FAST DAMPED GAUSS-NEWTON ALGORITHM FOR SPARSE AND NONNEGATIVE TENSOR FACTORIZATION

Anh Huy Phan[†], *Petr Tichavský*[‡*] *and Andrzej Cichocki*[†]

[†]Brain Science Institute, RIKEN, Wakoshi, Japan
[‡]Institute of Information Theory and Automation, Prague, Czech Republic

ABSTRACT

Alternating optimization algorithms for canonical polyadic decomposition (with/without nonnegative constraints) often accompany update rules with low computational cost, but could face problems of swamps, bottlenecks, and slow convergence. All-at-once algorithms can deal with such problems, but always demand significant temporary extra-storage, and high computational cost. In this paper, we propose an all-at-once algorithm with low complexity for sparse and nonnegative tensor factorization based on the damped Gauss-Newton iteration. Especially, for low-rank approximations, the proposed algorithm avoids building up Hessians and gradients, reduces the computational cost dramatically. Moreover, we proposed selection strategies for regularization parameters. The proposed algorithm has been verified to overwhelmingly outperform "state-of-the-art" NTF algorithms for difficult benchmarks, and for real-world application such as clustering of the ORL face database.

Index Terms— canonical polyadic decomposition (CP), nonnegative tensor factorization, Gauss-Newton, Levenberg-Marquardt, face clustering, sparsity, low rank approximation

1. INTRODUCTION

Canonical polyadic decomposition (CP) with nonnegative constraints also coined nonnegative tensor factorization (NTF) has been found in many important applications [1, 2], and can be formulated as follows: "Factorize a given nonnegative N-th order data tensor $\underline{\mathbf{Y}} \in \mathbb{R}_+^{I_1 \times I_2 \cdots \times I_N}$ into a set of N nonnegative factors $\mathbf{A}^{(n)} = [\boldsymbol{a}_1^{(n)}, \boldsymbol{a}_2^{(n)}, \ldots, \boldsymbol{a}_R^{(n)}] \in \mathbb{R}_+^{I_n \times R}$, $(n = 1, 2, \ldots, N)$", that is,

$$\underline{\mathbf{Y}} \approx \underline{\mathbf{I}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \cdots \times_N \mathbf{A}^{(N)} = \underline{\hat{\mathbf{Y}}}, \qquad (1)$$

where $\underline{\mathbf{I}}$ is an identity tensor, "$\times_n$" denotes product of a tensor and a matrix along the mode-n. In real-world applications such as EEG analysis [2], face clustering [3], and gene expression clustering [4, 5], sparse solutions are often preferred to non-sparse.

*The work of P. Tichavsky was supported by Ministry of Education, Youth and Sports of the Czech Republic through the project 1M0572 and by Grant Agency of the Czech Republic through the project 102/09/1278.

Most NTF algorithms minimize the cost function

$$D(\underline{\mathbf{Y}}, \{\mathbf{A}^{(n)}\}) = \|\underline{\mathbf{Y}} - \underline{\mathbf{I}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \cdots \times_N \mathbf{A}^{(N)}\|_F^2, \quad (2)$$

via alternating optimization which often accompanies update rules with low computational cost, but faces problems of swamps, bottlenecks, and slow convergence. All-at-once algorithms which simultaneously update all the factors can deal with such problems. However, these algorithms always demand significant temporary extra-storage, and high computational cost of large-scale gradients, Hessians or Jacobian matrices. In this paper, we propose an all-at-once algorithm with low complexity for sparse and nonnegative CP based on the damped Gauss-Newton (dGN) iteration. A logarithmic barrier penalty term and an ℓ_1-norm penalty function have been imposed on the cost function (2) to enforce nonnegativity and sparsity constraints. Especially, for low-rank approximation, the proposed algorithm avoids building up Hessians and gradients, reduces the computational cost dramatically. Moreover, we propose method to adaptively select regularization parameters. The proposed algorithm is verified to overwhelmingly outperform "state-of-the-art" NTF algorithms for difficult benchmarks, and clustering of the ORL face database.

