i.e. data with missing, unknown or unreliable values, in the polyadic decomposition of a nonnegative three-way tensor. The main advantage of the nonnegativity constraint is that the approximation problem becomes well posed. To tackle simultaneously these two problems, we suggest the use of a weighted least square cost function whose weights are gradually modified through the iterations. Moreover, the nonnegative nature of the loading matrices is taken into account directly in the problem parameterization. Then, the three gradient components can be explicitly derived allowing to efficiently implement the CP decomposition using standard optimization algorithms. In our case, we focus on the conjugate gradient and the BFGS algorithms. Finally, the good behaviour of the proposed approaches and their robustness versus possible model errors is illustrated through computer simulations in the context of data analysis.

Index Terms— Data analysis; Conjugate gradient; CanDecomp; Parafac; Canonical Polyadic (CP); tensor decomposition

1. INTRODUCTION

In many applications where multi-way arrays are considered, nonnegative data need to be handled, for example in hyperspectral imaging [16], in fluorescence analysis [15], in computer vision or in biomedical signal processing. Some solutions have been proposed in the literature to take into account this constraint see *e.g.* [3, 4, 10, 13, 14]. In addition, taking into account the nonnegativity nature of the data turns the decomposition problem into a well-posed problem [8]. On the other hand, during the acquisition process, it may happen that some measures are missing or unreliable, due to a sensor problem for example, so that data are incomplete. Solutions have been suggested to overcome this other problem too [1, 12]. In this work, we focus on nonnegative 3-way arrays with possible missing data. In [13, 14], we have already proposed two different approaches to take into account the nonnegativity constraint. Here, we present new algorithms whose aim is to handle the possibility of missing values too. Based on the use of a modified weighted least square cost function (whose weights will vary through the iterations) and a particular parameterization of the loading matrices, gradient components can be explicitly calculated. Then, standard optimization algorithms can be executed. We subsequently focus on a Quasi-Newton (Broyden-Fletcher-Goldfarb-Shanno implementation - BFGS) and the conjugate gradient algorithms. Concerning the choice of the stepsize, we suggest to use either an Enhanced Line Search (ELS) or a backtracking method. Computer simulations illustrate the algorithms in the context of data analysis; they are compared with the method introduced in [1].

2. PROBLEM STATEMENT

A tensor can be represented by a N-way array in a chosen basis. We focus on third-order tensors, say $\mathbf{X} \in \mathbb{R}^{I\times J\times K}$. Each entry of $\mathbf{X}$ is denoted by x_{ijk}. Tensors always admit the following trilinear decomposition, also known as the triadic decomposition [6] of $\mathbf{X}$ into a sum of rank-1 tensors:

$$x_{ijk} = \sum_{f=1}^{F} a_{if} b_{jf} c_{kf}, \tag{1}$$

where the three matrices $\mathbf{A} = (a_{if}) = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_F] \in \mathbb{R}^{I\times F}$, $\mathbf{B} = (b_{jf}) = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_F] \in \mathbb{R}^{J\times F}$, $\mathbf{C} = (c_{kf}) = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_F] \in \mathbb{R}^{K\times F}$ are the so-called *loading matrices*, whose F columns are the *loading factors*. F stands for a large enough integer. The minimum F that can be found such that the above equality remains valid is called the *tensor rank* and the decomposition is then named CP, which stands for Canonical Polyadic (or for Candecomp/Parafac). We shall also assume the more compact notation of (1): $\mathbf{X} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ where $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] = \sum_{f=1}^{F} \mathbf{a}_f \circ \mathbf{b}_f \circ \mathbf{c}_f$ and "$\circ$" represents an outer product of vectors. Three equivalent matrix writings of (1) also exist:

$$\mathbf{X}_{(1)}^{I,KJ} = \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T, \tag{2}$$

$$\mathbf{X}_{(2)}^{J,KI} = \mathbf{B}(\mathbf{C} \odot \mathbf{A})^T, \tag{3}$$

$$\mathbf{X}_{(3)}^{K,JI} = \mathbf{C}(\mathbf{B} \odot \mathbf{A})^T, \tag{4}$$

where $\mathbf{X}_{(1)}^{I,KJ}$, $\mathbf{X}_{(2)}^{J,KI}$ and $\mathbf{X}_{(3)}^{K,JI}$ are $I \times KJ$, $J \times KI$ and $K \times JI$ matrices obtained by unfolding the tensor $\mathbf{X}$ in the first, second and third modes, respectively. Operator $\odot$ stands for the Khatri-Rao (column-wise Kronecker) product.

2.1. CP decomposition of 3-way arrays via ALS

A classical way to determine the three loading matrices consists in minimizing the following cost function with respect to three loading matrices, alternately:

$$\begin{aligned}\mathcal{H}(\mathbf{A},\mathbf{B},\mathbf{C}) &= \|\mathbf{X}_{(1)}^{I,KJ} - \mathbf{A}(\mathbf{C}\odot\mathbf{B})^T\|_F^2 \\ &= \|\mathbf{X}_{(2)}^{J,KI} - \mathbf{B}(\mathbf{C}\odot\mathbf{A})^T\|_F^2 = \|\mathbf{X}_{(3)}^{K,JI} - \mathbf{C}(\mathbf{B}\odot\mathbf{A})^T\|_F^2\end{aligned} \quad (5)$$

where $\|.\|_F$ is the Frobenius norm. When performing the CP decomposition, the tensor rank F is assumed to be known and must therefore have been estimated first.

