2008 International Symposium on Nonlinear Theory and its Applications
NOLTA'08, Budapest, Hungary, September 7-10, 2008

Multi-way Nonnegative Tensor Factorization Using Fast Hierarchical Alternating Least Squares Algorithm (HALS)

Anh Huy PHAN and Andrzej CICHOCKI

RIKEN Brain Science Institute, Wako-shi, Saitama, JAPAN
Email:{phan, cia}@brain.riken.jp

Abstract— In this paper we derive a new fast and efficient iterative algorithm for N-th order Nonnegative Tensor Factorization (NTF). We propose to use a set of local cost functions whose simultaneous or sequential (one by one) minimization via a projected gradient technique leads to simple and very efficient algorithm. The proposed algorithm exploits nonlinear projected gradient and fixed point approaches and it is a natural extension of our Hierarchical - Alternating Least Squares (HALS) algorithm developed for Nonnegative Matrix Factorization (NMF) and NTF2 [1] for sparse and noisy data. In the special case, the proposed algorithm reduces to an improved fast HALS algorithm for the standard NMF. Extensive computer simulations confirmed validity and excellent convergence properties of the proposed algorithm.

1. Introduction and Problem Formulation

In this paper, we extend the hierarchical Alternating Least Squares (HALS) algorithm for NMF [1] proposed by us recently to a multi-way Non-negative Tensor Factorization (NTF). Tensors (also known as N-way arrays or multidimensional arrays) decomposition and factorization are used in a variety of applications ranging from neuroscience, image processing and psychometrics to chemometrics [2, 3, 4, 5]. Non-negative Matrix Factorization (NMF), Nonnegative Tensor Factorization (NTF) and parallel factor analysis (PARAFAC) models with non-negativity and sparsity constraints have been recently proposed as sparse and quite efficient representations of signals, images, or general data [2, 6]. From a viewpoint of data analysis and data mining, NTF is very attractive, and more accurate than 2D matrix factorizations, such as NMF, because it takes into account spatial and temporal correlations between variables. Moreover, it usually provides sparse common loading factors or hidden (latent) components with physiological meaning and interpretation [2]. In this paper we consider the special but very important form of the PARAFAC model with additional nonnegativity (and optionally sparsity) constraints, referred here as the NTF model.

Throughout this paper, common standard notations are used as indicated in Table 1. In the general case, the NTF model can be described as a factorization of a given N-th order tensor $\underline{\boldsymbol{Y}} \in \mathbb{R}_+^{I_1 \times I_2 \cdots \times I_N}$ into set of N unknown component matrices: $\boldsymbol{A}^{(n)} = [\boldsymbol{a}_1^{(n)}, \boldsymbol{a}_2^{(n)}, \ldots, \boldsymbol{a}_R^{(n)}] \in \mathbb{R}_+^{I_n \times R}$, $n = 1, 2, \ldots, N$ representing the common (or loading) factors

$$\underline{\boldsymbol{Y}} = \widehat{\underline{\boldsymbol{Y}}} + \underline{\boldsymbol{R}} = \sum_{r=1}^{R} \lambda_r \, \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \ldots \circ \boldsymbol{a}_r^{(N)} + \underline{\boldsymbol{R}} \tag{1}$$

$$= [\![\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}]\!] + \underline{\boldsymbol{R}} = [\![\boldsymbol{\lambda}; \{\boldsymbol{A}\}]\!] + \underline{\boldsymbol{R}}, \tag{2}$$

where $\widehat{\underline{\boldsymbol{Y}}}$ is an approximation of tensor $\underline{\boldsymbol{Y}}$, and $\underline{\boldsymbol{R}}$ denotes the residue or error tensor, $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_R] \in \mathbb{R}_+^R$ are scaling factors and vectors $\boldsymbol{a}_r^{(n)}$ are unit length columns $\|\boldsymbol{a}_r^{(n)}\|_2^2 = \boldsymbol{a}_r^{(n)\top} \boldsymbol{a}_r^{(n)} = 1$. This model can be referred to CANDECOMP by Carroll and Chang [7], or PARAFAC by Harshman [8], or Kruskal [9].

The objective is to estimate nonnegative component matrices: $\boldsymbol{A}^{(n)}$ or equivalently the set of vectors $\boldsymbol{a}_r^{(n)}$, ($n = 1, 2, \ldots, N$, $r = 1, 2, \ldots, R$).