2. DAMPED GAUSS-NEWTON ALGORITHM

Paatero [6] proposed the PMF3 algorithm for NTF which minimizes the cost function (2) with a logarithmic penalty function to prevent factors $\mathbf{A}^{(n)}$ reaching zeros. However, PMF faces problems of large-scale Hessian and Jacobian, and selection of regularization parameter. To this end, the proposed algorithm considers a similar cost function (2) with additional penalty terms P_l and P_s to enforce nonnegativity and sparsity constraints on factors $\mathbf{A}^{(n)}$, respectively,

$$D_+ = D(\underline{\mathbf{Y}}, \{\mathbf{A}^{(n)}\}) + P_l(\{\mathbf{A}^{(n)}\}) + P_s(\{\mathbf{A}^{(n)}\}), \qquad (3)$$

$$P_l = -\sum_{n=1}^{N} \alpha_n \sum_{i_n=1}^{I_n} \sum_{r=1}^{R} \log(a_{i_n r}^{(n)}), \quad P_s = \sum_{n=1}^{N} \beta_n \left\|\mathbf{A}^{(n)}\right\|_1, \qquad (4)$$

where parameters $\alpha_n > 0$ and $\beta_n > 0$ for $n = 1, 2, \ldots, N$. The update rule derived from the cost function (3) simultaneously

updates all the factors $\mathbf{A}^{(n)}$ based on the damped GN iteration [6], and can be expressed in a common formula as

$$\boxed{\boldsymbol{v} \leftarrow \boldsymbol{v} - \left(\mathbf{H} + \mu \mathbf{I}\right)^{-1} \boldsymbol{g},} \tag{5}$$

where $\boldsymbol{v}^T = \left[\text{vec}\left(\mathbf{A}^{(n)}\right)^T\right]_{n=1}^{N}$, $\mu > 0$ is the damping parameter. The gradient $\boldsymbol{g}$ and the Hessian $\mathbf{H}$ are given by

$$\begin{aligned}
\boldsymbol{g} &= \mathbf{J}^T(\hat{\boldsymbol{y}} - \boldsymbol{y}) + \frac{\partial P_l}{\partial \boldsymbol{v}} + \frac{\partial P_s}{\partial \boldsymbol{v}} \\
&= \mathbf{J}^T(\hat{\boldsymbol{y}} - \boldsymbol{y}) - \left[\left(\alpha_n \ \text{vec}\left(\mathbf{A}^{(n)}\right)^{\cdot[-1]} - \beta_n\right)^T\right]_{n=1}^{N} \quad^T, \quad (6) \\
\mathbf{H} &= \mathbf{J}^T \mathbf{J} + \frac{\partial^2 P_l}{\partial \boldsymbol{v}^2} + \frac{\partial^2 P_s}{\partial \boldsymbol{v}^2} \\
&= \mathbf{J}^T \mathbf{J} + \bigoplus_{n=1}^{N} \text{diag}\left\{\alpha_n \ \text{vec}\left(\mathbf{A}^{(n)}\right)^{\cdot[-2]}\right\}, \quad (7) \\
\mathbf{J} &= \left[\begin{array}{cccc} \mathbf{P}_1 \ (\mathbf{A}^{\odot -1} \otimes \mathbf{I}_1) & \cdots & \mathbf{P}_N \ (\mathbf{A}^{\odot -N} \otimes \mathbf{I}_N) \end{array}\right], \quad (8)
\end{aligned}$$

where $\hat{\boldsymbol{y}} = \text{vec}(\underline{\hat{\mathbf{Y}}})$, $\boldsymbol{y} = \text{vec}(\underline{\mathbf{Y}})$, $\bigoplus$ denotes a direct sum: $\mathbf{A} \oplus \mathbf{B} = \text{diag}\{\mathbf{A}, \mathbf{B}\}$, and $[\boldsymbol{x}]^{\cdot[p]}$ denotes element-wise powers, $\odot$ denotes Khatri-Rao product and $\mathbf{A}^{\odot -n} = \mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)}$, $\mathbf{P}_n$ is a permutation matrix: $\text{vec}(\underline{\mathbf{Y}}) = \mathbf{P}_n \ \text{vec}\left(\mathbf{Y}_{(n)}\right), n = 1, 2, \ldots, N$. The Jacobian matrix $\mathbf{J} \in \mathbb{R}^{(\prod I_n) \times RT}, T = \sum_{n=1}^{N} I_n$ can be directly utilized in the learning rule (5). However, this demands high computational cost for construction and inverse of the approximate Hessian $(\mathbf{H} + \mu \mathbf{I})^{-1}$. In the sequence, we present more efficient computation methods for the learning rule (5).