2.2. Nonnegative 3-way array factorization

Unconstrained CP decomposition leads to an ill-posed problem and the estimation of the loading matrices can become unstable. The symptoms of *degenerate array* [9, 5] occur when collinear columns with opposite signs are calculated, in such a way that their contributions cancel each other. Moreover, it is wise in some applications to consider the nonnegative nature of the data that are processed. Spectra and images are the most common examples.

Several works have been dedicated to that problem and solutions have been found to take into account the nonnegativity constraint [3, 4, 10, 13, 14]. In [13], the way to constrain the loading matrices to have nonnegative entries is via their parameterization, without modifying the cost function. This kind of parameterization has been previously used in nonnegative matrix factorization (NMF) problems [4, 7]. The matrix loadings are defined as $\mathbf{A}' = \mathbf{A} \boxdot \mathbf{A}$ and the same for $\mathbf{B}'$ and $\mathbf{C}'$, where $\boxdot$ is the Hadamard entry-wise multiplication for two matrices with the same dimensions, i.e. $(\mathbf{T} \boxdot \mathbf{T})_{ij} = t_{ij}^2$. This leads to the following cost function, where $\boldsymbol{\delta}_{(1)}$, $\boldsymbol{\delta}_{(2)}$ and $\boldsymbol{\delta}_{(3)}$ simplify the expressions:

$$\begin{aligned}\mathcal{F}(\mathbf{A},\mathbf{B},\mathbf{C}) &= \mathcal{H}(\mathbf{A}\boxdot\mathbf{A},\mathbf{B}\boxdot\mathbf{B},\mathbf{C}\boxdot\mathbf{C}) \\ &= \|\mathbf{X}_{(1)}^{I,KJ} - (\mathbf{A}\boxdot\mathbf{A})\left[(\mathbf{C}\boxdot\mathbf{C})\odot(\mathbf{B}\boxdot\mathbf{B})\right]^T\|_F^2 = \|\boldsymbol{\delta}_{(1)}\|_F^2 \\ &= \|\mathbf{X}_{(2)}^{J,KI} - (\mathbf{B}\boxdot\mathbf{B})\left[(\mathbf{C}\boxdot\mathbf{C})\odot(\mathbf{A}\boxdot\mathbf{A})\right]^T\|_F^2 = \|\boldsymbol{\delta}_{(2)}\|_F^2 \\ &= \|\mathbf{X}_{(3)}^{K,JI} - (\mathbf{C}\boxdot\mathbf{C})\left[(\mathbf{B}\boxdot\mathbf{B})\odot(\mathbf{A}\boxdot\mathbf{A})\right]^T\|_F^2 = \|\boldsymbol{\delta}_{(3)}\|_F^2\end{aligned} \quad (6)$$

Considering a function $\mathcal{I}$ depending on three matrices $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$, its matrix gradient components are denoted by $\nabla_{\mathbf{X}}\mathcal{I}(\mathbf{X},\mathbf{Y},\mathbf{Z}) = \partial\mathcal{I}(\mathbf{X},\mathbf{Y},\mathbf{Z})/\partial\mathbf{X}$, where $\partial\cdot/\partial\mathbf{X}$ denotes the matrix of partial derivatives with respect to the entries of $\mathbf{X}$; the notation being similar for $\mathbf{Y}$ and $\mathbf{Z}$. We have given in [13] the expressions of these matrix gradients:

$$\begin{aligned}\nabla_{\mathbf{A}}\mathcal{F}(\mathbf{A},\mathbf{B},\mathbf{C}) &= 4\mathbf{A}\boxdot\left((-\boldsymbol{\delta}_{(1)})\left[(\mathbf{C}\boxdot\mathbf{C})\odot(\mathbf{B}\boxdot\mathbf{B})\right]\right), \\ \nabla_{\mathbf{B}}\mathcal{F}(\mathbf{A},\mathbf{B},\mathbf{C}) &= 4\mathbf{B}\boxdot\left((-\boldsymbol{\delta}_{(2)})[(\mathbf{C}\boxdot\mathbf{C})\odot(\mathbf{A}\boxdot\mathbf{A})]\right), \\ \nabla_{\mathbf{C}}\mathcal{F}(\mathbf{A},\mathbf{B},\mathbf{C}) &= 4\mathbf{C}\boxdot\left((-\boldsymbol{\delta}_{(3)})[(\mathbf{B}\boxdot\mathbf{B})\odot(\mathbf{A}\boxdot\mathbf{A})]\right).\end{aligned} \quad (7)$$

