Non-negative sub-tensor ensemble factorization (NsTEF) algorithm. A new incremental tensor factorization for large data sets.

Vincent Vigneron, Andreas Kodewitz, Michele Nazareth da Costa, Ana Maria Tome, Elmar Langlang

Non-negative sub-tensor ensemble factorization (NsTEF) algorithm. A new incremental tensor factorization for large data sets.

Vincent Vigneron [a,*], Andreas Kodewitz [a], Michele Nazareth da Costa [b], Ana Maria Tome [c], Elmar Langlang [d]

[a] *IBISC, Univ Evry, Université Paris-Saclay, 91025, Evry, France*
[b] *DSPCom Laboratory, University of Campinas (UNICAMP), PO Box 6101, 13083-852, Campinas/SP, Brazil*
[c] *Departamento de Electronica, Telecomunicacoes e Informática, Universidade de Aveiro, Portugal*
[d] *Institut für Biophysik und physikalische Biochemie, University of Regensburg, Universitätsstrasse 31, D-93040 Regensburg, Germany*

In this work we present a novel algorithm for nonnegative tensor factorization (NTF). Standard NTF algorithms are very restricted in the size of tensors that can be decomposed. Our algorithm overcomes this size restriction by interpreting the tensor as a set of sub-tensors and by proceeding the decomposition of sub-tensor by sub-tensor. This approach requires only one sub-tensor at once to be available in memory.

1. Introduction

Since the pioneering works [42,52], non-negative matrix factorization (NMF) has attracted much interest in the context of several applications such as image and signal processing, computer vision, data analysis, blind source separation [12,22,27,30,32,42,45,53,69]. Particularly in image processing, the associated constraints are desirable to retain the non-negative characteristics of the original data, since the pixel values of the basis images essentially share this feature, leading to a natural meaning regarding the underlying components. As a result, we can, for instance, better represent a face as a linear combination of basis images by NMF in contrast with classical methods such as principal component analysis (PCA) [42,45].

Furthermore, NMF can be viewed as an implicit sparse representation of the input data [27,30,42], which allows representing local features of distributed parts over a human face such as eyes, nose and mouth, and, consequently, learning features of images in face recognition applications. Broadly speaking, a subspace representation by non-negative factorizations makes possible to determinate hidden structures and characteristics inherent to an object class of the input data set, which is helpful in object recognition, detection of semantic features of text documents and of spectral characteristics of hyperspectral images, among others [1,12,22,27,42,46,53,62,69].

Tensorial approaches naturally arise from multilinear structures or multidimensional data, and NMF has been extended to higher-order tensors by the non-negative tensor factorizations (NTFs). The NTF was firstly introduced [6] by imposing non-negative constraints over the matrix factors of the well-known decomposition called CANDECOMP/PARAFAC (or, shortly CP) [5,24]. Analogously, a non-negative version of the Tucker decomposition [61] has also been presented and is referred to here as non-negative Tucker decomposition (NTD) [3], representing a more complex model, as the core tensor could be dense and the matrix factors not necessarily have the same number of columns. An interesting advantage of the NTF/NTD is that, in general, tensor decompositions are essentially unique under mild conditions, as opposed to NMF. More precisely, the uniqueness issue associated with the NTD/NTF happens when the factors are not sufficiently sparse [70].

Almost all NMF algorithms can be generalized or extended to non-negative tensor factorizations by the use of unfolding matri-

* Corresponding author.
E-mail addresses: vincent.vigneron@ibisc.univ-evry.fr (V. Vigneron), andreas.kodewitz@ibisc.univ-evry.fr (A. Kodewitz), nazareth@decom.fee.unicamp.br (M.N. da Costa), ana@ua.pt (A.M. Tome), Elmar.Lang@biologie.uni-regensburg.de (E. Langlang).

ces of the higher-order tensor or by a multi-layer strategy (multi-factor model) [12,17]. A very popular multiplicative update (MU) method [42] can be derived, regarding gradient descent methods, by solving the following optimization problem

$$(\mathbf{A}^*, \mathbf{B}^*) = \arg\max_{A,B} f(\mathbf{A}, \mathbf{B}) = \arg\max_{A,B} \frac{1}{2}\|\mathbf{Y} - \mathbf{AB}\|_F^2. \quad (1)$$

The MU rule with a simple projection to the non-negative space at each updating step is given by

$$b_{n,p} \leftarrow b_{n,p}\left[\frac{\mathbf{A}^{\mathrm{T}}\mathbf{Y}}{\mathbf{A}^{\mathrm{T}}\mathbf{AB}}\right]_{n,p,+}, \qquad a_{m,n} \leftarrow a_{m,n}\left[\frac{\mathbf{YB}^{\mathrm{T}}}{\mathbf{ABB}^{\mathrm{T}}}\right]_{m,n,+}, \quad (2)$$

which has a simple and easy implementation despite presenting slow convergence [49].

A basic approach to NMF, called alternating non-negative least squares (ANLS), is driven by alternating least squares (ALS) techniques [12] based on the alternating minimization of the cost function (1) with respect to the nonnegativity-constrained matrices **A** and **B** separately. However, it does not necessarily lead to global minimization. Several algorithms have been proposed based on the ANLS framework with the purpose of accelerating and overcoming the unstable convergence properties of the standard ANLS and, also, becoming more robust to noise [12] by including penalty terms on the cost function (1) to add supplementary or to preserve constraints on **A** and **B** as nonnegativity, sparsity and smoothness, leading to generalized NMF methods [11,29,30,53,64].

The hierarchical ALS (HALS) algorithm [11] is an alternative method to ALS based on an optimization of a set of local cost functions, which updates each column of **A** and **B** instead of directly computing the whole matrices at each iterative step. This method is simple and often used for multi-layer models to improve performance; furthermore, it is efficient for large-scale NMF [12,17]. Another fast algorithm in the ANLS framework, referred to as ANLS-BPP, was proposed in [35], and employs the block principal pivoting (BPP) and active set methods [41]. The ANLS-BPP technique can outperform the HALS mainly when the matrix factors are sparse.

It is interesting to remark that the optimization problem given by (1) can also be formulated in terms of the Kullback–Leibler divergence [43,45] or other divergences [8,10,67] instead of the Frobenius norm. A problem that arises from the processing of large-scale or ill-conditioned data is the slow convergence, mainly for the MU methods, and the increase of computational complexity and memory requirements. An efficient way to reduce the complexity and to improve the performance of the NTF/NTD is to include a pre-processing step based on low-rank approximation techniques, as proposed in [69,70].

We present in this paper a novel algorithm for NTF and sparse NTF adapted to higher-order tensor decomposition with one large dimension. Algorithms for NTF presented in the past were often restricted in the size of tensors that can be decomposed. Algorithms designed to overcome this size restriction, for example based on *block wise decomposition*, require a frequent access to partitions of the whole data of the tensor. The presented algorithm in contrary requires only a minimum of data access and is even capable to start a decomposition before the whole tensor is known. In comparative tests the algorithm has proved to be competitive with state of the art algorithms. NsTEF is an incremental algorithm as incremental PCA [65] or INMF proposed by Bucak and Gunsel [4] but it deals with tensors, not with matrices. An important advantage of tensor decompositions over standard matrix approach is the model uniqueness; if it exists, is unique [9], which leads to an interesting benefit of our method.

The rest of the paper is organized as follows: related works are presented in Section 2 where problems encountered using standard NTF algorithms are detailed; Section 3 introduces the proposed algorithm, named NsTEF; we present the experimental results in Section 4; finally, we conclude this paper in Section 6.

Notation

N-th order tensors (for $N \geq 3$), matrices (second-order tensors), vectors (first-order tensors), and scalars (zero-order tensors) are respectively denoted by calligraphic ($\mathcal{A}, \mathcal{B}, \ldots$), boldface uppercase ($\mathbf{A}, \mathbf{B}, \ldots$), boldface lower-case ($\mathbf{a}, \mathbf{b}, \ldots$), and lower-case letters ($a, b, \ldots$). Each element of an N-order tensor $\mathcal{A}$ is denoted by $a_{i_1,i_2,\cdots,i_N}$. A tensor $\mathcal{A}$ is called non-negative if all its elements are non-negative, *i.e.* $a_{i_1,i_2,\cdots,i_N} \geq 0$. For non-negative real tensors we use the short hand notation $\mathcal{A} \geq 0$ and $\mathcal{A} \in \mathbb{R}_+$. $\mathbf{A}_{i_1\cdot\cdot} \in \mathbb{R}^{I_2 \times I_3}$, $\mathbf{A}_{\cdot i_2 \cdot} \in \mathbb{R}^{I_1 \times I_3}$, $\mathbf{A}_{\cdot\cdot i_3} \in \mathbb{R}^{I_1 \times I_2}$ represent the *slices* of a third-order tensor $\mathcal{A}$ constructed by fixing the mode 1, 2, and 3, respectively. Any higher-order tensor can be represented by *matrix unfoldings* from the rearrangement of its elements into a matrix from the *matrix slicings* by fixing one mode. Consider for example a third-order tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, we can define three different matrix unfoldings: $\mathbf{A}_{I_1 \times I_2 I_3} \triangleq \left[\mathbf{A}_{\cdot 1\cdot} \ \ldots \ \mathbf{A}_{\cdot I_2 \cdot}\right]$, $\mathbf{A}_{I_2 \times I_3 I_1} \triangleq \left[\mathbf{A}_{\cdot\cdot 1}^{\mathrm{T}} \ \ldots \ \mathbf{A}_{\cdot\cdot I_3}^{\mathrm{T}}\right]$ and $\mathbf{A}_{I_3 \times I_1 I_2} \triangleq \left[\mathbf{A}_{1\cdot\cdot}^{\mathrm{T}} \ \ldots \ \mathbf{A}_{I_1\cdot\cdot}^{\mathrm{T}}\right]$ to represent the same tensor $\mathcal{A}$. By convention, the indexes placed more to the left vary slower and the ones placed more to the right vary faster. The Kronecker, Khatri–Rao, Hadamard and outer products are denoted by $\otimes, \diamond, \bullet$ and $\circ$ respectively. The trace of **A** is denoted by $\mathrm{Tr}(\mathbf{A})$.

