A FURTHER IMPROVEMENT OF A FAST DAMPED GAUSS–NEWTON ALGORITHM FOR CANDECOMP-PARAFAC TENSOR DECOMPOSITION

Petr Tichavský[1], *Anh Huy Phan*[2], *and Andrzej Cichocki*[2]

[1]Institute of Information Theory and Automation,
P.O.Box 18, 182 08 Prague 8, Czech Republic
[2]Brain Science Institute, RIKEN, Wakoshi, Japan. E-mail: phan@brain.riken.jp.

ABSTRACT

In this paper, a novel implementation of the damped Gauss-Newton algorithm (also known as Levenberg-Marquart) for the CANDECOMP-PARAFAC (CP) tensor decomposition is proposed. The method is based on a fast inversion of the approximate Hessian for the problem. It is shown that the inversion can be computed on $O(NR^6)$ operations, where N and R is the tensor order and rank, respectively. It is less than in the best existing state-of-the art algorithm with $O(N^3R^6)$ operations. The damped Gauss-Newton algorithm is suitable namely for difficult scenarios, where nearly-colinear factors appear in several modes simultaneously. Performance of the method is shown on decomposition of large tensors ($100 \times 100 \times 100$ and $100 \times 100 \times 100 \times 100$) of rank 5 to 90.

***Index Terms*—** Multilinear models; canonical polyadic decomposition; damped Gauss-Newton algorithm

1. INTRODUCTION

Tensor decomposition has recently received increased attention with the interest in chemistry [1, 2, 3], astronomy, telecommunication [4, 5], neural sciences [6, 7], data mining [8, 9], separated representations involved in quantum mechanics [10], classification, clustering [11], and compression [12]. Canonical decomposition or Parallel factor analysis (CANDECOMP / PARAFAC or CP), is a widely used tensor factorization in all these applications, cf. [13, 14].

Among existing algorithms for CP decomposition (CPD), the damped Gauss-Newton algorithms [15, 16, 17, 18], also known as the Levenberg-Marquardt algorithm has been shown to be highly efficient in treatment of difficult decompositions such as tensor with nearly-collinear factors in several modes, or large difference in magnitude between components [18, 19, 20, 21]. Unfortunately, the standard LM algorithms for CPD [15, 16] are computationally demanding due to computation of the large-scale Jacobian, gradients and inversion of the large-scale approximate Hessian which normally costs $\mathcal{O}(R^3T^3)$ where T and R are perimeter and rank of the tensor. Explicit expressions for submatrices of the Hessian were then derived by Paatero [20] and Tomasi [3], which allowed to bypass construction of the Jacobian. In order to reduce complexity of inverse of $\mathbf{H}$, Paatero [20] employed the Cholesky decomposition of the approximate Hessian and back substitution. Tomasi [22] suggested using QR decomposition. However, the LM algorithms were still computationally demanding. Tichavský and Koldovský [23] proposed a method to invert the approximate Hessian based on $3R^2 \times 3R^2$ dimensional matrices for 3-way tensor. Recently, the fast LM (fLM) algorithm for CPD has been proposed in [18]. Complexity of each iteration of the fLM algorithm is $O(N^3R^6)$ where N is the tensor order. In this paper we propose a further improvement to reduce the complexity per iteration of the fLM algorithm to $O(NR^6)$ operations.

[0]The work of Petr Tichavský was supported by the Czech Science Foundation through the project 102/09/1278.

2. PROBLEM FORMULATION

Let $\mathcal{Y}$ be an $N-$ way tensor of dimension $I_1 \times I_2 \times \ldots \times I_N$. The tensor is said to be of rank R, if R is the smallest number of rank-one tensors which admits the decomposition of $\mathcal{Y}$ of the form

$$\mathcal{Y} = \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \ldots \circ \mathbf{a}_r^{(N)} \tag{1}$$

where $\circ$ denotes the outer vector product, $\mathbf{a}_r^{(n)}$, $r = 1, \ldots, R$, $n = 1, \ldots, N$ are vectors of the length I_n called factors. The tensor in (1) can be characterized by N factor matrices $\mathbf{A}_n = [\mathbf{a}_1^{(n)}, \mathbf{a}_2^{(n)}, \ldots, \mathbf{a}_R^{(n)}]$ of the size $I_n \times R$ for $n = 1, \ldots, N$. The model (1) is also referred to as a Kruskal form of a tensor [13].