3. HOW TO DEAL WITH MISSING DATA ?

3.1. A weighted cost function

As mentioned before, tensors resulting from experimental measures can lack some values. Yet, it is interesting to still be able to perform the decomposition ignoring those values. The previous works on this subject have consisted in introducing a tensor of weights, say $\mathbf{W}$ with the same size as the data tensor ($\mathbf{W} \in \mathbb{R}^{+I\times J\times K}$) and whose entries equal either 0 when a value is missing or 1 in the other case [1][12]. In our case, $\mathbf{W}$ entries are no more assumed to be binary. Denoting by $\mathbf{W}_{(1)}^{I,KJ}$ the unfolding of the tensor $\mathbf{W}$ in the first mode, the cost function expressed in the first mode is now given by:

$$\begin{aligned}&\mathcal{G}(\mathbf{A},\mathbf{B},\mathbf{C}) \\ &= \|\mathbf{W}_{(1)}^{I,KJ}\boxdot\left(\mathbf{X}_{(1)}^{I,KJ} - (\mathbf{A}\boxdot\mathbf{A})((\mathbf{C}\boxdot\mathbf{C})\odot(\mathbf{B}\boxdot\mathbf{B}))^T\right)\|_F^2 \\ &= \|\mathbf{W}_{(1)}^{I,KJ}\boxdot\delta_{(1)}\|_F^2 = \|\boldsymbol{\beta}_{(1)}\|_F^2\end{aligned} \quad (8)$$

We have the same kind of expression for the two other modes:

$$\mathcal{G}(\mathbf{A},\mathbf{B},\mathbf{C}) = \|\mathbf{W}_{(2)}^{J,KI}\boxdot\delta_{(2)}\|_F^2 = \|\boldsymbol{\beta}_{(2)}\|_F^2 \quad (9)$$

$$= \|\mathbf{W}_{(3)}^{K,JI}\boxdot\delta_{(3)}\|_F^2 = \|\boldsymbol{\beta}_{(3)}\|_F^2 \quad (10)$$

Once again, $\beta_{(i)}$, for $i = 1, ..., 3$ are introduced to shorten the expressions. Using $\|\mathbf{A}\|_F^2 = \mathsf{trace}\{\mathbf{A}^T\mathbf{A}\} = \langle\mathbf{A},\mathbf{A}\rangle$, where $\langle.\rangle$ is the Frobenius scalar product, (8) is rewritten:

$$\begin{aligned}&\langle\boldsymbol{\beta}_{(1)},\boldsymbol{\beta}_{(1)}\rangle = \mathsf{trace}\{\boldsymbol{\beta}_{(1)}^T\boldsymbol{\beta}_{(1)}\} = \\ &\mathsf{trace}\left\{\left[\mathbf{W}_{(1)}^{I,KJ}\boxdot\left(\mathbf{X}_{(1)}^{I,KJ} - (\mathbf{A}\boxdot\mathbf{A})((\mathbf{C}\boxdot\mathbf{C})\odot(\mathbf{B}\boxdot\mathbf{B}))^T\right)\right]^T\right. \\ &\left.\left[\mathbf{W}_{(1)}^{I,KJ}\boxdot\left(\mathbf{X}_{(1)}^{I,KJ} - (\mathbf{A}\boxdot\mathbf{A})((\mathbf{C}\boxdot\mathbf{C})\odot(\mathbf{B}\boxdot\mathbf{B}))^T\right)\right]\right\}\end{aligned}$$

The differential $\mathsf{d}\mathcal{G}$ of $\mathcal{G}$ has to be calculated to determine the three gradient components: the $I \times F$ matrix $\nabla_{\mathbf{A}}\mathcal{G}$, the $J \times F$ matrix $\nabla_{\mathbf{B}}\mathcal{G}$ and the $K \times F$ matrix $\nabla_{\mathbf{C}}\mathcal{G}$. The detail of the calculation is given in Appendix. We finally find that:

$$\begin{aligned}&\mathsf{d}\mathcal{G}(\mathbf{A},\mathbf{B},\mathbf{C}) \\ &= \langle 4\left[\mathbf{A}\boxdot\left((-\boldsymbol{\beta}_{(1)}\boxdot\mathbf{W}_{(1)}^{I,KJ})[(\mathbf{C}\boxdot\mathbf{C})\odot(\mathbf{B}\boxdot\mathbf{B})]\right)\right], \mathsf{d}\mathbf{A}\rangle \\ &+ \langle 4\left[\mathbf{B}\boxdot\left((-\boldsymbol{\beta}_{(2)}\boxdot\mathbf{W}_{(2)}^{J,KI})[(\mathbf{C}\boxdot\mathbf{C})\odot(\mathbf{A}\boxdot\mathbf{A})]\right)\right], \mathsf{d}\mathbf{B}\rangle \\ &+ \langle 4\left[\mathbf{C}\boxdot\left((-\boldsymbol{\beta}_{(3)}\boxdot\mathbf{W}_{(3)}^{K,JI})[(\mathbf{B}\boxdot\mathbf{B})\odot(\mathbf{A}\boxdot\mathbf{A})]\right)\right], \mathsf{d}\mathbf{C}\rangle\end{aligned}$$

Finally, it leads to the three ensuing gradient matrices:

$$\begin{aligned}\nabla_{\mathbf{A}}\mathcal{G}(\mathbf{A},\mathbf{B},\mathbf{C}) &= 4\mathbf{A}\boxdot\left[(-\boldsymbol{\beta}_{(1)}\boxdot\mathbf{W}_{(1)}^{I,KJ})\left((\mathbf{C}\boxdot\mathbf{C})\odot(\mathbf{B}\boxdot\mathbf{B})\right)\right] \\ \nabla_{\mathbf{B}}\mathcal{G}(\mathbf{A},\mathbf{B},\mathbf{C}) &= 4\mathbf{B}\boxdot\left[(-\boldsymbol{\beta}_{(2)}\boxdot\mathbf{W}_{(2)}^{J,KI})\left((\mathbf{C}\boxdot\mathbf{C})\odot(\mathbf{A}\boxdot\mathbf{A})\right)\right] \\ \nabla_{\mathbf{C}}\mathcal{G}(\mathbf{A},\mathbf{B},\mathbf{C}) &= 4\mathbf{C}\boxdot\left[(-\boldsymbol{\beta}_{(3)}\boxdot\mathbf{W}_{(3)}^{K,JI})\left((\mathbf{B}\boxdot\mathbf{B})\odot(\mathbf{A}\boxdot\mathbf{A})\right)\right]\end{aligned} \quad (11)$$