Definition 1. The n-mode product of a tensor $\mathcal{G} \in \mathbb{R}^{I_1 \times \cdots \times I_n \times \cdots \times I_N}$ and a matrix $\mathbf{A} \in \mathbb{R}^{J_n \times I_n}$ is an $(I_1 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times \cdots \times I_N)$-tensor given by

$$[\mathcal{G} \times_n \mathbf{A}]_{i_1,\ldots,i_{n-1},j_n,i_{n+1},\ldots,i_N} \triangleq \sum_{i_n=1}^{I_n} g_{i_1,\ldots,i_n,\ldots,i_N} a_{j_n,i_n}, \text{ for all index values.} \quad (3)$$

The n-mode product is a compact form to represent linear transformations involving tensors and (3) can be rewritten in terms of matrix unfoldings by fixing the n-th mode as follows

$$\mathcal{X} = \mathcal{G} \times_n \mathbf{A} \quad \Leftrightarrow \quad \mathbf{X}_{(n)} = \mathbf{AG}_{(n)}, \quad (4)$$

where $\mathbf{X}_{(n)}$ and $\mathbf{G}_{(n)}$ denote the matrix unfolding of $\mathcal{X}$ and $\mathcal{G}$ associated with the n-th mode.

2. Tensor models

Tensor decompositions were first discussed in 1927 by Hitchcock [28]. In the late 1960s tensor decompositions were rediscovered by Tucker [61], Carroll and Chang [5], and Harshman [24] respectively named Tucker decomposition, canonical decomposition (CANDECOMP), and parallel factors analysis (PARAFAC). The two last models, referred to herein as CP, were independently developed in psychometrics and phonetics, however both correspond to the same decomposition and the names report to different features of this model. Tucker model is a general version of the well-known CP model and was also applied in psychometrics. A particular case of this decomposition can be viewed as a multilinear generalization of the singular value decomposition (SVD) for higher-order tensors later introduced by Lathauwer [40]. Tensor decompositions appear today in various fields including image and signal processing, clustering analysis, data compression, blind source separation, direction of arrival estimation, hyperspectral imaging and others [2,51,55,62].

2.1. CANDECOMP/PARAFAC (CP) model

The CP model decomposes a tensor as a minimal sum of rank-one tensors, which can be defined in a concise form and denoted

by a sum of outer products of vectors or by the n-mode product according to

$$\mathcal{Y} \triangleq \left[\left[\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}\right]\right] \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$$
$$= \sum_{r=1}^{R} \mathbf{a}_{\cdot r}^{(1)} \circ \mathbf{a}_{\cdot r}^{(2)} \circ \cdots \circ \mathbf{a}_{\cdot r}^{(N)} = \mathcal{I} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \cdots \times_N \mathbf{A}^{(N)}, \quad (5)$$

where $\mathbf{a}_{\cdot r}^{(n)}$ denotes the r-th column of $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$ for all $n \in \{1, 2, \ldots, N\}$ and the tensor core $\mathcal{I}$ is the identity tensor with ones on the superdiagonal and zeros elsewhere. Despite the usual definition of tensor rank [38] being a generalization of the definition of matrix rank, the best rank-R approximation problem is ill-posed [20] and can only be achieved in some special cases [15,16,19]. From the matrix representation in (4), the N-th order tensor $\mathcal{Y}$ in (5) can be rewritten regarding the matrix unfolding associated with the n-th mode and using the Khatri–Rao product as

$$\mathbf{Y}_{(n)} = \mathbf{A}^{(n)}\left(\mathbf{A}^{(N)} \diamond \cdots \diamond \mathbf{A}^{(n+1)} \diamond \mathbf{A}^{(n-1)} \diamond \cdots \diamond \mathbf{A}^{(1)}\right)^{\mathrm{T}}$$
$$\in \mathbb{R}^{I_n \times I_N \ldots I_{n+1} I_{n-1} \ldots I_1}. \quad (6)$$

2.2. Tucker model

In contrast to CP, Tucker decomposition incorporates interacting dimensions and does not require the same number of columns for the factor matrices $\{\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}\}$. It can be expressed analogously to (5) and (6) as follows

$$\mathcal{Y} \triangleq \left[\left[\mathcal{G}; \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}\right]\right] \in \mathbb{R}^{I_1 \times \cdots \times I_N}$$
$$= \sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} g_{r_1, \ldots, r_N} \mathbf{a}_{\cdot r_1}^{(1)} \circ \cdots \circ \mathbf{a}_{\cdot r_N}^{(N)} = \mathcal{G} \times_1 \mathbf{A}^{(1)} \cdots \times_N \mathbf{A}^{(N)} \quad (7)$$

and

$$\mathbf{Y}_{(n)} = \mathbf{A}^{(n)} \mathbf{G}_{(n)} \left(\mathbf{A}^{(N)} \otimes \cdots \otimes \mathbf{A}^{(n+1)} \otimes \mathbf{A}^{(n-1)} \otimes \cdots \otimes \mathbf{A}^{(1)}\right)^{\mathrm{T}}$$
$$\in \mathbb{R}^{I_n \times I_N \ldots I_{n+1} I_{n-1} \ldots I_1}, \quad (8)$$

where $\mathbf{a}_{\cdot r_n}^{(n)}$ denotes the r_n-th column of $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ for all $n \in \{1, \ldots, N\}$, and $\mathbf{G}_{(n)}$ represents a matrix unfolding of $\mathcal{G}$. Notice that the core tensor $\mathcal{I} \in \mathbb{R}^{R \times \cdots \times R}$ in (5) is replaced by a general core tensor $\mathcal{G} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$. This allows every interaction between the factor matrices.

2.3. Advantages of tensor models

To understand the motivation for a tensor decomposition model let us first recall the linear mixing model widely-used in independent component analysis (ICA) [17], a successful spin-off of PCA. The task of ICA is to learn the basis (mixing) matrix $\mathbf{A} = [\mathbf{a}_1\ \mathbf{a}_2\ \ldots\ \mathbf{a}_J] \in \mathbb{R}^{m \times J}$ and the encoding variable matrix $\mathbf{S} \in \mathbb{R}^{J \times p}$ which minimizes the Frobenious norm $\|\mathbf{Y} - \mathbf{AS}\|_F^2$, given a data matrix $\mathbf{Y} = [\mathbf{y}_1\ \mathbf{y}_2\ \ldots\ \mathbf{y}_p] \in \mathbb{R}^{m \times p}$. The linear data model $\mathbf{X} = \mathbf{AS}$ is learned such that row vectors of the encoding variable matrix are as statistically independent as possible.

This model is adapted to one-dimensional signals, *e.g.* a time series signal. But if we intend to search for underlying sources of a two-dimensional image, a three-dimensional brain scan or a set of images or of measurements for this kind of data model is no longer directly applicable. To make the data fit into the model, a common practice is to vectorize the measured data, which means to concatenate the data into a vector. This vectorization is justified by the assumption that the $\mathbf{x}_i$ are independent which might not be strictly true.

Consider for example a human face image, the calculation of Eigen-faces is a quite common application of ICA, PCA, and other subspace analysis techniques, or scan brain images acquired by positron emission tomography (PET). By evidence the pixels of the images are related to their neighboring pixels, thus the pixels are not independent as required to justify vectorization. Nevertheless, many real data applications show good results using vectorization on face decomposition, brain imaging and other applications [33,34,37,66]. Literature provides to our knowledge no insight why the vectorization approach is successful even with the pixels being dependent. In a tensor model, on the contrary, it is possible to maintain the data in its natural shape [9]. The vectorization is not necessary and in the case of image data, pixel neighborhoods remain intact as in a tensorial representation since we can associate one index with each dimension or measurement. In the case of a series of images, for example, we can reserve two indices for the image dimensions and one index for the different images which naturally leads to a third-order tensor. A time-series of PET for different patients would form a fifth-order tensor.

In addition, there is also an interesting advantage of tensor decompositions over the standard matrix approach: the decomposition, if it exists, is unique [9]. The uniqueness issues for the CP model have been extensively investigated [18,20,23,25,38,39,47,58–60], but the most general sufficient condition and well-known result on uniqueness is attributed to Kruskal [38]. Lim and Comon [47] have recently recovered the uniqueness results given in [38] providing a more practical solution in terms of the coherence measure instead of Kruskal rank, which was employed in array processing context.

Standard NTF algorithms, like presented in [13,35,56], usually require considerable time to decompose a tensor, especially if the tensor to be decomposed presents a large size. The processing time especially rises if the dimensions of the input tensor are *imbalanced, i.e.* one dimension of the tensor is much larger than the other dimensions.

2.4. Non-negative factorization in the tensor domain

non-negative CP decomposition, also known as NTF, is a generalization of NMF to higher-order tensors. A nonnegativity constraint is incorporated to the CP model in (5), *i.e.* the input tensor is decomposed into a sum of outer products of non-negative vectors, which leads to a constrained CP model [70]. In several application problems [26,27,36,62,63], this non-negativity constraint facilitates the interpretation of the decomposition of input data as well as all components are constrained to be non-negative. The decomposition is therefore strictly additive and a graphical display of non-negative data is more convenient.