Table 1: Basic tensor operations and notations

$\circ$	outer product	$\times_n$	$n-mode$ product of tensor and matrix
$\odot$	Khatri-Rao product	$\overline{\times}_n$	$n-mode$ product of tensor and vector
$\otimes$	Kronecker product	$\underline{\boldsymbol{Y}}_{(n)}$	$n-mode$ matricized version of $\underline{\boldsymbol{Y}}$
$\circledast$	Hadamard product	$\boldsymbol{A}^{\odot}$	$\boldsymbol{A}^{(N)} \odot \boldsymbol{A}^{(N-1)} \odot \ldots \odot \boldsymbol{A}^{(1)}$
$\oslash$	element-wise division	$\boldsymbol{A}^{\odot -n}$	$\boldsymbol{A}^{(N)} \odot \cdots \odot \boldsymbol{A}^{(n+1)} \odot \boldsymbol{A}^{(n-1)} \odot \cdots \odot \boldsymbol{A}^{(1)}$
$[\bullet]_r$	r^{th} column vector of $[\bullet]$	$\boldsymbol{A}^{\circledast}$	$\boldsymbol{A}^{(N)} \circledast \boldsymbol{A}^{(N-1)} \circledast \ldots \circledast \boldsymbol{A}^{(1)}$
$\boldsymbol{A}^{(n)}$	the $n-th$ factor	$\boldsymbol{A}^{\circledast -n}$	$\boldsymbol{A}^{(N)} \circledast \cdots \circledast \boldsymbol{A}^{(n+1)} \circledast \boldsymbol{A}^{(n-1)} \circledast \cdots \circledast \boldsymbol{A}^{(1)}$
$\boldsymbol{a}_r^{(n)}$	r^{th} column vector of $\boldsymbol{A}^{(n)}$	SIR($\boldsymbol{a}$,$\boldsymbol{b}$)	$10\log_{10}(\|\boldsymbol{a}\|_2 / \|\boldsymbol{a} - \boldsymbol{b}\|_2)$
$\{\boldsymbol{a}_r\}$	$\{\boldsymbol{a}_r^{(1)}, \ldots, \boldsymbol{a}_r^{(N)}\}$	PSNR	$20\log_{10}$(Range of Signal/RMSE)
$\underline{\boldsymbol{Y}}$	tensor	Fit($\underline{\boldsymbol{Y}}$,$\widehat{\underline{\boldsymbol{Y}}}$)	$100(1 - \|\underline{\boldsymbol{Y}} - \widehat{\underline{\boldsymbol{Y}}}\|_F^2 / \|\underline{\boldsymbol{Y}} - \mathrm{E}(\underline{\boldsymbol{Y}})\|_F^2)$

2. Derivation of FAST HALS NTF Algorithm

Most of the algorithms for the NTF are based on Alternating Least Squares (ALS) minimization of the squared Euclidean distance [2, 7, 8]. In particular, we can attempt to minimize the following cost function:

$$D_F(\boldsymbol{a}_1^{(1)}, \ldots, \boldsymbol{a}_R^{(N)}) = \frac{1}{2} \left\| \underline{\boldsymbol{Y}} - \sum_{r=1}^{R} \lambda_r \, \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)} \right\|_F^2, \tag{3}$$

subject to some additional constraints such as nonnegativity and sparsity. In such a case a basic approach to the above formulated optimization problem is alternating minimization or alternating projection: the cost function is alternately minimized with respect to a sets of parameters

Dr A. Cichocki is also from Institute Research Systems, PAN Warsaw and University of Technology, Dept. of EE, Warsaw, POLAND

$\boldsymbol{a}_r^{(n)}$ each time optimizing one vector while keeping the others vectors fixed. In this paper, we consider a different approach: instead of minimizing only one global cost function, we perform sequential minimization of the set of local cost functions composed of the squared Euclidean terms (and optionally additional regularization terms):

$$D_F^{(j)}(\boldsymbol{a}_j^{(1)},\ldots,\boldsymbol{a}_j^{(N)}) = \frac{1}{2}\left\|\underline{\boldsymbol{Y}}^{(j)} - \lambda_j\, \boldsymbol{a}_j^{(1)} \circ \boldsymbol{a}_j^{(2)} \circ \cdots \circ \boldsymbol{a}_j^{(N)}\right\|_F^2 \quad (4)$$

$$= \frac{1}{2}\left\|\underline{\boldsymbol{Y}}_{(n)}^{(j)} - \lambda_j\, \boldsymbol{a}_j^{(n)} \left\{\boldsymbol{a}_j\right\}^{\odot_{-n}\top}\right\|_F^2, \quad (5)$$