4. OPTIMIZATION ALGORITHMS

4.1. The BFGS Quasi-Newton algorithm

First define a $(I+J+K)F \times 1$ vector $\mathbf{x}$ containing all the loading matrices stacked into one single column. We also define a vector $\mathbf{g}$ of the same size containing the three gradient matrices arranged in the same way:

$$\mathbf{x}^{(k)} = \begin{pmatrix} \mathsf{vec}\{\mathbf{A}^{(k)}\} \\ \mathsf{vec}\{\mathbf{B}^{(k)}\} \\ \mathsf{vec}\{\mathbf{C}^{(k)}\} \end{pmatrix}, \ \mathbf{g}^{(k)} = \begin{pmatrix} \mathsf{vec}\{\nabla_{\mathbf{A}}\mathcal{G}(\mathbf{A}^{(k)}, \mathbf{B}^{(k)}, \mathbf{C}^{(k)})\} \\ \mathsf{vec}\{\nabla_{\mathbf{B}}\mathcal{G}(\mathbf{A}^{(k)}, \mathbf{B}^{(k)}, \mathbf{C}^{(k)})\} \\ \mathsf{vec}\{\nabla_{\mathbf{C}}\mathcal{G}(\mathbf{A}^{(k)}, \mathbf{B}^{(k)}, \mathbf{C}^{(k)})\} \end{pmatrix}$$

where k represents the iteration, $k \in \{1, \ldots, n\}$, and n the total number of iterations, and operator $\mathsf{vec}(.)$ stacks the columns of a matrix into a column vector.
The BFGS algorithm follows the adaptation rule below:

$$\left\{ \begin{array}{ll} \mathbf{x}^{(k+1)} & = \mathbf{x}^{(k)} + \mu^{(k)}\mathbf{d}^{(k)} \\ \mathbf{d}^{(k+1)} & = -(\mathbf{H}^{(k)})^{-1}\mathbf{g}^{(k)}, \end{array} \right. \tag{12}$$

where the $(I+J+K)F \times (I+J+K)F$ matrix $\mathbf{H}$ is the Hessian and $\mu^{(k)}$ is the stepsize. The inverse of the Hessian $\mathbf{H}^{-1}$ can be directly updated using the inversion lemma and defining $\rho = \frac{1}{(\Delta\mathbf{g}^{(k)})^T\Delta\mathbf{x}^{(k)}}$:

$$\left\{ \begin{array}{l} \Delta\mathbf{x}^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} \\ \Delta\mathbf{g}^{(k)} = \mathbf{g}^{(k+1)} - \mathbf{g}^{(k)} \\ (\mathbf{H}^{(k+1)})^{-1} = (\mathbf{H}^{(k)})^{-1} \\ +\rho\left[1 + \rho(\Delta\mathbf{g}^{(k)})^T(\mathbf{H}^{(k)})^{-1}\Delta\mathbf{g}^{(k)}\right]\Delta\mathbf{x}^{(k)}(\Delta\mathbf{x}^{(k)})^T \\ -\rho\Delta\mathbf{x}^{(k)}(\Delta\mathbf{g}^{(k)})^T(\mathbf{H}^{(k)})^{-1} - \rho(\mathbf{H}^{(k)})^{-1}\Delta\mathbf{g}^{(k)}(\Delta\mathbf{x}^{(k)})^T \end{array} \right. \tag{13}$$

If the Hessian is initialized to the identity matrix, then the BFGS update theoretically guarantees positive definiteness. However, it is known that the Hessian rank is equal to the number of free parameters, *i.e.* to $(I+J+K-2)F$. Hence the above update will become more ill conditioned as more iterations are run.

4.2. The conjugate gradient (CG) algorithm

The non linear CG algorithm follows the adaptation rule:

$$\left\{ \begin{array}{ll} \mathbf{x}^{(k+1)} & = \mathbf{x}^{(k)} + \mu^{(k)}\mathbf{d}^{(k)} \\ \mathbf{d}^{(k+1)} & = -\mathbf{g}^{(k+1)} + \beta^{(k)}\mathbf{d}^{(k)} \end{array} \right. \tag{14}$$

Two expressions for β are classically used: the Fletcher-Reeves (β_{FR}) and the Polak-Ribiere (β_{PR}) formula [11]:

$$\beta_{\mathsf{FR}}^{(k+1)} = \frac{{\mathbf{g}^{(k+1)}}^T\mathbf{g}^{(k+1)}}{{\mathbf{g}^{(k)}}^T, \mathbf{g}^{(k)}} \tag{15}$$

$$\beta_{\mathsf{PR}}^{(k+1)} = \frac{{\mathbf{g}^{(k+1)}}^T(\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)})}{{\mathbf{g}^{(k)}}^T\mathbf{g}^{(k)}}. \tag{16}$$

Finally, as noticed in [11], if we reinitialize a conjugate gradient method by setting $\mathbf{d}^{(i)} = -\mathbf{g}^{(i)}$, from time to time (in our case, we have chosen to perform this "restart" every $(I+J+K)F$ iterations which corresponds to the number of unknowns), we might get better performance than by constructing $\mathbf{d}^{(i)}$ by one of the standard formulaes, *i.e.* combining (14) and (15) or (14) and (16) at each iteration.

