Nonnegative Tucker Decomposition

Yong-Deok Kim, Seungjin Choi
Department of Computer Science, POSTECH, Korea
{karma13,seungjin}@postech.ac.kr

Abstract

Nonnegative tensor factorization (NTF) is a recent multiway (multilinear) extension of nonnegative matrix factorization (NMF), where nonnegativity constraints are imposed on the CANDECOMP/PARAFAC model. In this paper we consider the Tucker model with nonnegativity constraints and develop a new tensor factorization method, referred to as nonnegative Tucker decomposition (NTD). The main contributions of this paper include: (1) multiplicative updating algorithms for NTD; (2) an initialization method for speeding up convergence; (3) a sparseness control method in tensor factorization. Through several computer vision examples, we show the useful behavior of the NTD, over existing NTF and NMF methods.

1. Introduction

Subspace analysis or principal component analysis (PCA) is a widely-used linear data model, the task of which is to learn the basis matrix $\boldsymbol{A} = [\boldsymbol{a}_1\ \boldsymbol{a}_2 \cdots \boldsymbol{a}_R] \in \mathbb{R}^{m \times R}$ and the encoding variable matrix $\boldsymbol{S} \in \mathbb{R}^{R \times l}$ which minimizes $\|\boldsymbol{X} - \boldsymbol{A}\boldsymbol{S}\|^2$ ($\|\cdot\|$ denotes the Euclidean norm which is also known as Frobenious norm when the argument is a matrix), given a data matrix $\boldsymbol{X} = [\boldsymbol{x}_1\ \boldsymbol{x}_2 \cdots \boldsymbol{x}_l] \in \mathbb{R}^{m \times l}$. A successful spin-off of PCA is independent component analysis (ICA) [9] where the linear data model $\boldsymbol{X} = \boldsymbol{A}\boldsymbol{S}$ is learned such that row vectors of the encoding variable matrix are as statistically independent as possible, while subspace analysis produces uncorrelated components. In the case where the data matrix contains only nonnegative elements, $\boldsymbol{X} \in \mathbb{R}_+^{m \times l}$, positive matrix factorization (PMF) [16] or nonnegative matrix factorization (NMF) [13] is known as a useful tool in learning parts-based representation as well as in feature extraction.

Computer vision involves a set of image or video data that can be well represented by 3-way or multiway data array which is known as *tensor*. For example, a vector is a 1-way tensor, a matrix is a 2-way tensor, a cube is a 3-way tensor, and so on. The multiway structure reflects rows, columns, RGB (or HSV) color coordinates, time, and so on. In conventional subspace analysis or matrix factorization methods, 2D image data are converted to 1D image vectors, discarding the spatial structure of the original 2D image data. Recently the 2D extension of subspace analysis methods have been proposed, including 2D-PCA [26], and 2D-NMF [27].

A general framework that takes the multiway structure into account, is multilinear algebra involving tensor decomposition. Exemplary tensor decomposition methods include: (1) the Tucker model (also known as multilinear SVD or N-mode SVD) [20, 11, 4]; (2) the CANDECOMP/PARAFAC model [2, 5]; (3) nonnegative tensor factorization (NTF) [25, 19, 7, 3] where nonnegativity constraints are incorporated into the CANDECOMP/PARAFAC model or the PARAFAC2 model. In computer vision applications, N-mode SVD was applied to face image representation [21], showing that considering multiple modes such as different people, expressions, head poses, and lighting conditions improves the face recognition performance [22]. Multilinear ICA was also applied to face image representation and recognition [23]. NTF preserves 2D structure of image data in the factorization, leading to a superior decomposition in the sense of sparse coding, compared to NMF [7].

Existing methods of NTF [25, 19, 7] considered the CANDECOMP/PARAFAC model where a nonnegative tensor is approximated by a linear sum of outer products of nonnegative vectors. In this paper we consider the Tucker model [20] and nonnegativity constraints on the core tensor and mode matrices as well. Then we develop multiplicative updating algorithms for learning a Tucker decomposition of a nonnegative tensor with restricting a core tensor and mode matrices to be nonnegative. The multiplicative updating algorithms iteratively matricize tensor into each mode and then solve NMF problem. The method is referred to as *nonnegative Tucker decomposition* (NTD), in order to distinguish it from previous work, NTF. By fixing the tensor as a structured tensor, NMF, NTF, and nsNMF [17] can be represented as special cases of the NTD. Moreover those multiplicative updating algorithms also can be re-derived from those of the NTD. We present a practically useful ini-

tialization method for NMF and NTD, which dramatically speeds up the convergence. We also stress out the sparseness control in NTD through a structured core tensor, just like nsNMF [17]. Recently, Mørup *et al.* independently developed a HONMF method similar to our NTD [15] with its emphasis on biomedical data analysis. HONMF incorporates the additive L_1norm penalty for sparseness but suffers from slow convergence with random initialization.

2. Background: Multiway analysis

A brief overview of CANDECOMP/PARAFAC and Tucker model is presented, which is necessary for further understanding NTD. Important notations are in Table. 1.

Notation	Description
$\boldsymbol{\mathcal{X}}$	N-way tensor
$\boldsymbol{X}$	matrix
$\boldsymbol{X}_{(n)}$	mode-n matricization of tensor $\boldsymbol{\mathcal{X}}$
$\boldsymbol{A}^{(n)}$	mode-n matrix in Tucker model
$\otimes$	Kronecker product
$\odot$	Khatri-Rao product
$\circ$	outer product
$\circledast$	Hadamard product

Table 1. Notations involving tensor algebra.

An N-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ has N indices $(i_1, i_2, \ldots, i_N)$ and its elements are denoted by $x_{i_1 i_2 \ldots i_N}$ where $1 \leq i_n \leq I_n$. The mode-n matricization of $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ rearranges the elements of $\boldsymbol{\mathcal{X}}$ to form the matrix $\boldsymbol{X}_{(n)} \in \mathbb{R}^{I_n \times I_{n+1} I_{n+2} \cdots I_N I_1 I_2 \cdots I_{n-1}}$, where $I_{n+1} I_{n+2} \cdots I_N I_1 I_2 \cdots I_{n-1}$ is in cyclic order. Following Kiers [10], the fastest-changing index is i_{n+1} in the case of mode-n matricization. Matricizing a tensor is analogous to vectorizing a matrix.

The scalar product of two tensors $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}$ is defined as $\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \rangle = \sum_{i_1, i_2, \ldots, i_N} x_{i_1 i_2 \cdots i_N} y_{i_1 i_2 \cdots i_N}$. The Frobenius norm of a tensor $\boldsymbol{\mathcal{X}}$ is given by $\|\boldsymbol{\mathcal{X}}\| = \sqrt{\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{X}} \rangle}$.