for $j = 1, 2, \ldots, R$, subject to additional constraints, where

$$\underline{\boldsymbol{Y}}^{(j)} = \underline{\boldsymbol{Y}} - \sum_{r\neq j} \lambda_r\, \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)} \quad (6)$$

$$= \underline{\boldsymbol{Y}} - \sum_{r=1}^{R} \lambda_r\, \boldsymbol{a}_r^{(1)} \circ \cdots \circ \boldsymbol{a}_r^{(N)} + \lambda_j\, \boldsymbol{a}_j^{(1)} \circ \ldots \circ \boldsymbol{a}_j^{(N)}$$

$$= \underline{\boldsymbol{Y}} - \widehat{\underline{\boldsymbol{Y}}} + [\![\lambda_j, \{\boldsymbol{a}_j\}]\!]. \quad (7)$$

Note that (5) is the $n-$mode matricized version of (4). The gradients of (5) with respect to elements $\boldsymbol{a}_j^{(n)}$ are given by

$$\frac{\partial D_F^{(j)}}{\partial \boldsymbol{a}_j^{(n)}} = -\lambda_j\, \underline{\boldsymbol{Y}}_{(n)}^{(j)} \left\{\boldsymbol{a}_j\right\}^{\odot_{-n}} + \lambda_j^2\, \boldsymbol{a}_j^{(n)} \left\{\boldsymbol{a}_j\right\}^{\odot_{-n}\top} \left\{\boldsymbol{a}_j\right\}^{\odot_{-n}}. \quad (8)$$

Based on the following property of Khatri-Rao product

$$\left(\boldsymbol{A}^{(1)} \odot \boldsymbol{A}^{(2)}\right)^{\top} \left(\boldsymbol{A}^{(1)} \odot \boldsymbol{A}^{(2)}\right) = \left(\boldsymbol{A}^{(1)\top}\boldsymbol{A}^{(1)}\right) \circledast \left(\boldsymbol{A}^{(2)\top}\boldsymbol{A}^{(2)}\right), \quad (9)$$

we have $\left\{\boldsymbol{a}_j\right\}^{\odot_{-n}\top} \left\{\boldsymbol{a}_j\right\}^{\odot_{-n}} = \left\{\boldsymbol{a}_j^{\top}\boldsymbol{a}_j\right\}^{\circledast_{-n}} = \{1\}^{\circledast_{-n}} = 1$. Therefore, new fixed point learning rules for $\boldsymbol{A}^{(n)}$ obtained by equating the gradient (8) to zero are given by

$$\boldsymbol{a}_j^{(n)} \leftarrow \underline{\boldsymbol{Y}}_{(n)}^{(j)} \left\{\boldsymbol{a}_j\right\}^{\odot_{-n}} / \lambda_j. \quad (10)$$

Actually, we can neglect the denominator in (10) due to normalization $\boldsymbol{a}_j^{(n)}$ to unit length vectors:

$$\boldsymbol{a}_j^{(n)} \leftarrow \underline{\boldsymbol{Y}}_{(n)}^{(j)} \left\{\boldsymbol{a}_j\right\}^{\odot_{-n}}, \quad \boldsymbol{a}_j^{(n)} = \boldsymbol{a}_j^{(n)} / \|\boldsymbol{a}_j^{(n)}\|_2. \quad (11)$$

The weight (scaling) factor λ_j can be estimated from vector $\boldsymbol{a}_j^{(N)}$ as $\lambda_j = \|\boldsymbol{a}_j^{(N)}\|_2$ before normalizing this vector. Note that the learning rule (11) has an equivalent expression given by

$$\boldsymbol{a}_j^{(n)} \leftarrow \underline{\boldsymbol{Y}}^{(j)}\, \overline{\times}_1 \boldsymbol{a}_j^{(1)\top} \cdots \overline{\times}_{n-1} \boldsymbol{a}_j^{(n-1)\top} \overline{\times}_{n+1} \boldsymbol{a}_j^{(n+1)\top} \cdots \overline{\times}_N \boldsymbol{a}_j^{(N)\top}$$
$$= \underline{\boldsymbol{Y}}^{(j)}\, \overline{\times}_{-n} \{\boldsymbol{a}_j^{\top}\}. \quad (12)$$