Let a vector parameter $\boldsymbol{\theta}$ containing all parameters of our model be arranged as

$$\boldsymbol{\theta} = [(\mathtt{vec}\, \mathbf{A}_1)^T, \ldots, (\mathtt{vec}\, \mathbf{A}_N)^T]^T \tag{2}$$

The maximum likelihood solution for $\boldsymbol{\theta}$ consists in minimizing the least squares criterion

$$\mathcal{Q}(\boldsymbol{\theta}) = \|\hat{\mathcal{Y}} - \mathcal{Y}(\boldsymbol{\theta})\|_F^2 \tag{3}$$

where $\hat{\mathcal{Y}}$ is a noisy observation of the tensor (with i.i.d Gaussian noise), and $\|\cdot\|_F$ stands for the Frobenius norm.

The damped Gauss-Newton algorithm, sometimes called the Levenberg-Marquardt algorithm, uses updates of the estimated parameter in the form

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + (\mathbf{H} + \mu \mathbf{I}_T)^{-1} \mathbf{J}^T \mathtt{vec}(\hat{\mathcal{Y}} - \mathcal{Y}(\boldsymbol{\theta})) \tag{4}$$

where $\mathbf{H} = \mathbf{H}(\boldsymbol{\theta}) = \mathbf{J}^T(\boldsymbol{\theta})\mathbf{J}(\boldsymbol{\theta})$ is the approximate Hessian, $\mathbf{J} = \mathbf{J}(\boldsymbol{\theta})$ is the Jacobian (matrix of the first-order derivatives) of $\mathtt{vec}[\mathcal{Y}(\boldsymbol{\theta})]$ with respect to $\boldsymbol{\theta}$, $\mathbf{I}_T$ is the identity matrix of the size $T \times T$, $T = \dim(\boldsymbol{\theta}) = R\sum_n I_n$, and $\mu > 0$ is a regularization parameter.

A popular update rule for the regularization parameter was proposed in [25]. In the CP decomposition algorithms, the term $\mathbf{g} = \mathbf{J}^T \mathtt{vec}(\hat{\mathcal{Y}} - \mathcal{Y}(\boldsymbol{\theta}))$ is sometimes called CP gradient or MTTKRP (Matricized Tensor Times Khatri-Rao Product) [24]. It appears in all of the known algorithms. A fast method of computing the CP gradient was proposed in [26].

3. PROBLEM SOLUTION

Let $\boldsymbol{\Gamma}_{nm}$ be defined as a Hadamard (elementwise) product of matrices $\mathbf{C}_k = \mathbf{A}_k^T \mathbf{A}_k, k \in \{1, \ldots, N\} - \{n, m\}$, that is

$$\boldsymbol{\Gamma}_{nm} = \circledast_{k \neq n,m} \mathbf{C}_k\,, \qquad \mathbf{C}_k = \mathbf{A}_k^T \mathbf{A}_k\,. \tag{5}$$

Theorem 1 [18]: The Hessian $\mathbf{H}$ can be decomposed into low rank matrices under the form as

$$\mathbf{H} = \mathbf{G} + \mathbf{Z}\,\mathbf{K}\,\mathbf{Z}^T \tag{6}$$

where $\mathbf{K} = [\mathbf{K}_{nm}]_{n,m=1}^N$ contains submatrices $\mathbf{K}_{nm}$ given by

$$\mathbf{K}_{nm} = (1 - \delta_{nm})\mathbf{P}_R\ \mathrm{dvec}\,(\boldsymbol{\Gamma}_{nm}) \tag{7}$$