But even more important, the nonnegativity constraint guarantees the existence of a rank reducing solution to the non-negative CP approximation problem as proven by Lim and Comon in [46] and for this reason, the minimum value of R in (5) is called *non-negative rank*. This proof holds as well for the matrix case (second-order, *i.e.* $N = 2$) as for the higher-order tensor case (i.e. for $N \geq 3$). Additionally, a recent result [54] establishes that a best non-negative rank-R approximation is almost always unique.

Analogously to NTF, NTD adds to the Tucker model, given in (7), a nonnegativity constraint. Implementations of NTF as direct multi-way generalization to the Lee and Seung NMF algorithm were presented by Kim and Choi [36], Lee et al. [44], Mørup et al. [50], Welling and Weber [63]. These implementations use the same MU rule and the Euclidean norm or Kulback-Leibler divergence as cost-function as Lee and Seung. More specifically, an alternating minimization of the set of nonnegativity constrained least squares (ANLS) [12] can be expressed in terms of the Euclidean norm as

$$\mathbf{A}^{(n)} = \arg\min_{\mathbf{A}^{(n)}} \left\|\mathbf{Y}_{(n)} - \mathbf{A}^{(n)}\mathbf{Z}^{(n)}\right\|_2^2 \quad \text{s.t. } \mathbf{A}^{(n)} \geq 0 \quad (9)$$

or, in case of the Kulback–Leibler divergence [43,45], as

$$\mathbf{A}^{(n)} = \arg\min_{\mathbf{A}^{(n)}} \sum_{i_1,\dots,i_N} y_{i_1,\dots,i_N} \log \frac{y_{i_1,\dots,i_N}}{x_{i_1,\dots,i_N}} \quad \text{s.t. } \mathbf{A}^{(n)} \geq 0 \tag{10}$$

where

$$\begin{aligned} \mathbf{X}_{(n)} &= \mathbf{A}^{(n)}\mathbf{Z}^{(n)}, \ \mathbf{Z}^{(n)} \\ &= \mathbf{G}_{(n)}\left(\mathbf{A}^{(N)} \otimes \dots \otimes \mathbf{A}^{(n+1)} \otimes \mathbf{A}^{(n-1)} \otimes \dots \otimes \mathbf{A}^{(1)}\right)^{\mathrm{T}} \\ \iff \mathcal{X} &= \mathcal{G} \times_1 \mathbf{A}^{(1)} \dots \times_N \mathbf{A}^{(N)} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N} \end{aligned} \tag{11}$$

and $\mathbf{Y}_{(n)}$, $\mathbf{X}_{(n)}$, and $\mathbf{G}_{(n)}$ denoting the n-th matrix unfolding of $\mathcal{Y}$, $\mathcal{X}$, and $\mathcal{G}$ respectively, is performed using the MU rule proposed by Lee and Seung [43].

Several other approaches were proposed to perform NTF: gradient-based descent [27,49], fixed point ALS and alternating interior-point gradient [44], ALS [21], HALS [11,13], pre-conditioned nonlinear conjugate gradient [56] and a block principal pivoting method (ANLS-BPP) [35].

3. A fast NTF algorithm with additional non-negativity constraints

3.1. The principle

In the search for faster NTF algorithms, various optimization strategies have been pursued. In this paper we consider the non-negative CP model (see Section 2.4). Then we develop MU algorithms for learning a non-negative CP decomposition of a non-negative input tensor. The MU algorithms is iteratively applied to a matrix representation of the input tensor associated with each mode and then solve the related NMF problem.

In the tensor decomposition for very large-scale problems, memory becomes a major factor. Block wise processing of the data is a common approach to implement parallel processing, but it has the drawback that the blocks of data have to be accessed multiple times. This creates a memory overhead and we want to avoid with our novel algorithm designed for the decomposition of large-scale tensors with imbalanced dimensions.

Consider an N-th order tensor, $\mathcal{Y} \in \mathbb{R}_+^{I_1 \times \dots \times I_n \times \dots \times I_N}$ and an ensemble of $(N-1)$-th order sub-tensors, defined as $\mathcal{Y}_{(i_n)} \in \mathbb{R}_+^{I_1 \times \dots \times I_{n-1} \times I_{n+1} \times \dots \times I_N}$ and constructed by fixing the index associated with the n-th mode. Our proposal is to approach the tensor decomposition for large-scale problems by decomposing each sub-tensor at a time and by proceeding until to obtain a decomposition for the entire set of sub-tensors. We introduce our algorithm, named as non-negative sub-tensor ensemble factorization (NsTEF), and especially derived by considering the input tensor with unbalanced dimensions, which means that one dimension is much larger than the others *i.e.* $I_n \gg I_1, \dots, I_{n-1}, I_{n+1}, \dots, I_N$. The proposed algorithm aims to circumvent the memory problems encountered with standard NTF algorithms.

For simplicity, we consider a third-order input tensor $\mathcal{Y} \in \mathbb{R}_+^{I_1 \times I_2 \times I_3}$ assuming that the third mode is the largest dimension, *i.e.* $I_3 \gg I_1$, I_2, and the identity tensor with unitary elements, *i.e.* $\lambda_r = 1$. In this way, we treat the input tensor from a set of its matrix slicings $\mathbf{Y}_{(i_3)} \triangleq \mathbf{Y}_{\cdot\cdot i_3} \in \mathbb{R}_+^{I_1 \times I_2}$ *i.e.* $\{\mathbf{Y}_{(1)}, \dots, \mathbf{Y}_{(I_3)}\}$. Each matrix $\mathbf{Y}_{(i_3)}$ is decomposed as the non-negative CP model obtaining a set of cost-functions given by

$$D(\mathbf{Y}_{(i_3)}, \hat{\mathbf{Y}}_{(i_3)}) = \left\| \mathbf{Y}_{(i_3)} - \left[[\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \mathbf{a}_{i_3\cdot}^{(3)}] \right] \right\|_F^2 \quad \text{s.t. } \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \mathbf{a}_{i_3\cdot}^{(3)} \geq 0, \tag{12}$$

where $\mathbf{a}_{i_3\cdot}^{(3)}$ denotes the i_3-th row of $\mathbf{A}^{(3)} \in \mathbb{R}_+^{I_3 \times R}$. Fig. 1 depicts this procedure associated with each row i_3.

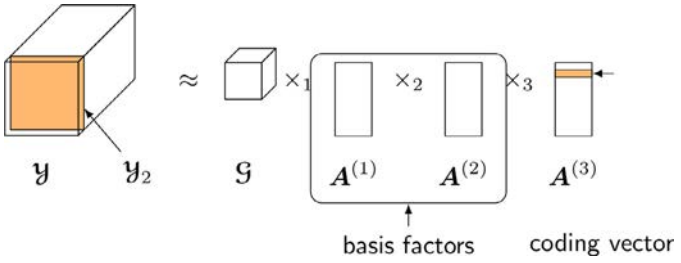


Fig. 1. Scheme of the sub-tensor decomposition for a third-order input tensor as described in (12).

The optimization of each cost function in (12) will lead to the optimization of $\sum_{i_3=1}^{I_3} D(\mathbf{Y}_{(i_3)}, \hat{\mathbf{Y}}_{(i_3)})$ i.e.

$$D(\mathcal{Y}, \hat{\mathcal{Y}}) = \left\| \mathcal{Y} - \left[[\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \mathbf{A}^{(3)}] \right] \right\|_F^2 \quad \text{s.t. } \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \mathbf{A}^{(3)} \geq 0. \tag{13}$$

We perform the optimization of the set of cost-functions $D(\mathbf{Y}_{(i_3)}, \hat{\mathbf{Y}}_{(i_3)})$, for $i_3 = \{1, \dots, I_3\}$, by converting the nonlinear problem (13) into three independent linear least squares problems. So we can update each row of the third factor, $\mathbf{a}_{i_3\cdot}^{(3)}$, and both matrix factors $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$ in an alternating way. These updating steps are repeated until all sub-tensors are treated, *i.e.* for all $i_3 \in \{1, \dots, I_3\}$. Our proposed algorithm is depicted in Algorithm 1. In the following, we will describe the details of each of these steps.

Algorithm 1: NsTEF algorithm.

Input: stream of matrix slicings $\{\mathbf{Y}_{(1)} \dots, \mathbf{Y}_{(I_3)}\}$
Randomly initialize $\mathbf{A}_0^{(1)}$, $\mathbf{A}_0^{(2)}$, and set $\mathbf{\Gamma}_0^{(n)} = \mathbf{0}$, $\mathbf{\Omega}_0^{(n)} = \mathbf{0}$ $(n = \{1, 2\})$, $\lambda \in \mathbb{R}$;
for $t = 1$ *to* I_3 **do**
  Load $\mathbf{Y}_{(t)}$;
  while $\Delta_{error} < 10^{-5}$ **do**
    Update: $\mathbf{a}_{t\cdot(it)}^{(3)}$

$$\mathbf{Z}_{(t)}^{(3)} = \left(\mathbf{A}_{(it-1)}^{(2)} \diamond \mathbf{A}_{(it-1)}^{(1)}\right)^{\mathrm{T}}, \quad \mathbf{y}_{t\cdot}^{(3)} = \mathrm{vec}\left(\mathbf{Y}_{(t)}\right)$$

$$\mathbf{a}_{t\cdot(it)}^{(3)} \leftarrow \mathbf{a}_{t\cdot(it)}^{(3)} \bullet \frac{\mathbf{y}_{t\cdot}^{(3)} \mathbf{Z}_{(t)}^{(3)^{\mathrm{T}}}}{\mathbf{a}_{t\cdot(it)}^{(3)} \mathbf{Z}_{(t)}^{(3)} \mathbf{Z}_{(t)}^{(3)^{\mathrm{T}}} + \lambda \|\mathbf{a}_{t\cdot(it)}^{(3)}\|_1} \tag{14}$$