The above updating formula is compact but with high computational cost. In the following section, we will derive an efficient realization of this learning rule. From the definition of Khatri-Rao product and the property that Khatri-Rao and Kronecker products of two vectors are identical, we have

$$\left[\boldsymbol{A}^{(1)} \odot \boldsymbol{A}^{(2)}\right]_j = \left[\boldsymbol{a}_1^{(1)} \otimes \boldsymbol{a}_1^{(2)} \;\ldots\; \boldsymbol{a}_R^{(1)} \otimes \boldsymbol{a}_R^{(2)}\right]_j = \boldsymbol{a}_j^{(1)} \odot \boldsymbol{a}_j^{(2)}, \quad (13)$$

Table 2: **FAST-HALS NTF**

1: Nonnegative random or ALS initialization $\boldsymbol{A}^{(n)}$ [1]
2: Compute $\boldsymbol{\lambda}: \lambda_r = \|\boldsymbol{a}_r^{(N)}\|_2$
3: Normalize all $\boldsymbol{a}_r^{(n)}$ to unit length
4: $\boldsymbol{T}_1 = (\boldsymbol{A}^{(1)\top}\boldsymbol{A}^{(1)}) \circledast \ldots \circledast (\boldsymbol{A}^{(N)\top}\boldsymbol{A}^{(N)})$
5: **repeat**
6: **for** $n = 1$ to N **do**
7: $\boldsymbol{T}_2 = \underline{\boldsymbol{Y}}_{(n)} \{\boldsymbol{A}^{\odot_{-n}}\}$
8: $\boldsymbol{T}_3 = \boldsymbol{T}_1 \oslash (\boldsymbol{A}^{(n)\top}\boldsymbol{A}^{(n)})$
9: **for** $r = 1$ to R **do**
10: $\boldsymbol{a}_r^{(n)} \Leftarrow \left[\lambda_r \boldsymbol{a}_r^{(n)} + [\boldsymbol{T}_2]_r - \boldsymbol{A}^{(n)}\ \mathrm{diag}(\boldsymbol{\lambda})\ [\boldsymbol{T}_3]_r\right]_+$ [2]
11: **if** n=N **then**
12: $\lambda_r = \|\boldsymbol{a}_r^{(N)}\|_2$
13: **end if**
14: Normalize $\boldsymbol{a}_r^{(n)}$ to unit length $\boldsymbol{a}_r^{(n)} = \boldsymbol{a}_r^{(n)} / \|\boldsymbol{a}_r^{(n)}\|_2$
15: **end for**
16: $\boldsymbol{T}_1 = \boldsymbol{T}_3 \circledast \boldsymbol{A}^{(n)\top}\boldsymbol{A}^{(n)}$
17: **end for**
18: **until** convergence criterion is reached

and in more general case,

$$\left\{\boldsymbol{a}_j\right\}^{\odot_{-n}} = \left[\boldsymbol{A}^{\odot_{-n}}\right]_j. \quad (14)$$

By replacing $\underline{\boldsymbol{Y}}_{(n)}^{(j)}$ in (11) by (7), and taking into account (14), the learning rule (11) can be expressed as follows

$$\boldsymbol{a}_j^{(n)} \leftarrow \underline{\boldsymbol{Y}}_{(n)} \left[\boldsymbol{A}^{\odot_{-n}}\right]_j - \widehat{\underline{\boldsymbol{Y}}}_{(n)} \left[\boldsymbol{A}^{\odot_{-n}}\right]_j + [\![\lambda_j, \{\boldsymbol{a}_j\}]\!]_{(n)} \left\{\boldsymbol{a}_j\right\}^{\odot_{-n}}$$
$$= \left[\underline{\boldsymbol{Y}}_{(n)}\boldsymbol{A}^{\odot_{-n}}\right]_j - \boldsymbol{A}^{(n)}\boldsymbol{D}_\lambda \boldsymbol{A}^{\odot_{-n}\top} \left[\boldsymbol{A}^{\odot_{-n}}\right]_j + \lambda_j \boldsymbol{a}_j^{(n)} \left\{\boldsymbol{a}_j\right\}^{\odot_{-n}\top} \left\{\boldsymbol{a}_j\right\}^{\odot_{-n}}$$
$$= \left[\underline{\boldsymbol{Y}}_{(n)}\boldsymbol{A}^{\odot_{-n}}\right]_j - \boldsymbol{A}^{(n)}\boldsymbol{D}_\lambda \left[\boldsymbol{A}^{\odot_{-n}\top}\boldsymbol{A}^{\odot_{-n}}\right]_j + \lambda_j \boldsymbol{a}_j^{(n)}$$
$$= \left[\underline{\boldsymbol{Y}}_{(n)}\boldsymbol{A}^{\odot_{-n}}\right]_j - \boldsymbol{A}^{(n)}\boldsymbol{D}_\lambda \left[\left\{\boldsymbol{A}^{\top}\boldsymbol{A}\right\}^{\circledast_{-n}}\right]_j + \lambda_j \boldsymbol{a}_j^{(n)}$$
$$= \left[\underline{\boldsymbol{Y}}_{(n)}\boldsymbol{A}^{\odot_{-n}}\right]_j - \boldsymbol{A}^{(n)}\boldsymbol{D}_\lambda \left[\{\boldsymbol{A}^{\top}\boldsymbol{A}\}^{\circledast} \oslash \left(\boldsymbol{A}^{(n)\top}\boldsymbol{A}^{(n)}\right)\right]_j + \lambda_j \boldsymbol{a}_j^{(n)}, \quad (15)$$