2.1. Fast computation of the gradient $\boldsymbol{g}$

From (8) and (2), we have the following result [7]

$$\left(\frac{\partial \hat{\boldsymbol{y}}}{\partial \mathbf{A}^{(n)}}\right)^T (\boldsymbol{y} - \hat{\boldsymbol{y}}) = \text{vec}\left(\mathbf{Y}_{(n)} \mathbf{A}^{\odot -n} - \mathbf{A}^{(n)} \mathbf{\Gamma}^{(n,n)}\right),$$

where $\mathbf{Y}_{(n)}$ is the mode-n matricization of the tensor $\underline{\mathbf{Y}}$, $\mathbf{\Gamma}^{(n,n)} = \circledast_{m \neq n} \mathbf{A}^{(m)T} \mathbf{A}^{(m)}$ is Hadamard product of $(N-1)$ matrices $\mathbf{A}^{(m)T} \mathbf{A}^{(m)}, m \neq n$. Therefore, by denoting $\mathbf{F}^{(n)} = \mathbf{Y}_{(n)} \mathbf{A}^{\odot -n} - \mathbf{A}^{(n)} \mathbf{\Gamma}^{(n,n)}$, the gradient (6) can be expressed as

$$\boldsymbol{g}^T = \left[\text{vec}\left(-\mathbf{F}^{(n)} - \alpha_n \left(\mathbf{A}^{(n)}\right)^{\cdot[-1]} + \beta_n\right)^T\right]_{n=1}^{N}. \tag{9}$$

2.2. Construction and Inverse of Hessian H

The Hessian $\tilde{\mathbf{H}} = \mathbf{H} + \mu \mathbf{I}$ can be expressed as the concatenation of $R I_n \times R I_m$ dimensional block matrices $\mathbf{H}^{(n,m)}$ as $\mathbf{H} + \mu \mathbf{I} = \left[\mathbf{H}^{(n,m)}\right]$.

Theorem 1 (Expression of block matrix $\mathbf{H}^{(n,m)}$)**.** *A diagonal block matrix $\mathbf{H}^{(n,n)}$ can be expressed by*

$$\mathbf{H}^{(n,n)} = \mathbf{\Gamma}^{(n,n)} \otimes \mathbf{I}_n + \text{diag}\left\{\alpha_n \ \text{vec}\left(\mathbf{A}^{(n)}\right)^{\cdot[-2]} + \mu\right\}, \tag{10}$$

and a block matrix $\mathbf{H}^{(n,m)}$ for $n \neq m$ is given by

$$\mathbf{H}^{(n,m)} = (\mathbf{I}_R \otimes \mathbf{A}^{(n)}) \, \mathbf{P}_{R,R} \, \text{diag}(\text{vec}(\mathbf{\Gamma}^{(n,m)}))(\mathbf{I}_R \otimes \mathbf{A}^{(m)T}), \tag{11}$$

where $\mathbf{\Gamma}^{(n,m)} = \circledast_{k \neq m,n} \mathbf{A}^{(k)T} \mathbf{A}^{(k)}$, and $\mathbf{P}_{R,R}$ is the commutation matrix of an $R \times R$ matrix $\mathbf{X}$: $\text{vec}(\mathbf{X}) = \mathbf{P}_{R,R} \, \text{vec}\left(\mathbf{X}^T\right)$.