    Update: $\mathbf{A}_{(it)}^{(1)}$

$$\mathbf{A}_{(it)}^{(1)} \leftarrow \mathbf{A}_{(it)}^{(1)} \bullet \frac{\mathbf{\Omega}_{(t)}^{(1)^{\mathrm{T}}}}{\mathbf{A}_{(it)}^{(1)} \mathbf{\Gamma}_{(t)}^{(1)}} \qquad \begin{cases} \mathbf{Z}_{(t)}^{(1)} = \left(\mathbf{a}_{t\cdot(it)}^{(3)} \diamond \mathbf{A}_{(it-1)}^{(2)}\right)^{\mathrm{T}} \\ \mathbf{\Gamma}_{(t)}^{(1)} = \mathbf{\Gamma}_{(t-1)}^{(1)} + \mathbf{Z}_{(t)}^{(1)} \mathbf{Z}_{(t)}^{(1)^{\mathrm{T}}} \\ \mathbf{\Omega}_{(t)}^{(1)} = \mathbf{\Omega}_{(t-1)}^{(1)} + \mathbf{Z}_{(t)}^{(1)} \mathbf{Y}_{(t)}^{\mathrm{T}} \end{cases} \tag{15}$$

    Update: $\mathbf{A}_{(it)}^{(2)}$

$$\mathbf{A}_{(it)}^{(2)} \leftarrow \mathbf{A}_{(it)}^{(2)} \bullet \frac{\mathbf{\Omega}_{(t)}^{(2)^{\mathrm{T}}}}{\mathbf{A}_{(it)}^{(2)} \mathbf{\Gamma}_{(t)}^{(2)}} \qquad \begin{cases} \mathbf{Z}_{(t)}^{(2)} = \left(\mathbf{a}_{t\cdot(it)}^{(3)} \diamond \mathbf{A}_{(it)}^{(1)}\right)^{\mathrm{T}} \\ \mathbf{\Gamma}_{(t)}^{(2)} = \mathbf{\Gamma}_{(t-1)}^{(2)} + \mathbf{Z}_{(t)}^{(2)} \mathbf{Z}_{(t)}^{(2)^{\mathrm{T}}} \\ \mathbf{\Omega}_{(t)}^{(2)} = \mathbf{\Omega}_{(t-1)}^{(2)} + \mathbf{Z}_{(t)}^{(2)} \mathbf{Y}_{(t)}^{\mathrm{T}} \end{cases} \tag{16}$$

    $it \leftarrow it + 1$

3.2. Update of the matrix factors

In the first step, each row of the third factor $\mathbf{a}^{(3)}_{i_3\cdot}$ is updated by fixing the matrix factors $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$. From the matrix representation (6), the optimization problem takes the form

$$\mathbf{a}^{(3)}_{i_3\cdot} = \arg\min_{\mathbf{a}^{(3)}_{i_3\cdot}} \frac{1}{2}\left\|\mathbf{y}^{(3)}_{i_3\cdot} - \mathbf{a}^{(3)}_{i_3\cdot}\left(\mathbf{A}^{(2)} \diamond \mathbf{A}^{(1)}\right)^{\mathrm{T}}\right\|_F^2 + \lambda\|\mathbf{a}^{(3)}_{i_3\cdot}\|_1 \quad \text{s.t. } \mathbf{a}^{(3)}_{i_3\cdot} \geq 0, \tag{17}$$

where $\mathbf{y}^{(3)}_{i_3\cdot} = \mathrm{vec}\left(\mathbf{Y}_{(i_3)}\right) \in \mathbb{R}_+^{1\times I_2 I_1}$ denotes the i_3-th row of the matrix unfolding of $\mathcal{Y}$ associated with the third mode *i.e.* $\mathbf{Y}_{(3)} \in \mathbb{R}_+^{I_3\times I_2 I_1}$. The expression (17) can be optimized by various methods but, for sake of simplicity, we will implement this step using the MU rule given in [43].

Using the regularization term $\lambda\|\mathbf{a}^{(3)}_{i_3\cdot}\|_1 = \lambda\sum_r |a^{(3)}_{i_3,r}|$ in (17), a sparse coding is enforced by λ, which $\lambda \in \mathbb{R}$ is the regularization parameter controlling the *amount* of sparseness. This means that a solution with many small or zero entries is favored over a dense solution. The sparse coding vector, $\mathbf{a}^{(3)}_{i_3\cdot}$, will influence the update of the basis factor, $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$, insofar as the changes in the factor matrices will be concentrated on the part of the factors that was relevant in the coding of the corresponding example.

So far we have only updated $\mathbf{a}^{(3)}_{i_3\cdot}$, it remains to update the basis factors $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$. In order to perform these updates, knowing only one sub-tensor $\mathbf{Y}_{(i_3)}$ at a time, we will load the information from the already considered sub-tensors. As both matrices $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$ are treated in exactly the same way, we will derive the method to store this information on the example of $\mathbf{A}^{(1)}$. Assuming that we store the information of the already treated sub-tensors, we are able to calculate the cost-function for the decomposition of the set of sub-tensors $\{\mathbf{Y}_{(i_3)}; i_3 = 1,\ldots,t\}$, where t is associated with the t-th slice of $\mathcal{Y}$ being treated in the current moment. Thus, the cost-function of the decomposition can be written as

$$\frac{D(\mathcal{Y},\hat{\mathcal{Y}})}{I_3} \approx \frac{1}{2t}\sum_{i_3=1}^{t} D(\mathbf{Y}_{(i_3)}, \hat{\mathbf{Y}}_{(i_3)}) = \frac{1}{2t}\sum_{i_3=1}^{t}\left\|\mathbf{Y}_{(i_3)} - \left[\left[\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \mathbf{a}^{(3)}_{i_3\cdot}\right]\right]\right\|_F^2. \tag{18}$$

A direct calculation of the sum in this cost function at each iteration for the known $\{\mathbf{a}^{(3)}_{1\cdot},\ldots,\mathbf{a}^{(3)}_{t\cdot}\}$, $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$ would be too expensive to create a fast algorithm. Therefore let us rewrite (18) using (6) with $n = 1$ as

$$\frac{D(\mathcal{Y},\hat{\mathcal{Y}})}{I_3} \approx \frac{1}{2t}\sum_{i_3=1}^{t}\left\|\mathbf{Y}_{(i_3)} - \mathbf{A}^{(1)}\left(\mathbf{a}^{(3)}_{i_3\cdot} \diamond \mathbf{A}^{(2)}\right)^{\mathrm{T}}\right\|_F^2. \tag{19}$$

From (19) and by fixing $\{\mathbf{a}^{(3)}_{1\cdot},\ldots,\mathbf{a}^{(3)}_{t\cdot}\}$ and $\mathbf{A}^{(2)}$, we can obtain an estimate of $\mathbf{A}^{(1)}$

$$\mathbf{A}^{(1)} = \arg\min_{\mathbf{A}^{(1)}} \frac{1}{2t}\sum_{i_3=1}^{t}\Big[\mathrm{Tr}\left(\mathbf{Y}_{(i_3)}\mathbf{Y}^{\mathrm{T}}_{(i_3)}\right) - 2\mathrm{Tr}\left(\mathbf{A}^{(1)}\left(\mathbf{a}^{(3)}_{i_3\cdot} \diamond \mathbf{A}^{(2)}\right)^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}}_{(i_3)}\right) + \mathrm{Tr}\left(\mathbf{A}^{(1)}\left(\mathbf{a}^{(3)}_{i_3\cdot} \diamond \mathbf{A}^{(2)}\right)^{\mathrm{T}}\left(\mathbf{a}^{(3)}_{i_3\cdot} \diamond \mathbf{A}^{(2)}\right)\mathbf{A}^{(1)\mathrm{T}}\right)\Big] \tag{20}$$

for $t \in \{1,\ldots,I_3\}$.

After rewriting the optimization problem in this way, we can now replace the sum over i_3 in (20) by the following recursion relations

$$\mathbf{\Gamma}^{(1)}_{(t)} = \mathbf{\Gamma}^{(1)}_{(t-1)} + \mathbf{Z}^{(1)}_{(t)}\mathbf{Z}^{(1)\mathrm{T}}_{(t)} \in \mathbb{R}_+^{R\times R} \tag{21a}$$

$$\mathbf{\Omega}^{(1)}_{(t)} = \mathbf{\Omega}^{(1)}_{(t-1)} + \mathbf{Z}^{(1)}_{(t)}\mathbf{Y}^{\mathrm{T}}_{(t)} \in \mathbb{R}_+^{R\times I_1} \tag{21b}$$

$$\mathbf{\Theta}^{(1)}_{(t)} = \mathbf{\Theta}^{(1)}_{(t-1)} + \mathbf{Y}_{(t)}\mathbf{Y}^{\mathrm{T}}_{(t)} \in \mathbb{R}_+^{I_1\times I_1} \tag{21c}$$

with

$$\mathbf{Z}^{(1)}_{(t)} = \left(\mathbf{a}^{(3)}_{t\cdot} \diamond \mathbf{A}^{(2)}\right)^{\mathrm{T}} \in \mathbb{R}_+^{R\times I_2}. \tag{22}$$

Thus the optimization problem given by (20) is simplified to the form

$$\mathbf{A}^{(1)} = \arg\min_{\mathbf{A}^{(1)}} \frac{1}{2t}\Big[\mathrm{Tr}\left(\mathbf{\Theta}^{(1)}_{(t)}\right) - 2\mathrm{Tr}\left(\mathbf{A}^{(1)}\mathbf{\Omega}^{(1)}_{(t)}\right) + \mathrm{Tr}\left(\mathbf{A}^{(1)}\mathbf{\Gamma}^{(1)}_{(t)}\mathbf{A}^{(1)\mathrm{T}}\right)\Big]. \quad \text{s.t. } \mathbf{A}^{(1)} \geq 0 \tag{23}$$

Updating them at each step for $t = 1,\ldots,I_3$ we can save the information of the past sub-tensors and avoid the calculation of the sum in (20). However the recursive estimation of these variables (21a)–(21b) leads to an approximation to the cost-function since $\mathbf{Z}^{(1)}_{(t)}$ depends on the knowledge of $\mathbf{A}^{(2)}$ and $\mathbf{a}^{(3)}_{t\cdot}$.