2.1. CANDECOMP/PARAFAC model

An N-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is referred to be of rank-1 if it is represented by the outer product of N vectors:

$$\boldsymbol{\mathcal{X}} = \boldsymbol{a}^{(1)} \circ \boldsymbol{a}^{(2)} \circ \cdots \circ \boldsymbol{a}^{(N)}, \tag{1}$$

where $\boldsymbol{a}^{(n)} \in \mathbb{R}^{I_n}$ for $n = 1, 2, \ldots, N$. In an element-wise form, it is written as

$$x_{i_1 i_2 \ldots i_N} = a^{(1)}_{i_1} a^{(2)}_{i_2} \cdots a^{(N)}_{i_N}, \tag{2}$$

where $a^{(n)}_{i_n}$ denotes the i_nth element of the vector $\boldsymbol{a}^{(n)}$. The rank of tensor $\boldsymbol{\mathcal{X}}$, denoted $R = \text{rank}(\boldsymbol{\mathcal{X}})$, is the minimal number of rank-1 tensors that is required to yield $\boldsymbol{\mathcal{X}}$:

$$\boldsymbol{\mathcal{X}} = \sum_{r=1}^{R} \boldsymbol{A}^{(1)}_{:,r} \circ \boldsymbol{A}^{(2)}_{:,r} \circ \cdots \circ \boldsymbol{A}^{(N)}_{:,r}, \tag{3}$$

where $\boldsymbol{A}^{(n)}_{:,r}$ represents the rth column vector of the mode matrix $\boldsymbol{A}^{(n)} \in \mathbb{R}^{I_n \times R}$.

The CANDECOMP/PARAFAC model seeks a rank-R approximation of the tensor $\boldsymbol{\mathcal{X}}$, i.e.,

$$\boldsymbol{\mathcal{X}} \approx \sum_{r=1}^{R} \boldsymbol{A}^{(1)}_{:,r} \circ \boldsymbol{A}^{(2)}_{:,r} \circ \cdots \circ \boldsymbol{A}^{(N)}_{:,r}, \tag{4}$$

which can be written in an element-wise form as

$$x_{i_1 i_2 \ldots i_N} \approx \sum_{r=1}^{R} a^{(1)}_{i_1 r} a^{(2)}_{i_2 r} \cdots a^{(N)}_{i_N r}. \tag{5}$$

The mode-n matricization of $\boldsymbol{\mathcal{X}}$ in the CANDECOMP/PARAFAC model (4), is expressed by Khatri-Rao products (column-wise Kronecker product) of mode matrices:

$$\begin{aligned} \boldsymbol{X}_{(n)} \quad \approx \quad & \boldsymbol{A}^{(n)} \Big[\boldsymbol{A}^{(n-1)} \odot \cdots \odot \boldsymbol{A}^{(2)} \odot \boldsymbol{A}^{(1)} \\ & \odot \boldsymbol{A}^{(N)} \odot \cdots \odot \boldsymbol{A}^{(n+2)} \odot \boldsymbol{A}^{(n+1)} \Big]^{\top}. \end{aligned} \tag{6}$$

2.2. Tucker model

The mode-n product of a tensor $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_n \times \cdots \times J_N}$ by a matrix $\boldsymbol{A}^{(n)} \in \mathbb{R}^{I_n \times J_n}$ is defined by

$$\begin{aligned} & \left[\boldsymbol{\mathcal{S}} \times_n \boldsymbol{A}^{(n)} \right]_{j_1 \cdots j_{n-1} i_n j_{n+1} \cdots j_N} \\ & \quad = \sum_{j_n=1}^{J_n} s_{j_1 \ldots j_{n-1} j_n j_{n+1} \cdots j_N} a_{i_n j_n}, \end{aligned} \tag{7}$$

leading to a tensor $\boldsymbol{\mathcal{S}} \times_n \boldsymbol{A}^{(n)} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times I_n \times \cdots \times J_N}$ With the mode-n product, a familiar matrix factorization $\boldsymbol{X} = \boldsymbol{U} \boldsymbol{S} \boldsymbol{V}^{\top}$ is written as $\boldsymbol{X} = \boldsymbol{\mathcal{S}} \times_1 \boldsymbol{U} \times_2 \boldsymbol{V}$ in the tensor framework. The mode-n product has following two properties:

$$(\boldsymbol{\mathcal{S}} \times_n \boldsymbol{U}) \times_m \boldsymbol{V} = (\boldsymbol{\mathcal{S}} \times_m \boldsymbol{V}) \times_n \boldsymbol{U} \tag{8}$$

$$(\boldsymbol{\mathcal{S}} \times_n \boldsymbol{U}) \times_n \boldsymbol{V} = \boldsymbol{\mathcal{S}} \times_n (\boldsymbol{V} \boldsymbol{U}). \tag{9}$$

The Tucker model seeks a decomposition of an N-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ as mode products of a core tensor $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$ and N mode matrices $\boldsymbol{A}^{(n)} \in \mathbb{R}^{I_n \times J_n}$,

$$\boldsymbol{\mathcal{X}} \approx \boldsymbol{\mathcal{S}} \times_1 \boldsymbol{A}^{(1)} \times_2 \boldsymbol{A}^{(2)} \cdots \times_N \boldsymbol{A}^{(N)}, \tag{10}$$

which can be written in an element-wise form as

$$x_{i_1 i_2 \cdots i_N} \approx \sum_{j_1, j_2, \ldots, j_N} s_{j_1 j_2 \cdots j_N} a^{(1)}_{i_1 j_1} a^{(2)}_{i_2 j_2} \cdots a^{(N)}_{i_N j_N}. \quad (11)$$

The mode-n matricization of $\boldsymbol{\mathcal{X}}$ in the Tucker model (10), is expressed by Kronecker products of the mode-n matricization of the core tensor and mode matrices:

$$\begin{aligned} \boldsymbol{X}_{(n)} &\approx \boldsymbol{A}^{(n)} \boldsymbol{S}_{(n)} \Big[\boldsymbol{A}^{(n-1)} \otimes \cdots \otimes \boldsymbol{A}^{(2)} \otimes \boldsymbol{A}^{(1)} \\ &\quad \otimes \boldsymbol{A}^{(N)} \otimes \cdots \otimes \boldsymbol{A}^{(n+2)} \otimes \boldsymbol{A}^{(n+1)} \Big]^\top, \quad (12) \end{aligned}$$

where $\boldsymbol{S}_{(n)}$ is the mode-n matricization of the core tensor $\boldsymbol{\mathcal{S}}$. The representation (12) plays a crucial role in deriving multiplicative updating algorithms for NTD.