Proof. Proof is directly derived from (8) and (7). □

Theorem 2 (Low rank Adjustment for the Hessian $\tilde{\mathbf{H}}$)**.** *The Hessian $\tilde{\mathbf{H}}$ can be decomposed under the form as*

$$\tilde{\mathbf{H}} = \mathbf{G} + \mathbf{Z} \, \mathbf{K} \, \mathbf{Z}^T. \tag{12}$$

where $\mathbf{K} = \left[\mathbf{K}^{(n,m)}\right]_{n,m} \in \mathbb{R}^{NR^2 \times NR^2}$ is a partitioned matrix with sub-matrices $\mathbf{K}^{(n,m)}$ given by

$$\mathbf{K}^{(n,m)} = \begin{cases} \mathbf{P}_{R,R} \ \text{diag}\left(\text{vec}\left(\mathbf{\Gamma}^{(n,m)}\right)\right), & m \neq n, \\ \mathbf{0}, & m = n, \end{cases} \tag{13}$$

and matrices $\mathbf{G} \in \mathbb{R}^{TR \times TR}$ and $\mathbf{Z} \in \mathbb{R}^{TR \times R^2}$ are given by

$$\mathbf{G} = \mathbf{P} \left(\bigoplus_{n=1}^{N} \bigoplus_{i_n=1}^{I_n} \left(\mathbf{\Gamma}^{(n,n)} + \text{diag}\{\alpha_n \boldsymbol{a}_{i_n:}^{(n)\cdot[-2]} + \mu\}\right)\right) \mathbf{P}^T, \tag{14}$$

$$\mathbf{Z} = \bigoplus_{n=1}^{N} \left(\mathbf{I}_R \otimes \mathbf{A}^{(n)}\right), \tag{15}$$

in which $\mathbf{P} = \bigoplus_{n=1}^{N} \mathbf{P}_{R,I_n}$, $\mathbf{P}_{R,I_n}$ is a commutation matrix of an $I_n \times R$ dimensional matrix $\mathbf{X}$: $\text{vec}(\mathbf{X}) = \mathbf{P}_{R,I_n} \, \text{vec}\left(\mathbf{X}^T\right)$.

Proof. Proof is derived from Theorem 1. □

Theorem 2 allows to inverse the large Hessian via the binomial inverse theorem

$$\tilde{\mathbf{H}}^{-1} = \mathbf{G}^{-1} - \mathbf{G}^{-1} \mathbf{Z} \left(\mathbf{K}^{-1} + \mathbf{Z}^T \mathbf{G}^{-1} \mathbf{Z}\right)^{-1} \mathbf{Z}^T \mathbf{G}^{-1}. \tag{16}$$

We note that $\mathbf{G}^{-1}$ can be efficiently computed as

$$\mathbf{G}^{-1} = \mathbf{P} \left(\bigoplus_{n=1}^{N} \bigoplus_{i_n=1}^{I_n} \mathbf{\Theta}_{i_n}^{(n)}\right) \mathbf{P}^T, \tag{17}$$

where $\mathbf{\Theta}_{i_n}^{(n)} = \left(\mathbf{\Gamma}^{(n,n)} + \text{diag}\left\{\alpha_n \boldsymbol{a}_{i_n:}^{(n)\cdot[-2]} + \mu\right\}\right)^{-1}$. The inverse of the partitioned matrix $\mathbf{K}$ has an explicit expression given in [7], and is rewritten in following theorem.

Theorem 3 (Inverse of matrix $\mathbf{K}$)**.** *Inverse of the kernel matrix $\mathbf{K}$ is a partitioned matrix $\mathcal{K} = \mathbf{K}^{-1}$ whose diagonal blocks $\mathcal{K}^{(n,n)}$, for $n = 1, \ldots, N$ are given by*

$$\mathcal{K}^{(n,n)} = \frac{2 - N}{N - 1} \mathbf{P}_{R,R} \ \text{diag}\left(\text{vec}\left(\mathbf{A}^{(n)T} \mathbf{A}^{(n)} \oslash \mathbf{\Gamma}^{(n,n)}\right)\right), \tag{18}$$

and other blocks $\mathcal{K}^{(n,m)}, \forall n \neq m$ are given by

$$\mathcal{K}^{(n,m)} = \frac{1}{N - 1} \mathbf{P}_{R,R} \ \text{diag}\left(\mathbf{1} \oslash \text{vec}\left(\mathbf{\Gamma}^{(n,m)}\right)\right). \tag{19}$$