3.3. Implementation

To implement our algorithm, we use a MU for positivity preserving as proposed by Lee and Seung [43] with additional sparseness constraint and factor normalization. We chose this kind of implementation for its simplicity in order to obtain quickly a proof-of-concept of our approach.

Considering a cost function $C(\boldsymbol{\theta})$ of non-negative variables θ_i, the MU has the form

$$\theta_i \leftarrow \theta_i \bullet \left(\frac{\partial C^-(\boldsymbol{\theta})}{\partial\theta_i} \Big/ \frac{\partial C^+(\boldsymbol{\theta})}{\partial\theta_i}\right), \tag{24}$$

with $\frac{\partial C^-(\boldsymbol{\theta})}{\partial\theta_i}$ the positive and $\frac{\partial C(\boldsymbol{\theta})^+}{\partial\theta_i}$ the negative part of the derivative with respect to θ_i. The complete derivative of the cost function $C(\boldsymbol{\theta})$ is $\frac{\partial C(\boldsymbol{\theta})}{\partial\theta_i} = \frac{\partial C(\boldsymbol{\theta})^+}{\partial\theta_i} + \frac{\partial C(\boldsymbol{\theta})^-}{\partial\theta_i}$. This MU rule can be explained as follows: In the case that the gradient is zero, *i.e.* $\frac{\partial C(\theta)^+}{\partial\theta_i} = \frac{\partial C(\theta)^-}{\partial\theta_i}$, θ_i remains unchanged. In the case that the gradient is positive the update rule will decrease the entries of θ_i and vice versa in the case of a negative gradient. For an interpretation in terms of a variational Bayes approach see [57].

From (17) with $i_3 = t$, the resulting update rule for the coding factors $\mathbf{a}^{(3)}_{t\cdot}$ (the first step) is given by

$$\mathbf{a}^{(3)}_{t\cdot} \leftarrow \mathbf{a}^{(3)}_{t\cdot} \bullet \frac{\mathbf{y}^{(3)}_{t\cdot}\mathbf{Z}^{(3)\mathrm{T}}}{\mathbf{a}^{(3)}_{t\cdot}\mathbf{Z}^{(3)}\mathbf{Z}^{(3)\mathrm{T}} + \lambda\|\mathbf{a}^{(3)}_{t\cdot}\|_1}, \tag{25}$$

with

$$\mathbf{Z}^{(3)} = \left(\mathbf{A}^{(2)} \diamond \mathbf{A}^{(1)}\right)^{\mathrm{T}} \in \mathbb{R}_+^{R\times I_2 I_1}. \tag{26}$$

The sparse non-negative updating of the coding vector has to respect a stopping criterion that is adjusted to the factors' needed precision of the sparse coding *vs.* avoidance of over-fitting and time needed for the sparse coding. For simplicity, it was chosen to stop the updates when the change of the cost function is such that $\Delta_{\text{error}} < 10^{-5}$. This value was empirically obtained to perform well for various input tensors.

The derivatives of (23) for $n \in \{1, 2\}$ are

$$\frac{\partial}{\partial\mathbf{A}^{(n)}}\mathrm{Tr}\left(\mathbf{A}^{(n)}\mathbf{\Gamma}^{(n)}_{(t)}\mathbf{A}^{(n)\mathrm{T}}\right) = 2\,\mathbf{A}^{(n)}\mathbf{\Gamma}^{(n)}_{(t)} \tag{27}$$

$$\frac{\partial}{\partial \mathbf{A}^{(n)}} \mathrm{Tr}\left(\mathbf{A}^{(n)} \mathbf{\Omega}_{(t)}^{(n)}\right) = \mathbf{\Omega}_{(t)}^{(n)} \tag{28}$$

and the update rule for the basis factor matrices $\mathbf{A}^{(n)}$ results in

$$\mathbf{A}^{(n)} \leftarrow \mathbf{A}^{(n)} \bullet \frac{\mathbf{\Omega}_{(t)}^{(n)^{\mathrm{T}}}}{\mathbf{A}^{(n)} \mathbf{\Gamma}_{(t)}^{(n)}}, \text{ for } n \in \{1, 2\}. \tag{29}$$

The factor matrices $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$ are randomly initialized with each entry in the range [0, 1]. The iterative variables $\mathbf{\Gamma}_{(t)}^{(n)}$ and $\mathbf{\Omega}_{(t)}^{(n)}$ are initialized with $\mathbf{0}$. We also require all entries of the input tensor to be in the same range [0, 1].

3.4. Factor normalization

Applying a sparseness constraint to only one factor matrix but not all, the normalization of the remaining factors is essential to avoid growing of the non constrained factors. A cost function of the form

$$D(\mathcal{Y}, \hat{\mathcal{Y}}) = \left\| \mathcal{Y} - \left[\!\left[\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \mathbf{A}^{(3)} \right]\!\right] \right\|_F^2 + \lambda \|\mathbf{A}^{(3)}\|_1 \tag{30}$$

could be decreased by simply shrinking the entries of $\mathbf{A}^{(3)}$ and growing the factors $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$ that do not obey the sparseness constraint. To avoid this effect different kinds of normalization can be applied to the remaining factors. Using a MU rule like Lee and Seung [43], a simple min-max normalization, i.e.*re-normalization* which maps the maximum value to 1 and the minimum value to 0, would create values that are exactly zero. These values would not change any more in consequence of the multiplicative nature of the update. Common solutions are normalization of all columns of the factors to a sum of one

$$\mathbf{A}^{(n)} \leftarrow \mathbf{A}^{(n)} \bullet \left(\frac{\mathbf{\Omega}_{(t)}^{(n)^{\mathrm{T}}}}{\mathbf{A}^{(n)} \mathbf{\Gamma}_{(t)}^{(n)}} \right) \bullet \mathbb{1} \left(\frac{1}{\mathbf{A}^{(n)} \mathbf{\Gamma}_{(t)}^{(n)}} \bullet \left(\frac{\mathbf{A}^{(n)} \mathbf{\Gamma}_{(t)}^{(n)}}{\mathbf{\Omega}_{(t)}^{(n)^{\mathrm{T}}}} \right) \right) \tag{31}$$

These are the normalizing update rules to be applied in the coding step.

4. Numerical experiments with two image databases

The usefulness of the NTF algorithm in the extraction of facial features for classification has been demonstrated in [26,50]. In [26] the NTF has been applied for feature extraction for face detection. The feature vector representing an image was the inner-product between the factors. Those measurement vectors over positive (faces) and negative (non-faces) examples were fed into various classifiers, such as Support Vector Machines (SVM) and Adaboost. The MIT CBCL face repository has been used and it has been shown that the features extracted by the NTF factors generated the higher classification accuracy when compared to NMF and PCA.

To examine the convergence behavior of the algorithm and the achieved decomposition accuracy, we performed decompositions of a set of example tensors. In order to allow the comparison of decomposition errors achieved in decompositions of different tensors, we use the *relative* error given by

$$C_{\text{rel.}} = \frac{\left\| \mathcal{Y} - \hat{\mathcal{Y}} \right\|_F}{\|\mathcal{Y}\|_F}, \tag{32}$$

where $\hat{\mathcal{Y}}$ is the reconstruction tensor of $\mathcal{Y}$ from the estimated matrix factors $\mathbf{A}^{(1)}$,$\mathbf{A}^{(2)}$,and $\mathbf{A}^{(3)}$.

For comparison a selection of four algorithms will be employed: The original NMF algorithm by [43], the direct extension of the same algorithm to NTD by [50] *i.e.* higher-order nonnegative matrix factorization (HONMF), a conjugate gradient (cgP) algorithm [56] and an ANLS algorithm with BPP [35] (referred to as ANLS-BPP). All algorithms are used in a pure Matlab®implementation. NMF is a 2-way method, *i.e.* designed for second-order tensors (matrices), therefore the input tensors have to be vectorized. The preconditioned nonlinear cgP is an implementation specialized to third-order tensors, consequently a vectorization becomes necessary if the order of the input tensor exceeds 3. For the evaluation and comparison of these algorithms, we use two sets of face images: The MIT center for biological and computational learning (CBCL) face database and IV^2 face database [14].

The MIT CBCL face database contains 2429 gray scale face images with a resolution of 19×19 pixels. Forming a tensor of dimensions $19 \times 19 \times 2429$ it is a good example of an input tensor of moderate size and heavily imbalanced dimensions. The face images of the CBCL database present a typical decomposition task and have been used in the evaluation of several publications dealing with matrix and tensor factorizations [21,27,31,42]. As the data set contains several images of the same person we have to assure by shuffling their order that they are not grouped together in the data tensor. To also perform tests on a fourth order tensor we selected 211 persons with at least 6 images available and formed a fourth order tensor of dimensions $19 \times 19 \times 6 \times 211$,where the first two ways correspond to the pixels, the third indexes the different images of one person and the fourth way corresponds to the different persons.