2.3. Difference between two models

CANDECOMP/PARAFAC and Tucker models were independently developed, providing a ground for multilinear analysis. The main difference between these two models is in the presence of the core tensor in the Tucker model. The core tensor allows column vectors of mode matrices to interact each other in order to reconstruct the original tensor. For example, an individual element of the core tensor, $s_{j_1 j_2 \cdots j_N}$, serves as the weight of the associated combination of $\boldsymbol{A}^{(1)}_{:,j_1}, \boldsymbol{A}^{(2)}_{:,j_2}, \ldots, \boldsymbol{A}^{(N)}_{:,j_N}$. In contrast, the CANDECOMP/PARAFAC model confine such interactions to occur among $\boldsymbol{A}^{(1)}_{:,j}, \boldsymbol{A}^{(2)}_{:,j}, \ldots, \boldsymbol{A}^{(N)}_{:,j}$. At first sight, the CANDECOMP/PARAFAC model can be viewed as a special case of the Tucker model when the core tensor equals to an unit superdiagonal tensor, i.e., $\boldsymbol{\mathcal{S}} = \boldsymbol{\mathcal{I}}$ where $i_{i_1 i_2 \cdots i_N} = \delta_{i_1 i_2 \cdots i_N}$. However in general, the core tensor in the Tucker model cannot be rotated to the unit superdiagonal tensor as in the CANDECOMP/PARAFAC model. Strictly speaking, only the CANDECOMP/PARAFAC model has an uniqueness property up to scaling and permutation of the mode matrices [6, 12]. Even though the Tucker model has not the uniqueness property, it is more useful in data compression [24]. The Tucker model requires a small number of component vectors for respective mode than the CANDECOMP/PARAFAC model, since it uses every combination of mode vectors (column vectors of mode matrices).

3. Nonnegative Tucker decomposition

Given a nonnegative N-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, nonnegative Tucker decomposition (NTD) seeks a factorization of $\boldsymbol{\mathcal{X}}$ that is of the form:

$$\boldsymbol{\mathcal{X}} \approx \widehat{\boldsymbol{\mathcal{X}}} = \boldsymbol{\mathcal{S}} \times_1 \boldsymbol{A}^{(1)} \times_2 \boldsymbol{A}^{(2)} \cdots \times_N \boldsymbol{A}^{(N)}, \quad (13)$$

where the core tensor $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$ and mode matrices $\boldsymbol{A}^{(n)} \in \mathbb{R}^{I_n \times J_n}$ for $n = 1, \ldots, N$ are restricted to have only nonnegative elements in the factorization. As discrepancy measures between the data $\mathcal{X}$ and the model $\widehat{\mathcal{X}}$, we consider LS error function $\mathcal{J}_{LS}$ and I-divergence $\mathcal{J}_I$ which are given by

$$\mathcal{J}_{LS} = \|\boldsymbol{\mathcal{X}} - \widehat{\boldsymbol{\mathcal{X}}}\|_F^2, \quad (14)$$

$$\begin{aligned} \mathcal{J}_I &= \sum_{i_1, i_2, \ldots, i_N} \Big\{ x_{i_1 i_2 \cdots i_N} \log \frac{x_{i_1 i_2 \cdots i_N}}{\widehat{x}_{i_1 i_2 \cdots i_N}} \\ &\quad - x_{i_1 i_2 \cdots i_N} + \widehat{x}_{i_1 i_2 \cdots i_N} \Big\}. \quad (15) \end{aligned}$$

NTD provides a general form of the nonnegative tensor factorization, including NMF, nsNMF, and NTF at special cases. Table. 2 summaries models for NMF, nsNMF, and NTF in the framework of tensor decomposition. As will be shown in Sec. 3.1, the core tensor and mode matrices are learned iteratively, through multiplicative update algorithms. However, instead of learning the core tensor in NTD, we can place a pre-specified specific structure on the core tensor. In this way, nsNMF and NTF emerge from the NTD.

3.1. Multiplicative updating algorithms

We derive multiplicative updating algorithms for mode matrices $\boldsymbol{A}^{(n)}$ as well as the core tensor $\boldsymbol{\mathcal{S}}$ for the NTD. Updating algorithms are quite similar to those in NMF. It is possible to directly derive multiplicative updating algorithms in an element-wise manner for minimizing the error function (14) or (15), as done in NTF [25, 19]. However, updating algorithms in an element-wise form seem to be very complicated, which might bother an efficient and compact implementation. Thus, the approach we take here is to fully use matrix representations of Tucker model given in (12), through properties of Kronecker product and vec$(\cdot)$ operation.

In order to derive an updating rule, we use the following properties:

$$\begin{aligned} \text{vec}\left(\boldsymbol{U}\boldsymbol{B}\boldsymbol{V}^\top\right) &= (\boldsymbol{V} \otimes \boldsymbol{U})\text{vec}(\boldsymbol{B}), \quad (16) \\ [\boldsymbol{U} \otimes \boldsymbol{V}]^\top &= \boldsymbol{U}^\top \otimes \boldsymbol{V}^\top, \\ (\boldsymbol{U} \otimes \boldsymbol{V})(\boldsymbol{B} \otimes \boldsymbol{C}) &= \boldsymbol{U}\boldsymbol{B} \otimes \boldsymbol{V}\boldsymbol{C}. \end{aligned}$$

We define

$$\begin{aligned} \boldsymbol{A}^{(\backslash n)} &= \Big[\boldsymbol{A}^{(n-1)} \otimes \cdots \otimes \boldsymbol{A}^{(1)} \otimes \cdots \\ &\quad \otimes \boldsymbol{A}^{(N)} \otimes \cdots \otimes \boldsymbol{A}^{(n+1)} \Big], \quad (17) \end{aligned}$$

which collects Kronecker products of mode matrices except for $\boldsymbol{A}^{(n)}$ in a backward cyclic manner.

3.1.1 Updating mode matrices

The mode-n matricization of the NTD leads to

$$\boldsymbol{X}_{(n)} = \boldsymbol{A}^{(n)} \boldsymbol{S}_{(n)} \boldsymbol{A}^{(\backslash n)\top} = \boldsymbol{A}^{(n)} \boldsymbol{S}^{(n)}_A, \quad (18)$$

Model	matrix representation	NTD representation
NMF	$\boldsymbol{X} \approx \boldsymbol{A}\boldsymbol{S}$	$\boldsymbol{X} \approx \boldsymbol{I} \times_1 \boldsymbol{A} \times_2 \boldsymbol{S}^\top$
nsNMF	$\boldsymbol{X} \approx \boldsymbol{A}\boldsymbol{M}\boldsymbol{S}$	$\boldsymbol{X} \approx \boldsymbol{M} \times_1 \boldsymbol{A} \times_2 \boldsymbol{S}^\top$
NTF	$\boldsymbol{\mathcal{X}} \approx \sum_{r=1}^{R} \boldsymbol{A}^{(1)}_{:,r} \circ \boldsymbol{A}^{(2)}_{:,r} \circ \cdots \circ \boldsymbol{A}^{(N)}_{:,r}$	$\boldsymbol{\mathcal{X}} \approx \boldsymbol{\mathcal{I}} \times_1 \boldsymbol{A}^{(1)} \times_2 \boldsymbol{A}^{(2)} \cdots \times_N \boldsymbol{A}^{(N)}$
nsNTF	N/A	$\boldsymbol{\mathcal{X}} \approx \boldsymbol{\mathcal{M}} \times_1 \boldsymbol{A}^{(1)} \times_2 \boldsymbol{A}^{(2)} \cdots \times_N \boldsymbol{A}^{(N)}$

Table 2. Models for NMF, nsNMF, and NTF are summarized in the framework of the tensor decomposition.

which is of the same form as the NMF model. Thus, updating algorithms for $\boldsymbol{A}^{(n)}$ follow the NMF updating algorithms for $\boldsymbol{A}$.