Products of matrices in (16) can be computed as

$$\mathbf{L} = \mathbf{G}^{-1}\mathbf{Z}^T = \mathbf{P}\left(\bigoplus_{n=1}^{N}\boldsymbol{\Upsilon}^{(n)}\right)(\mathbf{I}_N \otimes \mathbf{P}_{R,R}), \quad (20)$$

$$\mathbf{Z}^T\mathbf{G}^{-1}\mathbf{Z} = (\mathbf{I}\otimes\mathbf{P}_{R,R})\bigoplus_{n=1}^{N}\boldsymbol{\Psi}^{(n)}(\mathbf{I}\otimes\mathbf{P}_{R,R}), \quad (21)$$

where $\boldsymbol{\Upsilon}^{(n)T} = \left[\boldsymbol{a}_{i_n:}^{(n)T} \otimes \boldsymbol{\Theta}_{i_n}^{(n)T}\right]_{i_n=1}^{I_n}$, and $\boldsymbol{\Psi}^{(n)} = \left[\boldsymbol{\Psi}_{r,s}^{(n)}\right]$ is a partitioned matrix of sub-matrices $\boldsymbol{\Psi}_{r,s}^{(n)} = \sum_{i_n=1}^{I_n}\boldsymbol{\Theta}_{i_n}^{(n)}\, a_{i_n,r}^{(n)}\, a_{i_n,s}^{(n)}$, for $r = 1,2,\ldots,R$ and $s = 1,2,\ldots,R$.

From (9), (16), (17), (20), (21) and Theorem 3, the update rule (5) is formulated in a more efficient form as follows

$$\boxed{\boldsymbol{v} \leftarrow \boldsymbol{v} - \mathbf{G}^{-1}\boldsymbol{g} + \mathbf{L}\boldsymbol{\Phi}^{-1}\left(\mathbf{L}^T\boldsymbol{g}\right),} \quad (22)$$

where $\boldsymbol{\Phi} = \mathbf{K}^{-1} + \mathbf{Z}^T\mathbf{G}^{-1}\mathbf{Z} \in \mathbb{R}^{NR^2\times NR^2}$. We note that for low rank approximation $R \ll I_n, \forall n$, the matrix $\boldsymbol{\Phi}$ is much smaller than the Hessian $\mathbf{H} \in \mathbb{R}^{RT\times RT}$. Inverses of the matrix $\boldsymbol{\Phi}$ requires computational complexity of order $O(N^3R^6)$, while $(\mathbf{H}+\mu\mathbf{I})^{-1}$ has a computational complexity of order $O(R^3T^3)$ or $O(N^3R^3I^3)$ for a symmetric tensor $I_1 = \cdots = I_N = I$. As a consequence, solving an inverse problem $\boldsymbol{\psi} = \boldsymbol{\Phi}^{-1}\boldsymbol{w}$, $\boldsymbol{w} = \mathbf{L}^T\boldsymbol{g}$ in the update rule (22) is much less expensive than solving the problem $(\mathbf{H}+\mu\mathbf{I})^{-1}\boldsymbol{g}$ in the update rule (5). Moreover, because the learning rule (22) does not need to construct the Hessian $\mathbf{H}$ and the Jacobian $\mathbf{J}$, this update rule is significantly faster than the rule (5).

3. SELECTION OF BARRIER AND SPARSE PARAMETERS

Paatero [6] suggested an heuristic approach to control the barrier parameter $\alpha = \alpha_1 = \alpha_2 = \cdots = \alpha_N$. The parameter α should be initialized by a large enough value, then slowly descends down to near-zero after each 10 iterations. This strategy is efficient in almost all cases. However, we cannot control the convergence speed. An alternative approach is to adaptively select regularization parameters based on the Karush-Kuhn-Tucker (KKT) condition. In this direction, Rojas and Steihaug updated barrier parameter at each iteration [8]. Ding *et al.* derived an optimal formula for regularization parameters in orthogonal NMF [9]. The sparsity parameters β_n are often fixed to a small enough value. By employing this method, regularization parameters α_n and β_n for $\forall n$ are selected based on the KKT slackness condition

$$\boldsymbol{v} \geqslant 0, \quad \boldsymbol{v} \circledast \boldsymbol{g} = 0, \quad \boldsymbol{g} \geqslant 0. \quad (23)$$