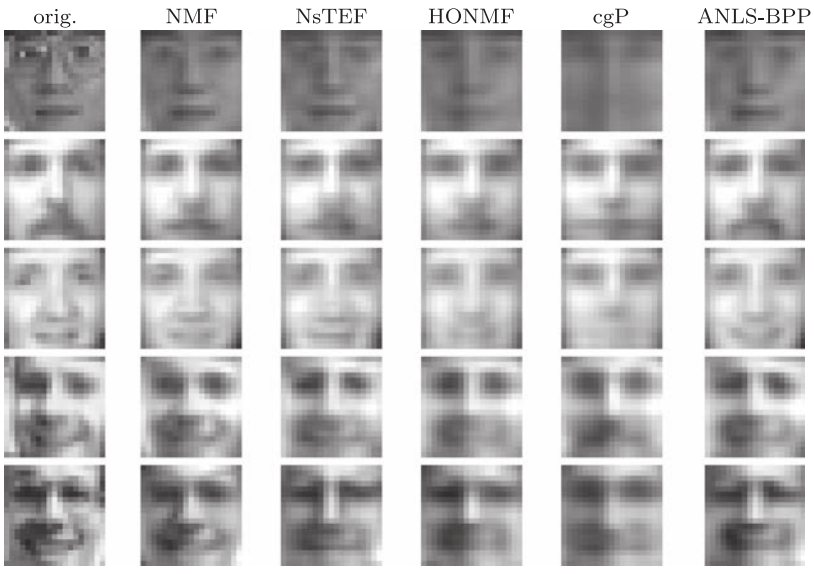
The IV^2 face database contains 4721 face gray scale face images with a resolution of 128×128 pixels. These images form a tensor of dimensions $128 \times 128 \times 4721$ which presents an example of much larger size. On the machine with 4 Gigabytes of RAM which we used to perform our experiments, this tensor uses up almost half of the available memory.

In Fig. 2 we display examples of reconstructed CBCL face images, as well as processing time and relative error, obtained with our algorithm and the 4 algorithms to compare with. The cgP is only used with 3D arrays and HONMF is the tensor extension of NMF. Both visual inspection as well as the reconstruction error and the processing time show that the NMF algorithm is superior to all NTF algorithms. This is due to the fact that NMF is especially designed for 2D-way arrays. From all NTF algorithms, NsTEF performs best. Both HONMF and cgP show poorer reconstruction and a slower processing time. ANLS-BPP is adapted to 3 way arrays but it is very slow (10 times as much as NsTEF) and can not reasonably be used in its actual form in the case of very large data set such as in big data context. The three algorithms converges similarly fast toward nearly the same relative error level.

In Fig. 3 we plotted the relative decomposition error over time. These plots show that NsTEF algorithm exhibits a good convergence behaviour, the latter having being recently analyzed in more details in, *e.g.* [7,48]. These plots also show that the stopping criterion of the ANLS-BPP algorithm has stopped quite late, and that the processing time is much longer than with NsTEF.

In Fig. 4 we show the basis images obtained by the decomposition with the NsTEF algorithm. These images build the basis for the reconstruction. As can be seen, the basis images are sparse, *i.e.* a large number of entries is zero, because of the purely additive nature of non-negative decompositions. The obtained basis images show a similarity to the basis images obtained by other NTF and NMF algorithms, see [21,31,68].

The tensor decomposition formed from IV^2 image database of dimensions $128 \times 128 \times 4721$ is much more demanding. In fact, the HONMF and the cgP algorithms did not converge within 24 hours. However, the NMF, NsTEF and ANLS-BPP algorithms converged within 1 h. The exact processing time as well as the relative error achieved with each of the algorithms and the reconstructed



time	-	31sec	48sec	34min	5h30	2min30
rel. error	-	0.1042	0.1180	0.1791	0.2679	0.1120
$\frac{\text{rel. error}}{\text{image}}$	-	0.11 ± 0.04	0.13 ± 0.05	0.14 ± 0.05	0.28 ± 0.09	0.12 ± 0.05

Fig. 2. Reconstruction after decomposition into 32 basis images; relative error is given as error $\pm$ standard deviation. Computation time raising from left to right.

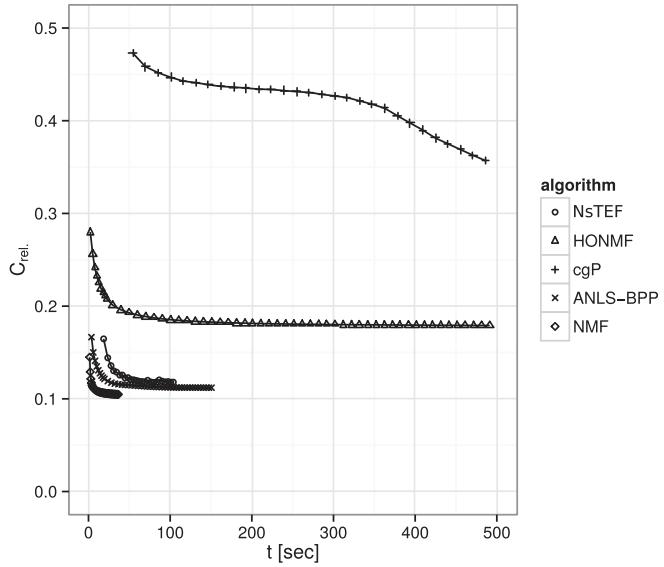


Fig. 3. Evolution of relative error over time for several NTF algorithms when decomposing in third-order CBCL face image tensor. Basis images $R = 32$. Time in seconds.

images are shown in Fig. 5. NMF achieved by far the best decomposition. The reconstruction error is almost one magnitude lower, and the reconstructed image shows much more similarity with the original than with both other algorithms. With almost identical processing time, NsTEF and ANLS-BPP have achieved a decomposition that lacks detail. Recognizing the person in the original image is still possible, and the reconstruction error is not higher than for the CBCL face images.

4.1. Discussion

One shortcoming of our algorithm is that an approximation is necessary to derive the basis update step. The numeric examples showed that our algorithm is comparable with the state-of-the-art algorithms in terms of the error reconstruction and also the processing time. We ascribe this to the coding step which does not need an approximation and thus can equilibrate the committed error. Another way to approach large-scale decompositions are block wise approaches, e.g. Kim and Park [35] and Cichocki et al. [13, see. 1.3.1]. In these approaches parts of the input tensor are selected randomly to be processed. NsTEF, on the contrary, does process parts of the tensor in a regular way. One might suspect that this leads to a better representation of the information processed in later steps in the decomposition but as the past information is stored in the basis update step we did not observe any dependence of the reconstruction error on the location of the data in the input tensor whatsoever. Additionally, the NsTEF avoids the need to access the same part of the tensor multiple times, which considerably reduces memory overhead and contributes to the algorithm speed. Furthermore, block wise algorithms require that parts of the tensor be selected according to an uniform distribution in order to obtain a decomposition with uniform reconstruction error over the whole tensor. In consequence, additional precautions are necessary to be able to start the decomposition of a tensor before all entries of the tensor are known. With our NsTEF algorithm, it is possible to add new examples at any time, and the algorithm will incorporate the new information available. After finishing the presented implementation, we also investigated whether it is possible to transfer our decomposition approach directly to an implementation using the Kullback-Leibler divergence as cost criterion. The use of this divergence, however, did not allow us to derive the iterative variables $\mathbf{\Omega}^t_{(n)}$ and $\mathbf{\Gamma}^t_{(n)}$, which are essential to circumvent the complete re-calculation of the sum in the cost function (18) in every update.

5. Algorithm cost

The approach to decompose an N-dimensional data tensor as a series of $(N-1)$-dimensional data tensors allows to keep less data in memory. Intending to decompose a set of tensors $\{\mathcal{Y}_{(t)} \in \mathbb{R}_+^{I_1 \times \ldots \times I_{N-1}}, t = 1, \ldots, I_N\}$ which is equivalent to the decomposition of a tensor $\mathcal{Y} \in \mathbb{R}_+^{I_1 \times \cdots \times I_N}$ with $I_N \geq I_{N-1} \geq \ldots \geq I_1 \geq R \geq N \geq 2$,

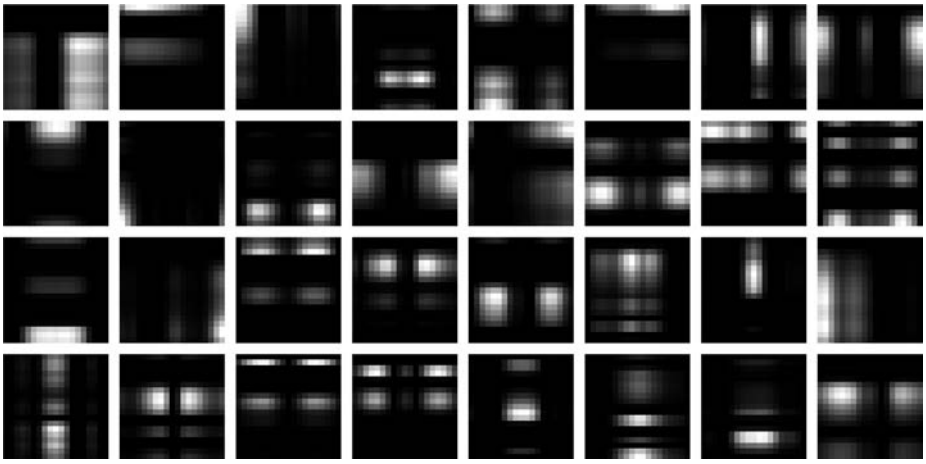


Fig. 4. 32 basis faces obtained by nonnegative sub-tensor ensemble factorization (NsTEF) decomposing 2429 CBCL face images. Images corresponding to nose (3rd row, 7th column), mouth (1st row, 4th column), chin (3rd row, 1st column), cheeks (3rd row, 5th column), eyes, eye-brows or combinations can be found. Others are harder to interpret.