- **LS:**

$$\boldsymbol{A}^{(n)} \leftarrow \boldsymbol{A}^{(n)} \circledast \frac{\left[\boldsymbol{X}_{(n)} \boldsymbol{S}_A^{(n)\top}\right]}{\left[\boldsymbol{A}^{(n)} \boldsymbol{S}_A^{(n)} \boldsymbol{S}_A^{(n)\top}\right]}. \tag{19}$$

- **I-divergence:**

$$\boldsymbol{A}^{(n)} \leftarrow \boldsymbol{A}^{(n)} \circledast \frac{\left[\boldsymbol{X}_{(n)} / \left(\boldsymbol{A}^{(n)} \boldsymbol{S}_A^{(n)}\right)\right] \boldsymbol{S}_A^{(n)\top}}{\boldsymbol{1}\, \boldsymbol{z}^\top}, \tag{20}$$

where $/$ is the element-wise division, $\boldsymbol{1} \in \mathbb{R}^{I_n \times 1}$, $\boldsymbol{z} \in \mathbb{R}^{J_n \times 1}$ with $z_i = \sum_j \boldsymbol{S}_{A\ ij}^{(n)}$. In updating $\boldsymbol{A}^{(n)}$, the rest of parameters denoted by $\boldsymbol{S}_A^{(n)}$ are fixed.

The encoding variable matrix $\boldsymbol{S}_A^{(n)}$ should be computed through Kronecker products, which requires a heavy computation. We compute $\boldsymbol{S}_A^{(n)}$ in the following way:

$$\begin{aligned}
\boldsymbol{S}_A^{(n)} &= \boldsymbol{I}\boldsymbol{S}_{(n)}\boldsymbol{A}^{(\backslash n)\top} \\
&= \Big[\boldsymbol{\mathcal{S}} \times_1 \boldsymbol{A}^{(1)} \cdots \times_{n-1} \boldsymbol{A}^{(n-1)} \times_n \boldsymbol{I} \\
&\quad \times_{n+1} \boldsymbol{A}^{(n+1)} \cdots \times_N \boldsymbol{A}^{(N)}\Big]_{(n)} \\
&= \Big[\boldsymbol{\mathcal{S}} \times_1 \boldsymbol{A}^{(1)} \times_2 \cdots \times_{n-1} \boldsymbol{A}^{(n-1)} \\
&\quad \times_{n+1} \boldsymbol{A}^{(n+1)} \cdots \times_N \boldsymbol{A}^{(N)}\Big]_{(n)} \\
&= \Big[\boldsymbol{\mathcal{S}} \times_{m \neq n} \boldsymbol{A}^{(m)}\Big]_{(n)}.
\end{aligned} \tag{21}$$

In the LS case, computation costs are more reduced by:

$$\begin{aligned}
\boldsymbol{A}^{(n)} \boldsymbol{S}_A^{(n)} \boldsymbol{S}_A^{(n)\top} &= \Big[\boldsymbol{\mathcal{S}} \times_{m \neq n} \boldsymbol{A}^{(m)\top} \boldsymbol{A}^{(m)}\Big]_{(n)} \boldsymbol{S}_{(n)}^\top \\
\boldsymbol{X}_{(n)} \boldsymbol{S}_A^{(n)\top} &= \Big[\boldsymbol{\mathcal{X}} \times_{m \neq n} \boldsymbol{A}^{(m)\top}\Big]_{(n)}
\end{aligned} \tag{22}$$

Updating algorithms for the rest of mode matrices can be easily derived in the same way, by matricizing the Tucker model into associated modes.

3.1.2 Updating core tensor

From (18) and (16)

$$\begin{aligned}
\mathrm{vec}\left(\boldsymbol{X}_{(n)}\right) &= \mathrm{vec}\left(\boldsymbol{A}^{(n)} \boldsymbol{S}_{(n)} \boldsymbol{A}^{(\backslash n)\top}\right) \\
&= \left(\boldsymbol{A}^{(\backslash n)} \otimes \boldsymbol{A}^{(n)}\right) \mathrm{vec}\left(\boldsymbol{S}_{(n)}\right),
\end{aligned} \tag{23}$$

which is of the same form as NMF. Therefore, we use updating algorithms for the encoding variable matrix $\boldsymbol{S}$ in NMF and show that associated parameters are efficiently computed in the framework of the tensor algebra.

LS: We use the updating algorithm for $\boldsymbol{S}$ in NMF, leading to

$$\mathrm{vec}\left(\boldsymbol{S}_{(n)}\right) \leftarrow \mathrm{vec}\left(\boldsymbol{S}_{(n)}\right) \circledast \boldsymbol{K}_n, \tag{24}$$

where

$$\boldsymbol{K}_n = \frac{\left[\boldsymbol{A}^{(\backslash n)} \otimes \boldsymbol{A}^{(n)}\right]^\top \mathrm{vec}\left(\boldsymbol{X}_{(n)}\right)}{\left[\boldsymbol{A}^{(\backslash n)} \otimes \boldsymbol{A}^{(n)}\right]^\top \left[\boldsymbol{A}^{(\backslash n)} \otimes \boldsymbol{A}^{(n)}\right] \mathrm{vec}\left(\boldsymbol{S}_{(n)}\right)}. \tag{25}$$

Invoking (16) again, leads to

$$\begin{aligned}
&\left[\boldsymbol{A}^{(\backslash n)} \otimes \boldsymbol{A}^{(n)}\right]^\top \mathrm{vec}\left(\boldsymbol{X}_{(n)}\right) \\
&\quad = \mathrm{vec}\left(\boldsymbol{A}^{(n)\top} \boldsymbol{X}_{(n)} \boldsymbol{A}^{(\backslash n)}\right) \\
&\quad = \mathrm{vec}\left(\left[\boldsymbol{\mathcal{X}} \times_1 \boldsymbol{A}^{(1)\top} \cdots \times_N \boldsymbol{A}^{(N)\top}\right]_{(n)}\right)
\end{aligned}$$