4.3. The stepsize issue

The stepsize $\mu^{(k)}$ can be found by a global search with ELS, or an approximation by a line search method like backtracking [2, 13]. We have opted for the former: we look for the value of μ that minimizes the polynomial $\mathcal{G}(\mathbf{P_A} \boxdot \mathbf{P_A}, \mathbf{P_B} \boxdot \mathbf{P_B}, \mathbf{P_C} \boxdot \mathbf{P_C})$ where $\mathbf{P_A} = \mathbf{A}^{(k)} + \mu\mathbf{d}_{\mathbf{A}}^{(k)}$, $\mathbf{P_B} = \mathbf{B}^{(k)} + \mu\mathbf{d}_{\mathbf{B}}^{(k)}$ and $\mathbf{P_C} = \mathbf{C}^{(k)} + \mu\mathbf{d}_{\mathbf{C}}^{(k)}$.

5. A VARYING WEIGHT (VW) ALGORITHM

Rather than letting the tensor of weights fixed through the iterations like in [1], we update this tensor and gradually estimate missing values of tensor $\mathbf{X}$ through the iterations.
ALGORITHM. Start with a given tensor $\mathbf{X} \in \mathbb{R}^{+I\times J\times K}$ with missing data, and a tensor of weights $\mathbf{W}$ with binary entries at the beginning (small constant ϵ instead of 0 if the value is missing and 1 if the value exists). Then do as long as the stopping criterion is true:

1. Compute the gradient matrices thanks to Eq. (11). Then, they can be used in any of the descent algorithms (cf. Section 4) to determine the descent direction.
2. Update the loading matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$.
3. The missing values of $\mathbf{X}$ are corrected by interpolation. First, reconstruct tensor $\mathbf{X}$ from loading matrices; this yields tensor $\mathbf{U}$. If x_{ijk} exists, do nothing. Otherwise, compute an averaged value with closest neighbors (six if present, otherwise fewer).
4. If $w_{ijk} \neq 1$, slightly increase its value by a small constant, say $\alpha = 10^{-3}$. By this way, the weight of the initially missing data is progressively increased while the tensor is reconstructed and the loading matrices are estimated. But, since this reconstruction is not perfect, we define an upper bound $M < 1$ (for example $M = 0.2$), such that for each w_{ijk} linked to a missing value, $w_{ijk} \leq M$.
5. Check whether the algorithm has converged. If so, stop, otherwise, continue to the next iteration.

6. COMPUTER SIMULATIONS

6.1. The data mining context

When a water solution containing organic matter is enlightened by an excitation wavelength, it produces two effects: Raman and Rayleigh diffusions and fluorescence. The latter

is characterized by the wavelength of the emission. For low concentrations, the Beer-Lambert law describes the relation between the intensity of fluorescence and these wavelengths:

$$I(\lambda_e, \lambda_f) = I_0\, \epsilon(\lambda_e)\, \gamma(\lambda_f)\, c, \tag{17}$$

where I_0 is a constant, ϵ is the relative excitation spectrum, γ is the relative emission spectrum, and c the concentration of the component. If the solution contains several mixed components, the relation above is slightly modified. The fluorescence intensity becomes a combination of the fluorescence intensity of the different components and reads:

$$I(\lambda_e, \lambda_f, k) = I_0 \sum_l \epsilon_l(\lambda_e)\gamma_l(\lambda_f)c_{kl}, \tag{18}$$

where l is the number of components, k is the number of samples, and c_{kl} is the concentration of the component l in the sample k. The analogy between (1) and (18) is straightforward. Indeed, thanks to the unicity of the CP decomposition, we can establish that parameter l corresponds to the tensor rank F, factor matrix a_{if} matches to $\epsilon_l(\lambda_e)$, factor matrix b_{jf} matches to $\gamma_l(\lambda_f)$, and factor matrix c_{kf} matches to c_{kl}. So, it is possible to recover the two spectra belonging to each component, and their associated concentrations.

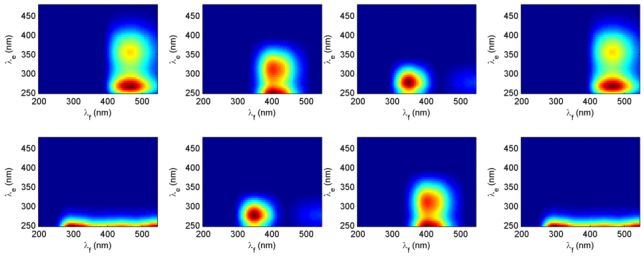

Fig. 1. Exact model ($F = 4$), the 4 estimated fluorescence emission-excitation images using: Left: the CG algorithm with nonnegativity constraint (no missing data). Right: the VW algorithm (30% missing data).