where $\boldsymbol{D}_\lambda = \mathrm{diag}(\boldsymbol{\lambda})$ is a diagonal matrix whose diagonal entries are λ_j.

The equation (15) represents the new fast learning rule for $\boldsymbol{a}_j^{(n)}$. In combination with a componentwise nonlinear operator defined as $[\boldsymbol{A}]_+ = \max\{0, \boldsymbol{A}\}$, we finally have a new algorithm referred as Fast HALS NTF algorithm

$$\boldsymbol{a}_j^{(n)} \leftarrow \left[\lambda_j \boldsymbol{a}_j^{(n)} + \left[\underline{\boldsymbol{Y}}_{(n)}\boldsymbol{A}^{\odot_{-n}}\right]_j - \boldsymbol{A}^{(n)}\boldsymbol{D}_\lambda \left[\{\boldsymbol{A}^{\top}\boldsymbol{A}\}^{\circledast} \oslash \left(\boldsymbol{A}^{(n)\top}\boldsymbol{A}^{(n)}\right)\right]_j\right]_+. \quad (16)$$

The detail pseudo-code of this algorithm is given in Table (2). In a special case of $N = 2$, FAST-HALS NTF becomes FAST-NMF for NMF problem.

2.1. Experiments

Extensive simulations were performed for synthetic and real-world data on a 2.66 GHz Quad-Core Windows 64-bit machine with 8GB memory. Results were compared with

[1] For 3-way tensor, direct trilinear decomposition could be used as initialization.

[2] In practice, vectors $\boldsymbol{a}_j^{(n)}$ are often fixed sign before rectifying.

NMWF [3], lsNTF [10] algorithms and also with two implementations of PARAFAC ALS algorithm [7, 8] by Kolda and Bader [11] (denoted as ALS_K) and by Andersson and Bro [12] (denoted as ALS_B) under the same condition of difference of fit value (1e-5) through three performance indices: Peak Signal to Noise Ratio (PSNR) for all frontal slices, Signal to Interference Ratio (SIR) for each columns of factors and the explained variation ratio (Fit) for a whole tensor.

Two noisy tensors generated by three benchmarks `X_spectra_sparse, ACPos24sparse10` and `X_spectra` [6] with size of $300 \times 300 \times 300$ (Example 1(a)) and $500 \times 500 \times 500$ (Example 1(b)) have been corrupted by Gaussian noise with SNR = 0dB. Illustrations for Example 1(a) with volume, iso-surface and factor visualizations are given in Fig. 1(b), 1(c) and 1(d); while running time and distributions of SIR and PSNR indices are depicted in Fig. 2 and also are available in Table 4. Fast HALS provides high accuracy for factor estimation based on SIR index, and the highest explained variation with the fastest running time.

Examples 2-6 show Fast HALS NTF in use with real world data sets which are described in Table 3. In Examples 2 and 3, Amino acids fluorescence data from five samples containing tryptophan, phenylalanine, and tyrosine (`claus.mat`) [13, 12] and Sugar process data (`sugar.mat`) [4] were corrupted by Gaussian noise with SNR = 0dB before reconstruction with $R = 5$ rank-one tensors. Comparisons of performance and running time for Example 2 are illustrated in Fig. 3 (also in Table 5). There are two illustrations of estimated factors for amino acids and sugar tensor factorized with 3 components by FAST HALS NTF given in Fig. 4(a) and 4(b), respectively. In Example 4, DOSY nuclear magnetic resonance lipoproteins [5] $268 \times 53 \times 7$ was factorized with 4 components.