In a similar way, we calculate

$$\begin{aligned}
&\left[\boldsymbol{A}^{(\backslash n)} \otimes \boldsymbol{A}^{(n)}\right]^\top \left[\boldsymbol{A}^{(\backslash n)} \otimes \boldsymbol{A}^{(n)}\right] \mathrm{vec}\left(\boldsymbol{S}_{(n)}\right) \\
&\quad = \left[\boldsymbol{A}^{(\backslash n)\top} \boldsymbol{A}^{(\backslash n)} \otimes \boldsymbol{A}^{(n)\top} \boldsymbol{A}^{(n)}\right] \mathrm{vec}\left(\boldsymbol{S}_{(n)}\right) \\
&\quad = \mathrm{vec}\left(\boldsymbol{A}^{(n)\top} \boldsymbol{A}^{(n)} \boldsymbol{S}_{(n)} \boldsymbol{A}^{(\backslash n)\top} \boldsymbol{A}^{(\backslash n)}\right) \\
&\quad = \mathrm{vec}\left(\left[\boldsymbol{\mathcal{S}} \times_1 \boldsymbol{A}^{(1)\top} \boldsymbol{A}^{(1)} \cdots \times_N \boldsymbol{A}^{(N)\top} \boldsymbol{A}^{(N)}\right]_{(n)}\right).
\end{aligned}$$

With these calculations, the updating algorithm for the core tensor $\boldsymbol{\mathcal{S}}$ is written as

$$\boldsymbol{\mathcal{S}} \leftarrow \boldsymbol{\mathcal{S}} \circledast \frac{\boldsymbol{\mathcal{X}} \times_1 \boldsymbol{A}^{(1)\top} \cdots \times_N \boldsymbol{A}^{(N)\top}}{\boldsymbol{\mathcal{S}} \times_1 \boldsymbol{A}^{(1)\top} \boldsymbol{A}^{(1)} \cdots \times_N \boldsymbol{A}^{(N)\top} \boldsymbol{A}^{(N)}}. \tag{26}$$

I-divergence: We only present a final result to save a space:

$$\boldsymbol{\mathcal{S}} \leftarrow \boldsymbol{\mathcal{S}} \circledast \frac{(\boldsymbol{\mathcal{X}}/\widehat{\boldsymbol{\mathcal{X}}}) \times_1 \boldsymbol{A}^{(1)\top} \cdots \times_N \boldsymbol{A}^{(N)\top}}{\boldsymbol{\mathcal{E}} \times_1 \boldsymbol{A}^{(1)\top} \cdots \times_N \boldsymbol{A}^{(N)\top}}, \tag{27}$$

where $\boldsymbol{\mathcal{E}}$ is a tensor whose every elements are one.

Table 3 summaries multiplicative updating algorithms for NMF, nsNMF, NTF, and NTD, in the case of the LS error measure. Algorithms for NMF, nsNMF, and NTF can be easily re-derived by plugging an adequate core tensor into updating algorithms for mode matrices in NTD. Our multiplicative updating algorithms for NTD are directly derived from NMF multiplicative updating algorithms, the monotonic convergence analysis in [14] can be applied to our case as well.

3.2. Initialization methods

A simple NMF fitting algorithm is iteratively solving least square problem and then projecting its solution in to first orthant (convert negative term into zero). Absolutely this naive approach never gives local optimal solutions. However, we modify it to find good initial starting points for NMF and NTD.

Consider a subproblem of the NMF, finding coefficients of a conic combination, $\boldsymbol{x} \approx \boldsymbol{A}\boldsymbol{s}$, where $\boldsymbol{A} \in \mathbb{R}_+^{m \times R}, \boldsymbol{s} \in \mathbb{R}_+^R$ The least square solution $\widehat{\boldsymbol{s}} = \boldsymbol{A}^+\boldsymbol{s}$ consists of nonnegative terms $\widehat{\boldsymbol{s}}_+$ and negative terms $\widehat{\boldsymbol{s}}_-$. If an energy of $\widehat{\boldsymbol{s}}_-$ is small, then there is a high probability of $\widetilde{\boldsymbol{s}} = \widehat{\boldsymbol{A}}^+\boldsymbol{x}$, where $\widehat{\boldsymbol{A}}$ consist of columns of $\boldsymbol{A}$ which associated with $\widehat{\boldsymbol{s}}_+$, will be nonnegative and a relatively good solution. The updating $\boldsymbol{S}$ in the NMF requires solving l, number of data samples, subproblems but $\widehat{\boldsymbol{A}}$ will be different for each subproblem. Our simple trick is that applying multiplicative updating algorithm again into projected least square solutions. Since 0 elements always remain 0 in multiplicative updating algorithm, $\widehat{\boldsymbol{A}}$ is automatically selected from $\boldsymbol{A}$ and then $\widehat{\boldsymbol{s}}_+$ is updated for each subproblem.

The numerical experiments show that the assumption, the energy of negative terms should be small, holds for both $\boldsymbol{A}$ and $\boldsymbol{S}$ in the NMF but only for the core tensor $\boldsymbol{\mathcal{S}}$ in the NTD. We present initialization methods for NMF and NTD based on these observations. Algorithms are described in Table. 4 and 5.

3.3. Sparseness control

The nsNMF model [17] is described by $\boldsymbol{X} \approx \boldsymbol{A}\boldsymbol{M}\boldsymbol{S}$, introducing a smoothing matrix $\boldsymbol{M}$ given by

$$\boldsymbol{M} = (1-\theta)\boldsymbol{I} + \frac{\theta}{R}\boldsymbol{1}\boldsymbol{1}^\top, \tag{28}$$

where the parameter θ satisfies $0 \le \theta \le 1$. The parameter θ controls the extent of smoothness of the matrix operator

Table 4. NMF initialization algorithm

Input: $\boldsymbol{X} \in \mathbb{R}_+^{m \times l}$, R.
Output: $\boldsymbol{A} \in \mathbb{R}_+^{m \times R}$, $\boldsymbol{S} \in \mathbb{R}_+^{R \times l}$.
$\cdot\ \boldsymbol{A} \leftarrow \mathrm{rand}(m, R)$
$\cdot\ \boldsymbol{S} \leftarrow \mathrm{rand}(R, m)$
Repeat 30 $\sim$ 50 times
$\cdot\ \boldsymbol{A} \leftarrow \max(\boldsymbol{X}\boldsymbol{S}^+, \epsilon)$
$\cdot\ \boldsymbol{A} \leftarrow \boldsymbol{A} \circledast \frac{\boldsymbol{X}\boldsymbol{S}^\top}{\boldsymbol{A}\boldsymbol{S}\boldsymbol{S}^\top}$
$\cdot\ \boldsymbol{S} \leftarrow \max(\boldsymbol{A}^+\boldsymbol{X}, \epsilon)$
$\cdot\ \boldsymbol{S} \leftarrow \boldsymbol{S} \circledast \frac{\boldsymbol{A}^\top\boldsymbol{X}}{\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{S}}$