6.2. Numerical results

A tensors T has been simulated, using $F = 4$ components whose 47×71 FEEM - Fluorescence Excitation Emission Matrices ($\mathbf{a}_i\mathbf{b}_i^T$, $\forall i = 1, \ldots, 4$) were similar to the ones displayed in [13] (or on the left of Fig. 1). A 15×4 random nonnegative matrix $\mathbf{C}$ has been used. A certain % of the data have been randomly replaced by Not A Number (NaN). To that aim the coordinates (i, j, k) of those points have been chosen according to an uniform law. On the right of Fig. 1, we have displayed the four estimated fluorescence emission-excitation images using the CG algorithm with nonnegativity constraint and taking into account missing data. The proposed VW algorithm was used, considering $M = 0.1$ and $\alpha = 10^{-5}$ and in this case 30% of the data were missing. On the left of the same figure, we provide the results obtained with the CG with nonnegativity constraint [13]. Results are really good: performances are nearly the same with 30% of missing data.

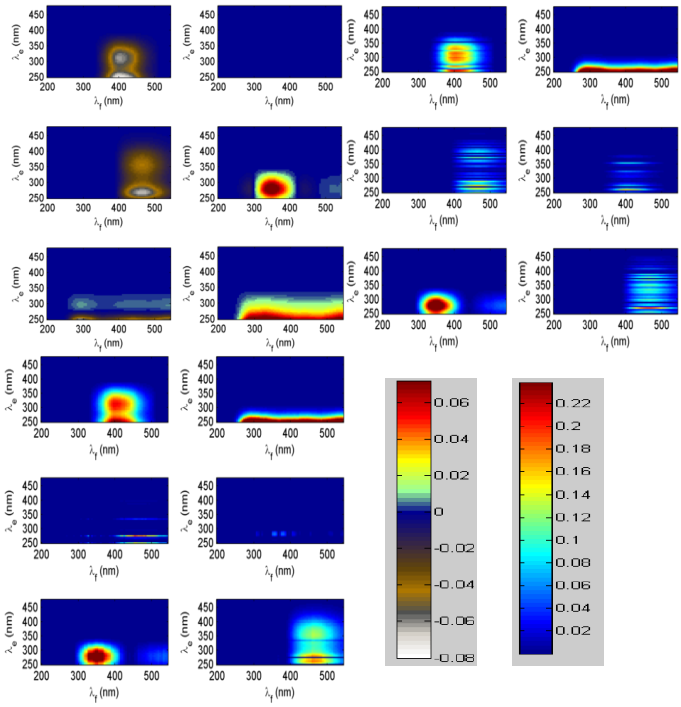

Fig. 2. 80% of missing data. Mixture of 4 factors, assuming $F = 6$. Top left: Estimated fluorescence emission-excitation images using the algorithm suggested in [1] (no nonnegativity contraint). Top right: CG algorithm with nonnegativity constraint and taking into account missing data with fixed weight. Bottom left: VW algorithm. Color maps: CP-WOPT (left); CG and VW (right).

6.3. Robustness versus model errors

The rank is now overestimated. We still consider $F = 4$ components, but, since F is generally unknown, we try to perform the CP decomposition with the assumption that $\widehat{F} = 6$. Here, we chose $M = 0.15$ and the same $\alpha = 10^{-5}$. Ideally, unnecessary components should be equal to 0. As it can be observed in Fig. 2, it is more likely the case using the improved method that we have introduced: only small residues are visible on this set of images. The same method with fixed weights and without reconstruction tends to estimate 6 components instead of 4 and the fourth one is badly estimated. Finally, the

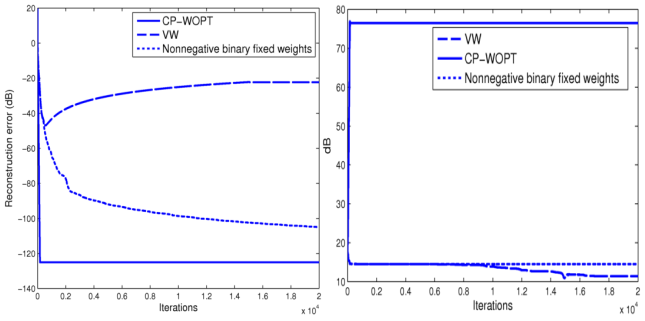

Fig. 3. 80% of missing data, in the overestimated case (F is assumed equal to 6 when $F = 4$). A comparison of the three methods: left: evolution of E_1 versus iterations; right: the same with E_2.