For EEG data, we used two data sets: `tutorialdataset1.set` (Example 5) and `tutorialdataset2.zip` (Example 6) [14]. After pre-processing data set `tutorialdataset1.set` by complex Morlet wavelet, it yields 64 channels $\times$ 72 epoches time-frequency spectra of 72×61. This data was converted to a 3-way tensor of size $64 \times 2769 \times 72$ before factorizing with 2 components. For Example 6, its data set contains the inter-trial phase coherence (ITPC) of 14 subjects during a proprioceptive pull of left and right hand (28 files) and generates tensor $64 \times 4392 \times 28$ [14]. Illustration of this example for FAST HALS NTF is given in Fig. 5 with scalp topographic maps and their correspondent IPTC time-frequency measurements.

For real world data sets, standard ALS algorithms returned a slightly higher performance than that of Fast HALS algorithm. However, their components contain negative elements in order to enforce them to be orthogonal. Therefore, FAST HALS NTF could be considered to achieve the highest performance for nonnegative data.

Through these examples, it is shown that FAST HALS NTF algorithm is robust to noise and produces the best performance with the fastest running time.

Table 3: Description of data sets and notation of Examples

Exp.	Data set	Size	R
1(a)	`X_spectra_sparse, ACPos24sparse10`	$300 \times 300 \times 300$	4
1(b)	and `X_spectra` [6]	$500 \times 500 \times 500$	4
2	Amino acids fluorescence data, `claus.mat` [13]	$5 \times 201 \times 61$	5
3	Sugar process data, `sugar.mat` [4]	$268 \times 53 \times 7$	5
4	DOSY NMR of lipoproteins, `webnmr.mat` [5]	$25 \times 24 \times 1600$	4
5	64 channels of EEG measurements $64 channels \times (71 frequency - 39 time) \times 72 epoches$, `tutorialdataset1.set` [14]	$72 \times 2769 \times 64$	2
6	ITPC of 14 subjects during a proprioceptive pull of left and right hand (28 datasets), $64 channels \times (61 frequency - 72 time) \times 28 subjects$, `tutorialdataset2.set` [14]	$64 \times 4392 \times 28$	3

Table 4: Performance Comparison for Examples 1(a)-(b)

Exp.		FastNTF			NMWF [1]			lsNTF			ALS_B			ALS_K		
		$A^{(1)}$	$A^{(2)}$	$A^{(3)}$	$A^{(1)}$	$A^{(2)}$	$A^{(3)}$	$A^{(1)}$	$A^{(2)}$	$A^{(3)}$	$A^{(1)}$	$A^{(2)}$	$A^{(3)}$	$A^{(1)}$	$A^{(2)}$	$A^{(3)}$
1(a)	SIR (dB)	**44.07**	45.35	41.19	43.64	43.30	34.07	43.99	**45.43**	41.20	43.11	39.38	41.18	43.10	39.72	**41.21**
		44.64	44.80	39.17	44.59	**44.93**	**39.32**	44.64	44.81	39.16	44.42	38.95	39.26	44.18	39.00	39.13
		41.21	42.36	**39.94**	40.98	41.45	39.84	41.21	**42.40**	39.75	37.93	35.21	39.66	37.94	35.23	39.68
		42.37	**42.76**	40.93	42.40	42.31	**41.00**	42.41	42.75	40.87	**42.54**	36.83	40.66	41.87	36.92	40.68
	Fit (%)		**99.9880**			99.9832			99.9866			99.9873			99.9872	
	Time (sec)		**6.1290**			56.5240			2829.9800			15.0800			36.1116	
		$A^{(1)}$	$A^{(2)}$	$A^{(3)}$	$A^{(1)}$	$A^{(2)}$	$A^{(3)}$				$A^{(1)}$	$A^{(2)}$	$A^{(3)}$	$A^{(1)}$	$A^{(2)}$	$A^{(3)}$
1(b)	SIR (dB)	**47.09**	**48.14**	**44.18**	47.08	47.45	43.86	✗			46.66	41.06	44.15	46.71	41.04	44.17
		50.04	**50.45**	47.39	45.82	41.78	**48.73**	(fail)			49.23	42.19	47.44	49.16	44.15	47.42
		46.39	**47.75**	46.82	46.22	41.42	**46.88**				43.51	39.99	44.15	44.48	40.04	44.31
		47.18	**47.43**	**45.60**	45.27	40.98	45.21				44.19	39.42	45.37	44.44	39.42	45.37
	Fit (%)		**99.9955**			99.9918		✗				99.9953			99.9953	
	Time (sec)		**51.7322**			513.3666		✗				145.7306			965.7621	