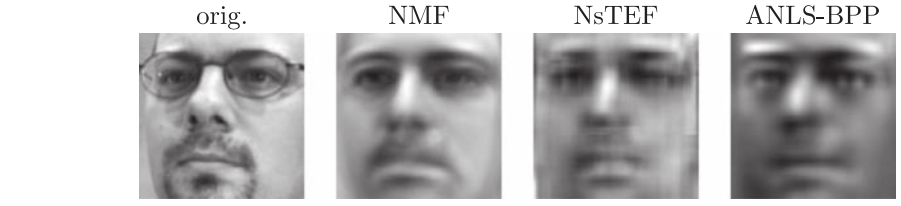


Fig. 5. Reconstruction after decomposition into 32 basis images.

where R is the reduced dimension, the proposed algorithm has a memory usage of

$$M = 2\prod_{i=1}^{N-1} I_i + R\prod_{i=1}^{N-2} I_i + \sum_{i=1}^{N-1} I_i + (N-1)R^2 \tag{33}$$

matrix entries. The first summand is the space required to keep the example $\mathcal{Y}_{(t)}$ and its reconstruction in memory, the second for a $\mathbf{Z}^t_{(n)}, n \neq N$, the third for the $\mathbf{\Omega}^t_{(n)}$ and the last for the $\mathbf{\Gamma}^t_{(n)}$.

In the case of a third-order tensor, *i.e.* $N = 3$, this is: $M = 2R^2 + R(I_1I_2 + 2I_1 + 2I_2 + I_3) + 2I_1I_2$. The memory consumption of the NsTEF algorithm is quadratic in R. Supposing cubic tensors to be decomposed, *i.e.* $I_n = I_m = I, \forall n, m$, we can also determine the memory consumption of the algorithm to be proportional to $I^{(N-1)}$ and exponential in N.

The numerical complexity can help to compare the performance of different algorithms. For the proposed algorithm, the computational cost for the two steps are different. The coding step, *i.e.* for updating each row of $\mathbf{A}^{(3)}$,has a computational cost of

$$\mathcal{O}\left((2R+1)\prod_{i=1}^{N-1} I_i + R\right) \tag{34}$$

and the basis update steps, *i.e.* for updating $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$,

$$\mathcal{O}\left(R\sum_{i=1}^{N-1} I_i + (N-1)(2R+1)\prod_{i=1}^{N-1} I_i\right). \tag{35}$$

For a third-order tensor, this cost is: $\mathcal{O}((2R+1)I_1I_2 + J)$ for the coding step and $\mathcal{O}(2(2R+1)I_1I_2 + R(I_1 + I_2))$ for the basis update step.

The cost of the ordinary matrix product $\mathbf{AB}$,with $\mathbf{A} \in \mathbb{R}^{I\times J}$ and $\mathbf{B} \in \mathbb{R}^{J\times K}$ is assumed to be $\mathcal{O}(IJK)$, *i.e.* in the case of a dense matrix. The computational cost of the element-wise product $\mathbf{A}\bullet\mathbf{B}$ is $\mathcal{O}(IJ)$. The algorithmic cost of both basis update and coding step are linear in R. Supposing the tensor to decompose be cubic, the algorithmic cost of both steps is of the order $I^{(N-1)}$ in I and exponential in N.

With this decomposition approach we have the possibility to start the decomposition of a tensor *before* all of its sub-tensors $\mathcal{Y}_{(t)}$ are known. In cases where the whole tensor, which we want to decompose, is known from the beginning we can present the sub-tensors in arbitrary order. This simple change of the order makes the algorithm more robust to decomposition problems originating from a strong order in the sub-tensors, *e.g.* the sub-tensors belong to several different classes of examples.

Our algorithm decomposes the data tensor along the largest dimension. By re-indexing the tensor we can decompose also along every other direction using our algorithm. In terms of computational cost and memory consumption it might be of no advantage to decompose along any other dimension but the largest one. There might be circumstances that make it necessary to choose this route.

6. Conclusion

We have presented a novel approach for fast decomposition of large non-negative tensors with imbalanced dimensions. We avoided the slowdown due to excessive memory use encountered with standard algorithms by accessing parts of the tensor, *sub-tensors* in a sequential way and storing the information obtained by already accessed sub-tensors. Tests of the proposed algorithm on examples of moderate to large size data repositories demonstrated a highly competitive convergence speed and comparable decomposition quality of the NsTEF algorithm. Because of the special structure, tensor factorization always contains, implicitly or explicitly, a core tensor, which does not exist in matrix factorization.

How to efficiently and effectively deal with it is one key problem in NTF. We can either fix it as an identity, or incorporate it into the optimization procedure. The latter one is computationally very expensive, which makes it unsuitable for large tensor data.

Acknowledgment

This work was supported by Fundação do Estado de São Paulo (FAPESP), Brazil, under Grant 2014/23936-4.