Table 5. NTD initialization algorithm

Input: $\boldsymbol{\mathcal{X}} \in \mathbb{R}_+^{I_1 \times \cdots \times I_N}$, $J_1, \ldots, J_N$.
Output: $\boldsymbol{\mathcal{S}} \in \mathbb{R}_+^{J_1 \times \cdots \times J_N}$, $\boldsymbol{A}^{(n)} \in \mathbb{R}_+^{I_n \times J_n}, \forall n$
$\cdot\ \boldsymbol{A}^{(1)} \leftarrow$ NMF intialization$(\boldsymbol{X}_{(1)}, J_1)$
$\vdots$
$\cdot\ \boldsymbol{A}^{(N)} \leftarrow$ NMF intialization$(\boldsymbol{X}_{(N)}, J_N)$
$\cdot\ \boldsymbol{\mathcal{S}} \leftarrow \mathrm{rand}(J_1, \ldots, J_N)$
Repeat 30 $\sim$ 50 times
$\cdot\ \boldsymbol{A}^{(1)} \leftarrow \boldsymbol{A}^{(1)} \circledast \frac{\left[\boldsymbol{X}_{(1)}\boldsymbol{S}_A^{(1)\top}\right]}{\left[\boldsymbol{A}^{(1)}\boldsymbol{S}_A^{(1)}\boldsymbol{S}_A^{(1)\top}\right]}$
$\vdots$
$\cdot\ \boldsymbol{A}^{(N)} \leftarrow \boldsymbol{A}^{(N)} \circledast \frac{\left[\boldsymbol{X}_{(N)}\boldsymbol{S}_A^{(N)\top}\right]}{\left[\boldsymbol{A}^{(N)}\boldsymbol{S}_A^{(N)}\boldsymbol{S}_A^{(N)\top}\right]}$
$\cdot\ \boldsymbol{\mathcal{S}} \leftarrow \max(\boldsymbol{\mathcal{X}} \times_1 \boldsymbol{A}^{(1)+} \cdots \times_N \boldsymbol{A}^{(N)+}, \epsilon)$
$\cdot\ \boldsymbol{\mathcal{S}} \leftarrow \boldsymbol{\mathcal{S}} \circledast \frac{\boldsymbol{\mathcal{X}} \times_1 \boldsymbol{A}^{(1)\top} \cdots \times_N \boldsymbol{A}^{(N)\top}}{\boldsymbol{\mathcal{S}} \times_1 \boldsymbol{A}^{(1)\top}\boldsymbol{A}^{(1)} \cdots \times_N \boldsymbol{A}^{(N)\top}\boldsymbol{A}^{(N)}}$

$\boldsymbol{M}$. For $\theta = 0$, the model (28) is equivalent to the original NMF. As $\theta \to 1$, stronger smoothing is imposed on $\boldsymbol{M}$, leading to a strong sparseness on both $\boldsymbol{A}$ and $\boldsymbol{S}$ in order to maintain the faithfullness of the model to the data.

The same idea can be applied to the NTD. We propose a nsNTD model:

$$\begin{aligned} \boldsymbol{\mathcal{X}} &\approx \left(\boldsymbol{\mathcal{S}} \times_1 \boldsymbol{M}^{(1)} \cdots \boldsymbol{M}^{(N)}\right) \times_1 \boldsymbol{A}^{(1)} \cdots \times_N \boldsymbol{A}^{(N)} \\ &= \boldsymbol{\mathcal{S}} \times_1 \boldsymbol{A}^{(1)}\boldsymbol{M}^{(1)} \cdots \times_M \boldsymbol{A}^{(M)}\boldsymbol{M}^{(N)}, \end{aligned} \tag{29}$$

where $\boldsymbol{M}^{(1)}, \ldots, \boldsymbol{M}^{(N)}$ are smoothing matrices for each mode. They smooth the core tensor and mode matrices simultaneously as $\boldsymbol{M}$ does $\boldsymbol{A}$ and $\boldsymbol{S}$ in the nsNMF. For the same reason, a sparseness on both $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}$ is obtained. Moreover we don't have to derive new updating algorithms. New updating algorithms are given by just replacing $\boldsymbol{S}$ into $\boldsymbol{\mathcal{S}} \times_1 \boldsymbol{M}^{(1)} \cdots \boldsymbol{M}^{(N)}$ and $\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}$ into $\boldsymbol{A}^{(1)}\boldsymbol{M}^{(1)}, \ldots, \boldsymbol{A}^{(N)}\boldsymbol{M}^{(N)}$ in original mode matrices and core tensor updating algorithms, respectively.

Model	Update rule
NMF	$\boldsymbol{A} \leftarrow \boldsymbol{A} \circledast \frac{\boldsymbol{X}\boldsymbol{S}^\top}{\boldsymbol{A}\boldsymbol{S}\boldsymbol{S}^\top}, \quad \boldsymbol{S} \leftarrow \boldsymbol{S} \circledast \frac{\boldsymbol{A}^\top\boldsymbol{X}}{\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{S}}$
nsNMF	$\boldsymbol{A} \leftarrow \boldsymbol{A} \circledast \frac{\boldsymbol{X}(\boldsymbol{M}\boldsymbol{S})^\top}{\boldsymbol{A}(\boldsymbol{M}\boldsymbol{S})(\boldsymbol{M}\boldsymbol{S})^\top}, \quad \boldsymbol{S} \leftarrow \boldsymbol{S} \circledast \frac{(\boldsymbol{A}\boldsymbol{M})^\top\boldsymbol{X}}{(\boldsymbol{A}\boldsymbol{M})^\top(\boldsymbol{A}\boldsymbol{M})\boldsymbol{S}}$
NTF	$\boldsymbol{A}^{(n)} \leftarrow \boldsymbol{A}^{(n)} \circledast \frac{\boldsymbol{X}_{(n)}\boldsymbol{S}_A^{(n)\top}}{\boldsymbol{A}^{(n)}\boldsymbol{S}_A^{(n)}\boldsymbol{S}_A^{(n)\top}}$
NTD	$\boldsymbol{A}^{(n)} \leftarrow \boldsymbol{A}^{(n)} \circledast \frac{\boldsymbol{X}_{(n)}\boldsymbol{S}_A^{(n)\top}}{\boldsymbol{A}^{(n)}\boldsymbol{S}_A^{(n)}\boldsymbol{S}_A^{(n)\top}}, \quad \boldsymbol{\mathcal{S}} \leftarrow \boldsymbol{\mathcal{S}} \circledast \frac{\boldsymbol{\mathcal{X}} \times_1 \boldsymbol{A}^{(1)\top} \cdots \times_N \boldsymbol{A}^{(N)\top}}{\boldsymbol{\mathcal{S}} \times_1 \boldsymbol{A}^{(1)\top}\boldsymbol{A}^{(1)} \cdots \times_N \boldsymbol{A}^{(N)\top}\boldsymbol{A}^{(N)}}$

Table 3. LS multiplicative update algorithms for NMF, nsNMF, NTF, and NTD, are summarized. With abuse of notations, in the case of the NTF, $\boldsymbol{S}_A^{(n)} = \left[\boldsymbol{A}^{(n-1)} \odot \cdots \odot \boldsymbol{A}^{(1)} \odot \boldsymbol{A}^{(N)} \odot \cdots \odot \boldsymbol{A}^{(n+1)}\right]^\top$, while in the case of the NTD, $\boldsymbol{S}_A^{(n)} = \boldsymbol{S}_{(n)}\boldsymbol{A}^{(\backslash n)\top}$. The updating algorithms in the NTF, are equivalent to the ones in [25, 19], but they are of a compact form.