From conditions (23), and the gradient (9), we obtain

$$\mathbf{A}^{(n)} \circledast \left(\mathbf{F}^{(n)} + \alpha_n\left(\mathbf{A}^{(n)}\right)^{\cdot[-1]} - \beta_n\right) = \mathbf{0}, \quad \forall n, (24)$$

$$\mathbf{F}^{(n)} + \alpha_n\left(\mathbf{A}^{(n)}\right)^{\cdot[-1]} - \beta_n \leqslant \mathbf{0}, \quad \forall n. (25)$$

This leads to solve a constrained LS problem

$$\min_{\boldsymbol{x}} \|\mathbf{B}^{(n)}\boldsymbol{x} - \boldsymbol{u}^{(n)}\|_2^2 \text{ such that } \begin{cases}\mathbf{B}^{(n)}\boldsymbol{x} \leqslant \boldsymbol{u}^{(n)},\\ \boldsymbol{x} = [\alpha_n\,\beta_n]^T \geqslant \mathbf{0},\end{cases} \quad \forall n, (26)$$

where $\mathbf{B}^{(n)} = \left[\mathbf{1}\ \text{vec}\left(-\mathbf{A}^{(n)}\right)\right]$, and $\boldsymbol{u}^{(n)} = \text{vec}\left(-\mathbf{A}^{(n)} \circledast \mathbf{F}^{(n)}\right)$.

For NTF without the penalty term $\mathbf{P}_s$, conditions (23) lead to the result $\mathbf{A}^{(n)} \circledast \mathbf{F}^{(n)} + \alpha_n \leqslant \mathbf{0}, \forall n$. The parameters α_n can be chosen as

$$\alpha_n = \max\left(0, \min\left(\text{vec}\left(-\mathbf{A}^{(n)} \circledast \mathbf{F}^{(n)}\right)\right)\right), \quad \forall n. \quad (27)$$

4. SIMULATIONS

4.1. Synthetic data

In the first simulation, the proposed algorithm (LM_+) will be verified and compared with the multiplicative KL and LS [2], ALS, HALS [10], QALS [11] algorithms for benchmarks including bias, collinear factors. We constructed 3-D synthetic tensors $\underline{\mathbf{Y}}$ with $I_1 = I_2 = I_3 = 100$ composed from random factors comprising $R = 10$ components. The components were forced to be collinear with others by a simple modification $\boldsymbol{a}_r^{(n)} = \boldsymbol{a}_1^{(n)} + 0.5\,\boldsymbol{a}_r^{(n)}$, $(r = 2,3,\ldots,R)$. All the algorithms were initialized using leading singular components, and stopped when difference of the consecutive relative errors $\varepsilon = \frac{\|\underline{\mathbf{Y}}-\hat{\underline{\mathbf{Y}}}\|_F^2}{\|\underline{\mathbf{Y}}\|_F^2}$ was lower than 10^{-10}, or the maximum number of iterations (200) was exceeded. The SIR index ($SIR = -20\log_{10}\frac{\|\boldsymbol{a}_r^{(n)}-\hat{\boldsymbol{a}}_r^{(n)}\|_2}{\|\boldsymbol{a}_r^{(n)}\|_2}$ dB) is calculated for the true and estimated components after permutation matching and normalization. Comparison of performances of various algorithms averaged over 100 runs is given in Table 1. The proposed algorithm achieved almost perfect performances with $\varepsilon \approx$ 2.87e-9 after only 67 (averaged) iterations. The other algorithms whose SIR indices are listed in the 4-th column of Table 1 could not estimate hidden components in 200 iterations. To analyze their convergences, we set new stopping criterion with one million iterations, and $\varepsilon \leqslant 10^{-10}$. Most algorithms except the KL algorithm converged to the desired results with averaged SIRs $>$ 30 dB (given in the 5-th column of Table 1). Fig. 1 illustrates relative errors as functions of iterations for NTF algorithms for one MC run. The Lm_+ converged after 104 iterations, whereas the QALS stopped after 50K iterations, respectively. To yields comparable performances to that of our algorithm, the other algorithms need much more iterations that ours. Moreover, in general the other algorithms often achieve a different solution because the optimization problem is hard and the algorithms get stacked in a side local minimum more frequently than our algorithm.