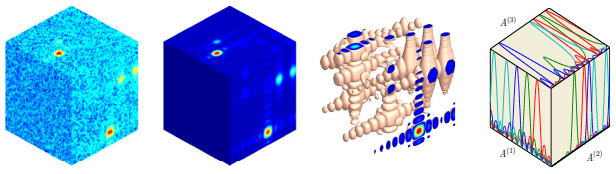
Table 5: Performance Comparison for Examples 2-6

	Fit (%)					**Time** (seconds)				
Exp.	2	3	4	5	6	2	3	4	5	6
FastNTF	**98.0218**	**99.1366**	**99.5027**	**69.7721**	**52.4131**	**0.36**	**0.46**	**8.55**	**4.52**	**7.08**
NMWF	97.8153	99.0940	99.4981	69.7681	52.3806	1.69	6.29	41.64	30.65	58.19
lsNTF	97.6360	98.9133	99.3414	46.3228	51.3317	3.30	38.93	101.70	2667.70	4029.84
ALS_B	96.9702	99.0521	**99.5053**	**69.7762**	**53.1749**	0.87	1.42	20.53	14.22	67.24
ALS_K	97.0264	99.0812	99.3933	69.7734	53.1336	0.46	0.66	17.19	36.99	66.39

3. Conclusions and Discussion

The main objective and motivations of this paper is to derive fast and efficient algorithm which is suitable to NTF problem. The extended algorithm FAST-HALS NTF is verified for many different benchmarks. The developed algorithm is robust to noisy data and has many potential applications. This algorithm is also suitable to large scale dataset due to its local learning rules, and fast processing speed. We can easily increase number of rank one tensors in processing progress. The algorithm can be extended to semi-NTF and to sparse PARAFAC using suitable nonlinear projections [15].

[1] In fact, the NMWF fails for noisy data in Examples 1-3 cause of negative elements. We enforced the estimated components to have nonnegative values by half-wave rectifying.



(a) Noisy data (b) Volume 99.99% (c) Iso-surface 99.99% (d) Factors

Figure 1: Illustration of tensor reconstruction by Fast HALS NTF for Example 1(a) with tensor $\underline{\boldsymbol{Y}} \in \mathbb{R}_{+}^{300\times300\times300}$ degraded by Gaussian noise with SNR = 0dB.

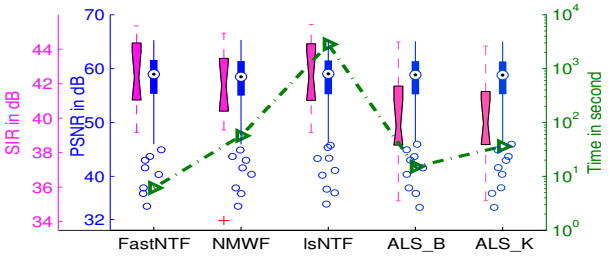


Figure 2: Comparison of performance and running time for Example 1(a) with tensor corrupted by Gaussian noise with SNR = 0dB.

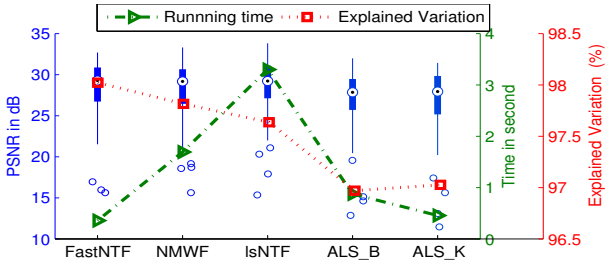
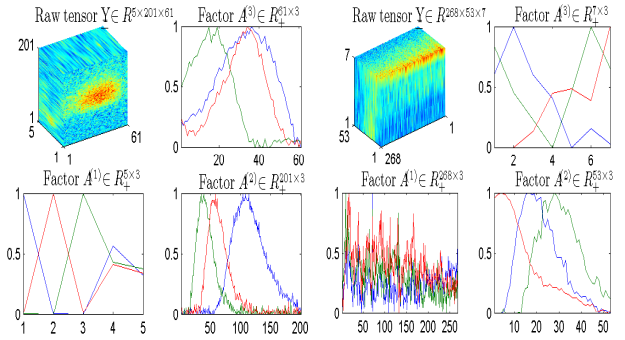
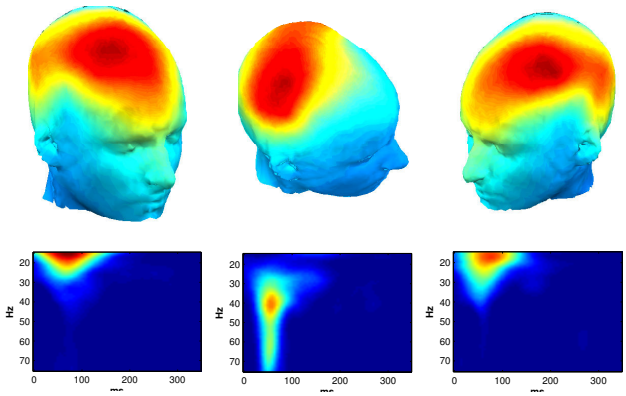