References

[1] M.W. Berry, M. Browne, A.N. Langville, V.P. Pauca, R.J. Plemmons, Algorithms and applications for approximate nonnegative matrix factorization, Comput. Stat. Data Anal. 52 (1) (2007) 155–173.
[2] M. Boussé, O. Debals, L. De Lathauwer, A tensor-based method for large-scale blind system identification using segmentation, in: Proc. 24th European Signal Processing Conference, 2016. accepted
[3] R. Bro, Multi-way analysis in the food industry. Models, algorithms and applications., University of Amsterdam, Netherlands, 1998 Ph.D. thesis.
[4] S.S. Bucak, B. Gunsel, Incremental subspace learning via non-negative matrix factorization, Pattern Recognit. 42 (5) (2009) 788–797.
[5] J.D. Carroll, J.J. Chang, Analysis of individual differences in multidimensional scaling via an N-way generalization of Eckart-Young decomposition, Psychometrika (35) (1970) 283–319.
[6] J.D. Carroll, G. De Soete, S. Pruzansky, Multiway data analysis (1989) 463–472.
[7] J. Chen, C. Richard, J.-C.M. Bermudezz, P. Honeine, Variants of non-negative least means square algorihtm and convergence analysis, IEEE Trans. Signal Process. 62 (15) (2014).
[8] A. Cichocki, S. Amari, R. Zdunek, R. Kompass, G. Hori, Z. He, Extended smart algorithms for nonnegative matrix factorization, in: Artificial Intelligence and Soft Computing - ICAISC 2006, volume 4029 of Lecture Notes in Computer Science, 2006, pp. 548–562.
[9] A. Cichocki, D. Mandic, A.-H. Phan, C. Caiafa, G. Zhou, Q. Zhao, L.D. Lathauwer, Tensor decompositions for signal processing applications from two-way to multiway component analysis, IEEE Signal Process. Mag. 32 (2) (2014) 145–163.
[10] A. Cichocki, R. Zdunek, S.-I. Amari, Csiszar's divergences for nonnegative matrix factorization: family of new algorithms, in: Independent Component Analysis and Blind Signal Separation, Vol. 3889, Springer Berlin / Heidelberg, 2006, pp. 32–39.
[11] A. Cichocki, R. Zdunek, S.-I. Amari, Hierarchical ALS algorithms for nonnegative matrix and 3D tensor factorization, in: Independent Component Analysis and Signal Separation, Vol. 4666, Springer Berlin / Heidelberg, 2007, pp. 169–176. ICA 2007
[12] A. Cichocki, R. Zdunek, A.-H. Phan, S. Amari, Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation, John Wiley & Sons, Ltd, 2009.
[13] A. Cichocki, A.-H. Phan, Fast local algorithms for large scale nonnegative matrix and tensor factorizations, IEICE Trans. Fundam. Electron. Commun. Comput. Sci. E92-A (3) (2009) 1–14.
[14] J. Colineau, J. D'Hose, B. Ben Amor, M. Ardabilian, L. Chen, B. Dorizzi, 3D face recognition evaluation on expressive faces using the IV 2 database, in: J. Blanc–Talon, S. Bourennane, W. Philips, D. Popescu, P. Scheunders (Eds.), Advanced Concepts for Intelligent Vision Systems, Lecture Notes in Computer Science, Vol. 5259, Springer Berlin Heidelberg, 2008, pp. 1050–1061.
[15] P. Comon, B. Mourrain, Decomposition of quantics in sums of powers of linear forms, Signal Process. 53 (1996) 93–108. Special Issue on Higher Order Statistics.
[16] P. Comon, J.M.F. Ten Berge, L. de Lathauwer, J. Castaing, Generic and typical ranks of multi-way arrays, Linear Algebra Appl. 430 (11) (2009) 2997–3007.
[17] P. Comon, C. Jutten, Handbook of Blind Source Separation: Independent Component Analysis and Applications, 1st, Academic Press, 2010.
[18] L. de Lathauwer, A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization, SIAM J. Matrix Analy. Appl. 28 (3) (2006) 642–666.
[19] L. de Lathauwer, B. de Moor, J. Vandewalle, Computation of the canonical decomposition by means of a simultaneous generalized Schur decomposition, SIAM J. Matrix Anal. Appl. 26 (2) (2004) 295–327.
[20] V. de Silva, L. Lim, Tensor rank and the ill-posedness of the best low-rank approximation problem, SIAM J. Matrix Anal. Appl. 30 (3) (2008) 1084–1127.
[21] M. Friedlander, K. Hatz, Computing non-negative tensor factorizations, Optim. Methods Softw. 23 (4) (2008) 631–647.
[22] D. Guillamet, B. Schiele, J. Vitriá, Analyzing nonnegative Matrix factorization for image classification, in: Proceedings of the 16th International Conference on Pattern Recognition, Volume 2, IEEE Computer Society, 2002, pp. 116–119.
[23] X. Guo, S. Miron, D. Brie, A. Stegeman, Uni-mode and partial uniquennes conditions for CANDECOMP/PARAFAC of three-way arrays with linearly dependent loadings., IEEE Trans. Signal Process. 33 (1) (2012) 111–129.
[24] R.A. Harshman, Foundations of the PARAFAC procedure: models and conditions for an "explanatory" multi-modal factor analysis, UCLA Work. Pap. Phonetics 16 (1970) 1–84.
[25] R.A. Harshman, Determination and proof of minimum uniqueness conditions for PARAFAC1, UCLA Work. Pap. Phonetics 22 (1972) 111–117.
[26] T. Hazan, S. Polak, A. Shashua, Non-negative tensor factorization with applications to statistics and computer vision, in: International Conference of Machine Learning, Bonn, Germany, 2005.
[27] T. Hazan, S. Polak, A. Shashua, Sparse image coding using a 3D nonnegative tensor factorization, in: Tenth IEEE International Conference on Computer Vision (ICCV'05), Vol. 1, 2005, pp. 50–57.
[28] F.L. Hitchcock, The expression of a tensor or a polyadic as a sum of products, J. Math. Phys. 6 (1) (1927) 164–189.
[29] P.O. Hoyer, Nonnegative sparse coding, in: Proceedings of IEEE Workshop Neural networks for signal processing, 2002, pp. 557–565.
[30] P.O. Hoyer, Nonnegative matrix factorization with sparseness constraints, J. Mach. Learn. Res. (2004) 1457–1469.
[31] P.O. Hoyer, P. Dayan, Non-negative matrix factorization with sparseness constraints, J. Mach. Learn. Res. 5 (2004) 1457–1469.
[32] L.-Y. Hu, G.-D. Guo, C.-F. Ma, Image processing using Newton-based algorithm of nonnegative matrix factorization, Appl. Math. Comput. 269 (2015) 956–964.
[33] I.A. Illán, Análisis en Componentes de Imágenes Funcionales para la Ayuda al Diagnástico de la Enfermedad de Alzheimer, Departamento de Arquitectura y Tecnología de Computadores, 2009 Ph.D. thesis.
[34] I.A. Illán, J.M. Górriz, J. Ramírez, D. Salas-Gonzalez, M. López, F. Segovia, C.G. Puntonet, M. Gómez-Rio, ^{18}F-FDG PET imaging for computer aided Alzheimer's diagnosis 181 (4) (2011) 903–916.
[35] J. Kim, H. Park, Fast nonnegative matrix factorization: an active-set-like method and comparisons, SIAM J. Sci. Comput. 33 (6) (2011) 3261–3281.
[36] Y.-D. Kim, A. Cichocki, S. Choi, Nonnegative Tucker decomposition with alpha–divergence, in: Proceedings of International Conference on Acoustics, Speech and Signal Processing, 2008, pp. 1829–1832.
[37] A. Kodewitz, S. Lelandais, C. Montagne, V. Vigneron, Learning and using brain maps for Alzheimer's disease detection, Electron. Lett. Comput. Vision Image Anal. 12 (1) (2013) 42–56.
[38] J.B. Kruskal, Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics, Linear Algebra Appl. 18 (1977) 111–117.
[39] J.B. Kruskal, Multiway Data Analysis, vol. 33 of Biometrial Journal, chapter Rank, decomposition, and uniqueness for 3-way and N-way arrays, Elsevier Science Publishers B.V., North-Holland, 1989.
[40] L.D. Lathauwer, Signal processing based on multilinear algebra., Université Catholique de Leuven, Netherlands, 1997 Ph.D. thesis.
[41] C. Lawson, R. Hanson, Solving Least Squares Problems, Society for Industrial and Applied Mathematics, 1995.
[42] D.D. Lee, H.S. Seung, Learning the parts of objects by nonnegative matrix factorization, Nature 401 (1999) 788–791.
[43] D.D. Lee, H.S. Seung, Algorithms for nonnegative matrix factorization, in: T.K. Leen, T.G. Dietterich, V. Tresp (Eds.), Advances in Neural Information Processing Systems 13, MIT Press, 2001, pp. 556–562.
[44] H. Lee, Y.-D. Kim, A. Cichocki, S. Choi, Non-negative tensor factorization for continuous eeg classification, Int. J. Neural Syst. 17 (4) (2007) 305–317.
[45] S.Z. Li, X. Hou, H. Zhang, Q. Cheng, Learning spatially localized, parts-based representation, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Hawaii, USA, Vol. I, 2001, pp. 207–212.
[46] L.-H. Lim, P. Comon, Nonnegative approximations of nonnegative tensors, J. Chemom. 23 (7–8) (2009) 432–441.
[47] L.-H. Lim, P. Comon, Blind multilinear identification, IEEE Trans. Inf. Theory 60 (2) (2014) 1260–1280.
[48] C.-J. Lin, On the convergence of multiplicative update algorithms for non negative matrix factorization, IEEE Trans. on Neural Networks 16 (6) (2007).
[49] C.-J. Lin, Projected gradient methods for nonnegative matrix factorization, Neural Comput. 19 (10) (2007) 2756–2779.
[50] M. Mørup, L.K. Hansen, S.M. Arnfred, Algorithms for sparse non-negative Tucker decompositions, Neural Comput. 20 (8) (2008) 2112–2131.
[51] M. N. da Costa, R. R. Lopes, J.M. T. Romano, Randomized methods for higher-order subspace speration, in: Proc. 24th European Signal Processing Conference, 2016. accepted.
[52] P. Paatero, U. Tapper, Positive matrix factorization: a nonnegative factor model with optimal utilization of error estimates of data values, Environmetrics 5 (2) (1994) 111–126.
[53] V.P. Pauca, J. Piper, R.J. Plemmons, Nonnegative matrix factorization for spectral data analysis, Linear Algebra Appl. 416 (1) (2006) 29–47.
[54] Y. Qi, P. Comon, L.-H. Lim, Uniqueness of nonnegative tensor approximations, IEEE Trans. Inf. Theory 62 (4) (2016) 2170–2183.
[55] F. Raimondi, P. Comon, O. Michel, S. Sahnoun, A. Helmstetter, Tensor decomposition exploiting diversity of propagation velocities; application to localization of icequake events, Signal Process. 118 (2016) 75–88.
[56] J.-P. Royer, N. Thirion-Moreau, P. Comon, Computing the polyadic decomposition of nonnegative third order tensors, Signal Process. 91 (9) (2011) 2159–2171.
[57] R. Schachtner, G. Pöppel, A.M. Tomé, E.W. Lang, A Bayesian approach to the Lee–Seung update rules for NMF, Pattern Recognit. Lett. 45 (2014) 251–256.
[58] A. Stegeman, On uniqueness of the n-th order tensor decomposition into rank-1 terms with linear independence in one mode, SIAM J. Matrix Anal. Appl. 31 (5) (2010) 2498–2516.
[59] A. Stegeman, N.D. Sidiropoulos, On Kruskal's uniqueness condition for the CANDECOMP/PARAFAC decomposition, Linear Algebra Appl. 420 (2007) 540–552.
[60] J.M.F. Ten Berge, N.D. Sidiropoulos, On uniqueness in CANDECOMP/PARAFAC, Psychometrika 67 (3) (2002) 399–409.

[61] L. Tucker, Some mathematical notes of three-mode factor analysis, Psychometrika 31 (3) (1966) 279–311.
[62] M.A. Veganzones, J.E. Cohen, R.C. Farias, J. Chanussot, P. Comon, Nonnegative tensor CP decomposition of hyperspectral data, IEEE Trans. Geosci. Remote Sens. 54 (5) (2016) 2577–2588.
[63] M. Welling, M. Weber, Positive tensor factorization, Pattern Recognit. Lett. 22 (2001) 1255–1261.
[64] T. Yokota, R. Zdunek, A. Cichocki, Y. Yamashita, Smooth non-negative matrix and tensor factorizations for robust multi-way data analysis, Signal Process. 113 (2015) 234–249.
[65] L. Yongmin, On incremental and robust subspace learning, Pattern Recognit. 37 (2004) 1509–1518.
[66] S. Zafeiriou, Tensors in Image Processing and Computer Vision, Advances in Pattern Recognition, Springer, 2009, pp. 105–124.
[67] R. Zdunek, Trust-region algorithm for nonnegative matrix factorization with alpha- and beta-divergences, in: Proc. Joint 34th DAGM and 36th OAGM Symposium on Pattern Recognition (DAGM/OAGM), Vol. 7476, Springer Berlin / Heidelberg, 2012, pp. 226–235.
[68] D. Zhang, S. Chen, Z.-H. Zhou, Two-dimensional non-negative matrix factorization for face representation and recognition, in: in Proc. ICCV'05 Workshop on Analysis and Modeling of Faces and Gestures (AMFG'05), Vol. 3723, Springer, 2005, pp. 350–363.
[69] G. Zhou, A. Cichocki, S. Xie, Fast nonnegative matrix/tensor factorization based on low-Rank approximation, IEEE Trans. Signal Process. 60 (6) (2012) 2928–2940.
[70] G. Zhou, A. Cichocki, Q. Zhao, S. Xie, Nonnegative matrix and tensor factorizations: an algorithmic perspective, IEEE Signal Process. Mag. 31 (3) (2014) 54–65.