CP-WOPT method introduced in [1] cannot restore the correct FEEMs since it allows negative values. Now, to be able to compare the obtained results, two error indices are introduced. The first one is based on the error between observed and reconstructed tensors: $E_{1\mathsf{dB}} = 10\mathsf{log}_{10}(\|T - \widehat{T})\|_F^2)$, where $\widehat{T} = \sum_{f=1}^{F} \widehat{\mathbf{a}}_f \circ \widehat{\mathbf{b}}_f \circ \widehat{\mathbf{c}}_f$ and $\widehat{\mathbf{a}}$, $\widehat{\mathbf{b}}$ and $\widehat{\mathbf{c}}$ are the estimated factors. The best results should be reached when $E_{1\mathsf{dB}}$ is minimal and close to $-\infty$. The second performance index (which cannot be used on experimental data) is based on the error between the FEEM of actual and estimated components: $E_{2\mathsf{dB}} = 10\mathsf{log}_{10}(\sum_{i=1}^{F} \|\mathbf{a}_i\mathbf{b}_i^T - \widehat{\mathbf{a}}_i\widehat{\mathbf{b}}_i^T)\|_F)$. But first, to compare the images corresponding to the same organic component, actual and estimated FEEM's have to be normalized and then sorted. In Fig. 3, we observe that, despite the fact that E_1 is really low with the CP-WOPT method, the FEEMs of the organic components are badly estimated (really high E_2). On the contrary, with our two suggested methods, E_1 is more important but E_2 tends to be much smaller. Note that the greedy criterion defined in [5] and involving the 3 matrix factors would be too computationally heavy; this is why our criterion involves only $\mathbf{A}$ and $\mathbf{B}$.

7. CONCLUSION

We have described a novel algorithm to handle missing data in the context of nonnegative three-way tensors factorization. Two optimization algorithms have been used. Their ELS versions have been considered. Computer simulations have been provided to enlighten the effectiveness and the robustness of the proposed approach (even in really difficult cases where more than 50% of the data were missing). The method has been compared with other existing approaches.

Appendix A

By defining $\mathbf{M}_1 = (\mathbf{C} \boxdot \mathbf{C}) \odot (\mathbf{B} \boxdot \mathbf{B})$, $\mathbf{M}_2 = (\mathbf{C} \boxdot \mathbf{C}) \odot (\mathbf{A} \boxdot \mathbf{A})$ and $\mathbf{M}_3 = (\mathbf{B} \boxdot \mathbf{B}) \odot (\mathbf{A} \boxdot \mathbf{A})$, we can rewrite Eq. (8) and then calculate its differential:

$$
\begin{aligned}
&\mathsf{d}\mathcal{G}(\mathbf{A},\mathbf{B},\mathbf{C}) = 2\mathsf{trace}\{\boldsymbol{\beta}_{(1)}^T \mathsf{d}\boldsymbol{\beta}_{(1)}\} + 2\mathsf{trace}\{\boldsymbol{\beta}_{(2)}^T \mathsf{d}\boldsymbol{\beta}_{(2)}\} \\
&+ 2\mathsf{trace}\{\boldsymbol{\beta}_{(3)}^T \mathsf{d}\boldsymbol{\beta}_{(3)}\} \\
&= 2\mathsf{trace}\{\boldsymbol{\beta}_{(1)}^T \mathsf{d}[\mathbf{W}_{(1)}^{I,KJ} \boxdot (\mathbf{X}_{(1)}^{I,KJ} - (\mathbf{A} \boxdot \mathbf{A})\mathbf{M}_1^T)]\} \\
&+ 2\mathsf{trace}\{\boldsymbol{\beta}_{(2)}^T \mathsf{d}[\mathbf{W}_{(2)}^{J,KI} \boxdot (\mathbf{X}_{(2)}^{J,KI} - (\mathbf{B} \boxdot \mathbf{B})\mathbf{M}_2^T)]\} \\
&+ 2\mathsf{trace}\{\boldsymbol{\beta}_{(3)}^T \mathsf{d}[\mathbf{W}_{(3)}^{K,JI} \boxdot (\mathbf{X}_{(3)}^{K,JI} - (\mathbf{C} \boxdot \mathbf{C})\mathbf{M}_3^T)]\} \\
&= 4\mathsf{trace}\{-\boldsymbol{\beta}_{(1)}^T \left[\mathbf{W}_{(1)}^{I,KJ} \boxdot \left((\mathbf{A} \boxdot \mathsf{d}\mathbf{A})\mathbf{M}_1^T\right)\right]\} \\
&+ 4\mathsf{trace}\{-\boldsymbol{\beta}_{(2)}^T \left[\mathbf{W}_{(2)}^{J,KI} \boxdot \left((\mathbf{B} \boxdot \mathsf{d}\mathbf{B})\mathbf{M}_2^T\right)\right]\} \\
&+ 4\mathsf{trace}\{-\boldsymbol{\beta}_{(3)}^T \left[\mathbf{W}_{(3)}^{K,JI} \boxdot \left((\mathbf{C} \boxdot \mathsf{d}\mathbf{C})\mathbf{M}_3^T\right)\right]\} \\
&= 4\mathsf{trace}\{-\boldsymbol{\beta}_{(1)}^T \boxdot \mathbf{W}_{(1)}^{I,KJ\,T} (\mathbf{A} \boxdot \mathsf{d}\mathbf{A})\mathbf{M}_1^T\} \\
&+ 4\mathsf{trace}\{-\boldsymbol{\beta}_{(2)}^T \boxdot \mathbf{W}_{(2)}^{J,KI\,T} (\mathbf{B} \boxdot \mathsf{d}\mathbf{B})\mathbf{M}_2^T\} \\
&+ 4\mathsf{trace}\{-\boldsymbol{\beta}_{(3)}^T \boxdot \mathbf{W}_{(3)}^{K,JI\,T} (\mathbf{C} \boxdot \mathsf{d}\mathbf{C})\mathbf{M}_3^T\} \\
&= 4\mathsf{trace}\{\mathbf{M}_1^T(-\boldsymbol{\beta}_{(1)} \boxdot \mathbf{W}_{(1)}^{I,KJ})^T (\mathbf{A} \boxdot \mathsf{d}\mathbf{A})\} \\
&+ 4\mathsf{trace}\{\mathbf{M}_2^T(-\boldsymbol{\beta}_{(2)} \boxdot \mathbf{W}_{(2)}^{J,KI})^T (\mathbf{B} \boxdot \mathsf{d}\mathbf{B})\} \\
&+ 4\mathsf{trace}\{\mathbf{M}_3^T(-\boldsymbol{\beta}_{(3)} \boxdot \mathbf{W}_{(3)}^{K,JI})^T (\mathbf{C} \boxdot \mathsf{d}\mathbf{C})\} \\
&= 4\mathsf{trace}\{\mathbf{M}_1^T(-\boldsymbol{\beta}_{(1)} \boxdot \mathbf{W}_{(1)}^{I,KJ})^T \boxdot \mathbf{A}^T \mathsf{d}\mathbf{A})\} \\
&+ 4\mathsf{trace}\{\mathbf{M}_2^T(-\boldsymbol{\beta}_{(2)} \boxdot \mathbf{W}_{(2)}^{J,KI})^T \boxdot \mathbf{B}^T \mathsf{d}\mathbf{B})\} \\
&+ 4\mathsf{trace}\{\mathbf{M}_3^T(-\boldsymbol{\beta}_{(3)} \boxdot \mathbf{W}_{(3)}^{K,JI})^T \boxdot \mathbf{C}^T \mathsf{d}\mathbf{C})\} \\
&= \langle 4\left[\mathbf{A} \boxdot \left((-\boldsymbol{\beta}_{(1)} \boxdot \mathbf{W}_{(1)}^{I,KJ})\mathbf{M}_1\right)\right], \mathsf{d}\mathbf{A}\rangle \\
&+ \langle 4\left[\mathbf{B} \boxdot \left((-\boldsymbol{\beta}_{(2)} \boxdot \mathbf{W}_{(2)}^{J,KI})\mathbf{M}_2\right)\right], \mathsf{d}\mathbf{B}\rangle \\
&+ \langle 4\left[\mathbf{C} \boxdot \left((-\boldsymbol{\beta}_{(3)} \boxdot \mathbf{W}_{(3)}^{K,JI})\mathbf{M}_3\right)\right], \mathsf{d}\mathbf{C}\rangle
\end{aligned}
$$