Figure 3: Comparison of performance and running time for Example 2 with tensor $\underline{\boldsymbol{Y}} \in \mathbb{R}_{+}^{5\times201\times61}$ corrupted by Gaussian noise with SNR = 0dB.



(a) Amino acid tensor (b) Sugar tensor

Figure 4: Illustration of estimated factors by FAST-HALS NTF for two datasets: (a) amino acid data; (b) sugar data.



(a) Left hand stimuli (b) Gamma activity of both stimuli (c) Right hand stimuli

Figure 5: Illustration of FAST HALS NTF for Example 6 with factor $\boldsymbol{A}^{(1)}$ for scalp topographic map (first row), factor $\boldsymbol{A}^{(2)}$ for spectral map (second row) (see [14] for details).

References

[1] Cichocki, A., Zdunek, R., Amari, S.: Hierarchical ALS Algorithms for Nonnegative Matrix and 3D Tensor Factorization. In: Lecture Notes in Computer Science. Volume 4666. (2007) 169–176

[2] Smilde, A., Bro, R., Geladi, P.: Multi-way Analysis: Applications in the Chemical Sciences. John Wiley and Sons, New York (2004)

[3] Mørup, M., Hansen, L.K., Parnas, J., Arnfred, S.M.: Decomposing the time-frequency representation of EEG using non-negative matrix and multi-way factorization. Technical report (2006)

[4] Bro, R.: Exploratory study of sugar production using fluorescence spectroscopy and multi-way analysis. Chemom. Intell. Lab. Syst

[5] Dyrby, M., Peteresen, M., Whittaker, A.D., Lambert, L., Nrgaard, L., Bro, R., Engelsen, S.B.: Analysis of lipoproteins using 2D diffusion-edited NMR spectroscopy and multi-way chemometrics. Anal.Chim.Acta **531** (2005) 209–216

[6] Cichocki, A., Zdunek, R.: NMFLAB – NTFLAB for Signal and Image Processing. Technical report, Laboratory for Advanced Brain Signal Processing, BSI, RIKEN, `http://www.bsp.brain.riken.jp` (2006)

[7] Carroll, J.D., Chang, J.J.: Analysis of Individual Differences in Multidimensional Scaling via an N-way Generalization of Eckart–Young Decomposition. Psychometrika **35**(3) (1970) 283–319

[8] Harshman, R.A.: Foundations of the PARAFAC procedure: Models and conditions for an "explanatory"multimodal factor analysis. UCLA Working Papers in Phonetics **16** (1970) 1–84

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

[10] Friedlander, M.P., Hatz, K.: Computing nonnegative tensor factorizations. Tech. Rep. TR-200621, Dept. Computer Science, University of British Columbia, Vancouver (December 2007) To appear in *Optimization Methods and Software*.

[11] Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. Technical Report SAND2007-6702, Sandia National Laboratories, Albuquerque, NM and Livermore, CA (November 2007)

[12] Andersson, C.A., Bro, R.: The *N*-way Toolbox for MATLAB. Chemometrics Intell. Lab. Systems **52** (2000) 1–4

[13] Bro, R.: PARAFAC. Tutorial and applications. In: Special Issue 2nd Internet Conf. in Chemometrics (INCINC'96). Volume 38., Chemom. Intell. Lab. Syst (1997) 149–171

[14] Mørup, M., Hansen, L.K., Arnfred, S.M.: ERPWAVELAB a toolbox for multi-channel analysis of time-frequency transformed event related potentials. Journal of Neuroscience Methods **161** (2007) 361–368

[15] Cichocki, A., Phan, A.H., Zdunek, R., Zhang, L.Q.: Flexible component analysis for sparse, smooth, nonnegative coding or representation. In: Lecture Notes in Computer Science. Volume 4984., Springer (2008) 811–820