4.2. Clustering of the ORL face database

This example considers the ORL face database consisting of 400 faces for 40 subjects. In the first analysis, 100 faces from

Table 1. Performance comparison for various algorithms for Example 4.1.

Algorithm	Error	No. Iters	SIR (dB)	SIR* (dB)
KL	$(1.56 \pm 0.39)\ 10^{-3}$	200	5 ± 5	13 ± 3
LS	$(1.02 \pm 0.17)\ 10^{-4}$	200	14 ± 2	33 ± 6
ALS	$(7.83 \pm 6.02)\ 10^{-6}$	200	19 ± 4	63 ± 37
HALS	$(10.6 \pm 0.83)\ 10^{-6}$	200	13 ± 3	57 ± 5
QALS	$(15.0 \pm 7.26)\ 10^{-7}$	200	20 ± 2	140 ± 73
LM_+	$(\mathbf{2.87 \pm 20.2})\ 10^{-9}$	**67**	$\mathbf{97 \pm 16}$	$\mathbf{150 \pm 23}$

*: algorithms were run till the error $\varepsilon < 10^{-10}$ in 1M iterations.

Table 2. Accuracies (Acc) and normalized mutual information (NMI) for various algorithms for Example 4.2.

Algorithm	10 classes			30 classes		40 classes	
	Error	Acc (%)	NMI	Acc (%)	NMI	Acc (%)	NMI
KL	4.51e-2	80.00	8.64e-1	92.67	9.49e-1	82.25	9.08e-1
LS	1.27e-2	88.00	9.06e-1	96.33	9.75e-1	85.75	9.42e-1
ALS	6.79e-2	93.00	9.20e-1	95.33	9.71e-1	85.50	9.30e-1
HALS	**1.17e-2**	91.00	9.10e-1	97.00	9.76e-1	86.25	9.47e-1
QALS	1.18e-2	92.00	9.21e-1	98.33	9.84e-1	88.00	9.50e-1
LM_+	**1.17e-2**	94.00	9.44e-1	99.00	9.90e-1	87.25	9.44e-1
LM_+s	1.24e-2	**100**	**1.00**	**99.67**	**9.97e-1**	**92.75**	**9.69e-1**

the first 10 subjects were down-sampled, then vectorized to give a (400×100) matrix $\mathbf{Y}$. We applied NMF to find $R = 20$ features for each face, and used the K-means algorithm to cluster them. The accuracy (%) and normalized mutual information (NMI) for algorithms are given in Table 2. The proposed algorithm explained data with a lowest relative error and achieved a higher clustering accuracy than the other algorithms. Especially, the LM_+ with sparsity constraints (LM_+s) successfully clustered the selected faces.

Next, we constructed 32 Gabor feature tensors of 8 orientations at 4 scales which were then down-sampled to $16 \times 16 \times 32 \times 400$ dimensional tensor $\underline{\mathbf{Y}}$. Because of low correlation or rare common parts between Gabor features which are not in the same levels (orientations and scales), we found common bases $\mathbf{A}^{(n_l)} \in \mathbb{R}^{16 \times R_l}$, $n = 1, 2$ for 3-D sub-tensors $\underline{\mathbf{Y}}_l = \underline{\mathbf{Y}}(:, :, l, :) \in \mathbb{R}^{16 \times 16 \times 400}$ $(l = 1, 2, \ldots, 32)$ along the two first dimensions: $\underline{\mathbf{Y}}_l \approx \underline{\mathbf{I}}_l \times_1 \mathbf{A}^{(1_l)} \times_2 \mathbf{A}^{(2_l)} \times_3 \mathbf{A}^{(3_l)}$. In this experiment, we set $R_l = 8, \forall l$. Hence, a sample had 256 $(= 8 \times 32)$ features compressed from 8192 $(= 16 \times 16 \times 32)$ Gabor features. In the second stage, the matrix of features $\mathbf{X} \in \mathbb{R}^{400 \times 256}$ was factorized to reduce the number of features to the number of classes. Finally, the data was clustered using the K-means algorithm. In Table 2, we compare clustering performances for various algorithms. For clustering of faces for the first 30 subjects, the LM_+ achieved 99% accuracy, outperformed the other algorithms. The LM_+ with sparsity constraints (LM_+s) slightly improved performance up to an accuracy of 99.67%. Increasing the number of classes to 35 or 40 subjects, the LM_+s always gave the highest performances.