8. REFERENCES

[1] E. ACAR, T. G. KOLDA, D. M. DUNLAVY, M. MØRUP, *Scalable Tensor Factorizations for Incomplete Data*, Chem. Intel. Lab. Syst., Vol. 106, pp. 41–56, 2011.

[2] S. BOYD AND L. VANDENBERGHE, *Convex optimization*, Cambridge University Press, Mar. 2004.

[3] R. BRO, S. DE JONG, *A fast non–negativity–constrained least squares algorithm*, J. Chem., Vol. 11, pp. 393–401, 1997.

[4] A. CICHOCKI, R. ZDUNEK, A. H. PHAN, AND S. I. AMARI, *Non negative matrix and tensor factorizations*, Wiley, 2009.

[5] P. COMON, X. LUCIANI, AND A. L. F. DE ALMEIDA, "Tensor decompositions, alternating least squares and other tales," *Jour. Chemometrics*, Vol. 23, pp. 393–405, August 2009.

[6] F. L. HITCHCOCK, *The expression of a tensor or a polyadic as a sum of products*, *J. Math. Phys.*, Vol. 6, pp. 165–189, 1927.

[7] D. LEE AND H. SEUNG, *Learning the parts of objects by non-negative matrix factorization*, *Nature*, pp. 788-791, 1999.

[8] L-H. LIM AND P. COMON, *Nonnegative approximations of nonnegative tensors*, J. Chem., Vol. 23, pp. 432–441, 2009.

[9] P. PAATERO, *Construction and analysis of degenerate PARAFAC models*, J. Chem., 14 (3), pp. 285–299, 2000.

[10] P. PAATERO, *A weigthed non-negative least-squares algorithm for three-way PARAFAC factor analysis*, Chem. Intel. Lab. Syst., Vol. 38, pp. 223–242, 1997.

[11] E. POLAK, *Optimization algorithms and consistent approximations*, *Springer*, 1997.

[12] G. TOMASI, R. BRO, *PARAFAC and missing values*, Chem. Intel. Lab. Syst., Vol. 75, pp. 163-180, 2005.

[13] J.-P. ROYER, N. THIRION-MOREAU, P. COMON, *Computing the polyadic decomposition of nonnegative third order tensors*, Sigal Processing, Vol. 91, pp. 2159-2171, 2011.

[14] J.-P. ROYER, P. COMON, N. THIRION-MOREAU, *Computing the nonnegative 3-way tensor factorization using Tikhonov regularization*, *ICASSP*, pp. 2732–2735, Prague, May 2011.

[15] A. SMILDE, R. BRO, AND P. GELADI, *Multi-Way Analysis with applications in the chemical sciences*, Wiley, 2004.

[16] Q. ZHANG, H. WANG, R. PLEMMONS, AND P. PAUCA, *Tensors methods for hyperspectral data processing: a space object identification study*, J. Opt. Soc. Am. A, Dec. 2008.