A nsNTF, multiway generalized version of nsNMF, can be easily derived in the framework of the NTD. A tensor which has all elements zero except those for which all indices are the same is called a superdiagonal tensor. If all nonzero elements equal unity, then is is referred to as the unit superdiagonal tensor $\boldsymbol{\mathcal{I}}$. In the framework of the NTD, the cases where: (1) $\boldsymbol{\mathcal{S}} = \boldsymbol{I}$; (2) $\boldsymbol{\mathcal{S}} = \boldsymbol{M}$; (3) $\boldsymbol{\mathcal{S}} = \boldsymbol{\mathcal{I}}$, lead to (1) NMF; (2) nsNMF; (3) NTF, respectively (see Table 2). Now, we define a smoothing core tensor $\boldsymbol{\mathcal{M}}$ as

$$\boldsymbol{\mathcal{M}} = (1-\theta)\boldsymbol{\mathcal{I}} + \frac{\theta}{R^{N-1}}\mathbf{1} \circ \mathbf{1} \cdots \circ \mathbf{1}. \tag{30}$$

With this core tensor $\boldsymbol{\mathcal{M}}$ fixed, the nsNTF emerges from the NTD.

4. Numerical experiments

Our MATLAB implementation of the NTD partly uses the tensor toolbox [1]. We use AT&T and Cambridge University, ORL dataset [18] which consists of 400 face images in our experiments. We fix the location of the two eyes, crop the face, and resize to 48×48 pixel. So $48 \times 48 \times 400$ tensor and $48 \cdot 48 \times 400$ matrix are used for NTD and NMF respectively. We set $J_1 = J_2 = 24, J_3 = 30$ and $R = 30$ for NTD and NMF respectively in all experiment.

At first, we compare our initialization method with random initialization. The results, Fig. 1, show that our methods dramatically speed up convergence.

We show that the NTD indeed controls the sparseness with smoothing matrices. We set the smoothing parameter θ into 0, 0.5, and 0.8. The sparseness, measured by Hoyer[8], of $\boldsymbol{\mathcal{S}}, \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \boldsymbol{A}^{(3)}$ increases from $[0.50\ 0.56\ 0.53\ 0.44]$ to $[0.67\ 0.59\ 0.56\ 0.56]$ and $[0.82\ 0.59\ 0.57\ 0.58]$ respectively. Fig. 2 shows some basis images of each result. Slices of the $\boldsymbol{\mathcal{S}} \times_1 \boldsymbol{A}^{(1)} \times_2 \boldsymbol{A}^{(2)}$ are basis images in our setting, where image is 2D matrix rather than 1D vector. As the θ increases, basis images also become more sparser.

We synthesize simple toy data for testing nsNTF. The

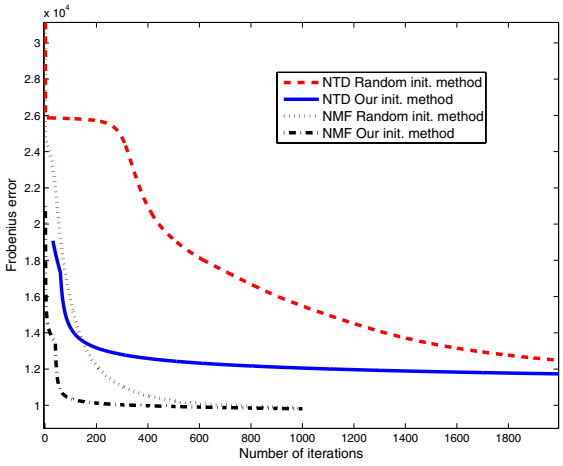


Figure 1. Frobenius error trace of: NMF and NTD with different initialization methods. Our propose methods dramatically speeds up convergence. Iterations in initialization step are also added in these error trace for fair comparisons.

Figure 2. Sample basis images. From top to bottom: (1) $\theta = 0$; (2) $\theta = 0.5$; (3) $\theta = 0.8$. As the θ increases, basis images become more sparser.

data tensor is generated by

$$\boldsymbol{\mathcal{X}} = \sum_{r=1}^{3} \boldsymbol{A}_{:,r}^{(1)} \circ \boldsymbol{A}_{:,r}^{(2)} \circ \boldsymbol{A}_{:,r}^{(3)} + |\boldsymbol{\mathcal{V}}|, \tag{31}$$

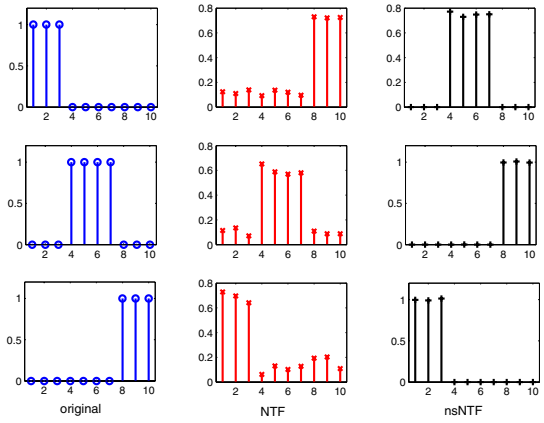


Figure 3. From left to right: (1) original signal; (2) recovered signal from the NTF; (3) recovered signal from the nsNTF wihth $\theta = 0.8$;
The nsNTF find original sparse signal successfully.

where $\boldsymbol{A}^{(1)} = \boldsymbol{A}^{(2)} = \boldsymbol{A}^{(3)}$ are 10×3 matrices and elements of $\boldsymbol{\mathcal{V}}$ follow i.i.d $\mathcal{N}(0, 0.3^2)$. The original and recovered element of $\boldsymbol{A}^{(1)}$ are shown in Fig. 3. Only the nsNTF find original sparse signal successfully.

Finally, we show the behavior of NMF and NTD, given noise-contaminated images. Each face images were superimposed by 'pepper & salt' noise. Learned basis images are shown in Fig. 5, where one can see that basis images learned by NTD are not sensitive to 'pepper & salt' noise, whereas those by NMF are deteriorated. Reconstructed images by NMF and NTD are also shown in Fig. 4, where the superiority of the NTD regarding the robustness to noise is observed. Averages peak signal-to-noise ratio (PSNR) for NMF and NTD are 22.16dB and 24.12dB, respectively. Most of pixels in a neighborhood are more likely to be similar each other, while this is not true in the case of the 'pepper & salt' noise. NMF breaks the neighborhood structure since it needs reshaping 2D images into 1D vectors. In contrast to NTD, 2D neighboring structure is preserved in the framework of 3-way tensor, leading to better reconstructed images in the case of the 'pepper & salt' noise. and this is the reason why NTD is robust to noise.