5. CONCLUSIONS

A robust and low complexity dGN algorithm has been proposed for NTF. Instead of solving large-scale inverse problems of size $RT \times RT$ with $O(R^3T^3)$ or $O(N^3R^3I^3)$ for symmetric tensors, the proposed algorithm considers problems of much smaller matrices of size $NR^2 \times NR^2$, with a dramatically reduced computational cost of $O(N^3R^6)$ for $R \ll I$. Selection strategies of regularization parameters have also been proposed. Especially, the proposed algorithm with sparsity constraints improved the clustering performance for the ORL face database.

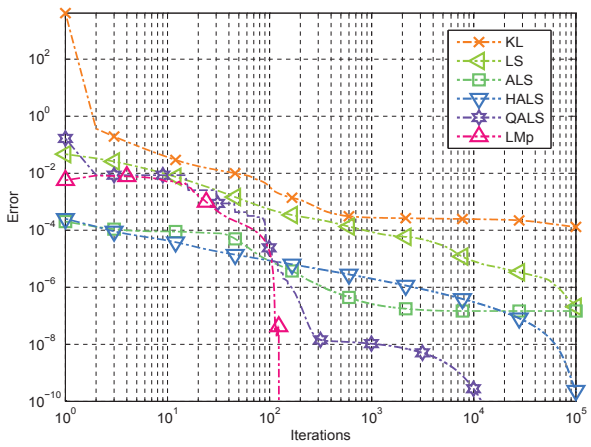

Fig. 1. Convergence of NTF algorithms for Example 4.1.

6. REFERENCES

[1] A. Cichocki, R Zdunek, A.-H. Phan, and S. Amari, *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*, Wiley, Chichester, 2009.

[2] M. Mørup, L.K. Hansen, and S.M. Arnfred, "ERPWAVELAB a toolbox for multi-channel analysis of time-frequency transformed event related potentials," *Journal of Neuroscience Methods*, vol. 161, no. 2, pp. 361–368, 2006.

[3] M. Heiler and C. Schnörr, "Controlling sparseness in nonnegative tensor factorization," in *ECCV*. 2006, pp. 56–67, Springer.

[4] P. Carmona-Saez, R.D. Pascual-Marqui, F. Tirado, J.M. Carazo, and A. Pascual-Montano, "Biclustering of gene expression data by non-smooth non-negative matrix factorization," *BMC Bioinformatics*, vol. 7, no. 78, 2006.

[5] H. Kim and H. Park, "Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis," *Bioinformatics*, vol. 23, no. 12, pp. 1495–1502, 2007.

[6] P. Paatero, "A weighted non-negative least squares algorithm for three-way PARAFAC factor analysis," *Chemometrics Intelligent Laboratory Systems*, vol. 38, no. 2, pp. 223–242, 1997.

[7] A.-H. Phan, P. Tichavský, and A. Cichocki, "Low complexity damped Gauss-Newton algorithms for parallel factor analysis," *SIAM, SIMAX*, 2010, (under review).

[8] M. Rojas and T. Steihaug, "An interior-point trust-region-based method for large-scale non-negative regularization," *Inverse Problems*, vol. 18, pp. 1291–1307, 2002.

[9] C. Ding, T. Li, W. Peng, and H. Park, "Orthogonal nonnegative matrix tri-factorizations for clustering," in *KDD06*, New York, NY, USA, 2006, pp. 126–135, ACM Press.

[10] A. Cichocki and A.-H. Phan, "Fast local algorithms for large scale nonnegative matrix and tensor factorizations," *IEICE (invited paper)*, March 2009.

[11] A.-H. Phan, A. Cichocki, R. Zdunek, and Th. Vu-Dinh, "Novel alternating least squares algorithm for nonnegative matrix and tensor factorizations," in *(ICONIP)*, 2010.