5. Conclusions

We have presented a method of nonnegative Tucker decomposition which included NMF, nsNMF, and NTF as its special cases. Exploiting standard NMF algorithms, we have derived multiplicative update algorithms for NTD in a compact form in the framework of tensor algebra. Depending on the structure of the core tensor in NTD, existing methods such as NMF, nsNMF, and NTF emerged. Initialization method which helps fast convergence and sparseness control method (nsNTD and nsNTF) also has been presented. We have verified our presented methods from numerical experiment with face image data set. Useful behavior of the NTD in image representation and image denoising has been shown. We are now investigating on analysis of a large scale sparse tensor with NTD.

Acknowledgments: This work was supported by Korea MCIE under Brain Neuroinformatics Program, and Korea MIC under ITRC support program supervised by the IITA (IITA-2006-C1090-0603-0045).

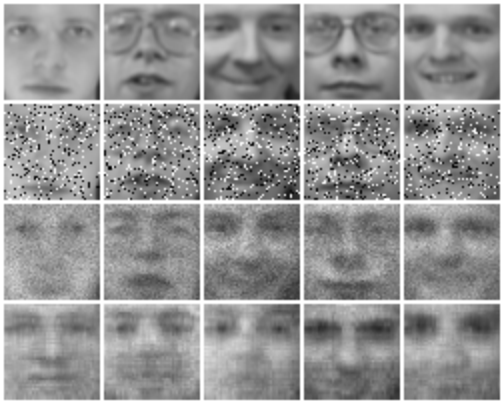
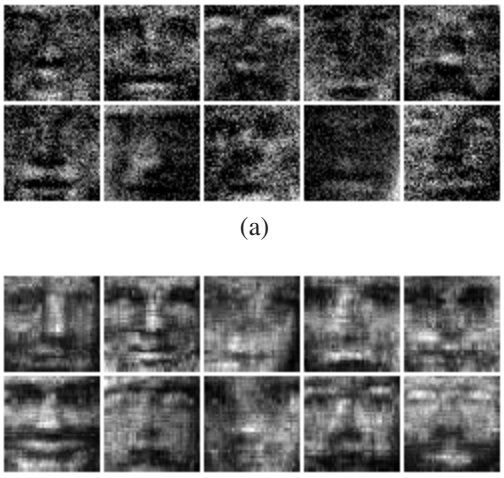


Figure 4. From top to bottom: (1) original face images; (2) images contaminated by 'pepper & salt' noise; (3) reconstructed images by NMF; (4) reconstructed images by NTD.



(a)

(b)

Figure 5. Basis images learned by (a) NMF and (b) NTD, given a set of face imaged contaminated by pepper & salt noise.

References

[1] B. W. Bader and T. G. Kolda. Algorithm 862: MATLAB tensor classes for fast algorithm prototyping. *ACM Trans. Mathematical Software*, 32(4):635–653, 2006. 6

[2] J. D. Carroll and J. J. Chang. Analysis of individual differences in multidimensional scaling via an N-way generalization of 'Eckart-Young' decomposition. *Psychometrika*, 35:283–319, 1970. 1

[3] A. Cichocki, R. Zdunek, S. Choi, R. J. Plemmons, and S. Amari. Non-negative tensor factorization using alpha and beta divergences. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Honolulu, Hawaii, 2007. 1

[4] L. de Lathauwer, B. de Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278, 2000. 1

[5] R. A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an "Exploratory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84, 1970. 1

[6] R. A. Harshman. Determination and proof of minimum uniqueness conditions for PARAFAC1. *UCLA Working Papers in Phonetics*, 22:111–117, 1972. 3

[7] T. Hazan, S. Polak, and A. Shashua. Sparse image coding using a 3D non-negative tensor factorization. In *Proceedings of International Conference on Computer Vision*, Beijing, China, 2005. 1

[8] P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004. 6

[9] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley & Sons, Inc., 2001. 1

[10] H. A. L. Kiers. Towards a standardized notation and terminology in multiway analysis. *Journal of Chemometrics*, 14:105–122, 2000. 2

[11] P. M. Kroonenberg and J. de Leeuw. Principal component analysis of three-mode data by means of alternating least squares algorithms. *Psychometrika*, 45:69–97, 1980. 1

[12] J. B. Kruskal. Three-way arrays: Rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear Algebra and its Applications*, 18(2):95–138, 1977. 3

[13] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999. 1

[14] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2001. 5

[15] M. Mørup, L. K. Hansen, and S. M. Arnfred. Sparse higher order non-negative matrix factorization. Technical report, Technical University of Denmark, 2006. http://www.mortenmorup.dk. 2

[16] P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5:111–126, 1994. 1

[17] A. Pascual-Montano, J. M. Carazo, K. K. D. Lehmann, and R. D. Pascual-Margui. Nonsmooth nonnegtive matrix facotorization (nsNMF). *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(3):403–415, 2006. 1, 2, 5

[18] F. Samaria and A. Harter. Parameterisation of a stochastic model for human face identification. In *Proceedings of 2nd IEEE Workshop on Applications of Computer Vision*, Sarasota, FL, 1994. 6

[19] A. Shashua and T. Hazan. Non-negative tensor factorization with applications to statistics and computer vision. In *Proceedings of International Conference on Machine Learning*, Bonn, Germany, 2005. 1, 3, 6

[20] L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31:279–311, 1966. 1

[21] M. A. O. Vasilescu and D. Terzopoulos. Multilinear analysis of image ensembles: Tensor faces. In *Proceedings of European Conference on Computer Vision*, Copenhagen, Denmark, 2002. 1

[22] M. A. O. Vasilescu and D. Terzopoulos. Multilinear subsapce analysis of image ensembles. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, Madison, Wisconsin, 2003. 1

[23] M. A. O. Vasilescu and D. Terzopoulos. Multilinear independent component analysis. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, San Diego, California, 2005. 1

[24] H. Wang and N. Ahuja. Rank-R approximation of tensors using image-as-matrix representation. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, San Diego, CA, 2005. 3

[25] M. Welling and M. Weber. Positive tensor factorization. *Pattern Recognition Letters*, 22:1255–1261, 2001. 1, 3, 6

[26] J. Yang, D. Zhang, A. F. Frangi, and J. Y. Yang. Two-dimensional PCA: A new approach to appearance-based face representation and recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(1):131–137, 2004. 1

[27] D. Zhang, S. Chen, and Z. H. Zhou. Two-dimensional non-negative matrix factorization for face representation and recognition. In *ICCV-2005 Workshop on Analysis and Modeling of Faces and Gestures*, 2005. 1