$\mathbf{P}_R$ is the permutation matrix of dimension $R^2 \times R^2$ defined in [18] such that $\mathtt{vec}(\mathbf{M}) = \mathbf{P}_R\,\mathtt{vec}\,(\mathbf{M}^T)$ for any $R \times R$ matrix $\mathbf{M}$, δ_{nm} is the Kronecker delta, and $\mathtt{dvec}(\mathbf{M})$ is a short-hand notation for $\mathtt{diag}(\mathtt{vec}(\mathbf{M}))$, i.e. a diagonal matrix containing all elements of a matrix $\mathbf{M}$ on its main diagonal. Next,

$$\mathbf{G} = \mathtt{bdiag}\,(\boldsymbol{\Gamma}_{nn} \otimes \mathbf{I}_{I_n})_{n=1}^N \tag{8}$$

and

$$\mathbf{Z} = \mathtt{bdiag}\,(\mathbf{I}_R \otimes \mathbf{A}_n)_{n=1}^N \tag{9}$$

where $\otimes$ denotes the Kronecker product, $\mathbf{I}_{I_n}$ is an identity matrix of the size $I_n \times I_n$, and $\mathtt{bdiag}(\cdot)$ is a block diagonal matrix with the given blocks on its diagonal. Note that the Hessian $\mathbf{H}$ in (6) is rank deficient because of the scale ambiguity of columns of factor matrices [20, 17]. It has the dimension $T \times T$, $T = R\sum_n I_n$, but its rank is at most $T - (N-1)R$.

Theorem 1 allows to inverse the large Hessian via employing a much smaller matrix by using the binomial inverse theorem (32). In particular, we wish to invert "diagonally loaded" Hessian $\mathbf{H}_\mu = \mathbf{H} + \mu\mathbf{I}_T$,

$$\begin{aligned} \mathbf{H}_\mu^{-1} &= \left(\mathbf{G}_\mu + \mathbf{Z}\,\mathbf{K}\,\mathbf{Z}^T\right)^{-1} \\ &= \mathbf{G}_\mu^{-1} - \mathbf{G}_\mu^{-1}\mathbf{Z}\mathbf{B}_\mu^{-1}\mathbf{Z}^T\mathbf{G}_\mu^{-1} \end{aligned} \tag{10}$$

where

$$\begin{aligned} \mathbf{G}_\mu &= \mathbf{G} + \mu\,\mathbf{I}_T \\ &= \mathtt{bdiag}\left((\boldsymbol{\Gamma}_{nn} + \mu\,\mathbf{I}_R) \otimes \mathbf{I}_{I_n}\right)_{n=1}^N && (11) \\ {\mathbf{G}_\mu}^{-1} &= \mathtt{bdiag}\left((\boldsymbol{\Gamma}_{nn} + \mu\,\mathbf{I}_R)^{-1} \otimes \mathbf{I}_{I_n}\right)_{n=1}^N && (12) \\ \mathbf{B}_\mu &= \mathbf{K}^{-1} + \mathbf{Z}^T\mathbf{G}_\mu^{-1}\mathbf{Z}\,. && (13) \end{aligned}$$

We can see that inversion of the Hessian can be done through an inversion of the matrix $\mathbf{B}_\mu$, which has the size $NR^2 \times NR^2$. Complexity of the inversion is $O(N^3R^6)$. We show, however, that the matrix can be inverted in $O(NR^6)$ operations.

The fast inversion of $\mathbf{B}_\mu$ is based on the observation that

$$\begin{aligned} \mathbf{K} &= \mathbf{K}_0 + \mathbf{D}\mathbf{F}\mathbf{D}^T && (14) \\ \mathbf{K}_0 &= \mathtt{bdiag}(\mathbf{K}_n)_{n=1}^N && (15) \\ \mathbf{K}_n &= -\mathbf{P}_R\,\mathtt{dvec}(\boldsymbol{\Gamma}_{nn} \oslash \mathbf{C}_n) && (16) \\ \mathbf{D} &= \begin{bmatrix} \mathbf{D}_1 \\ \vdots \\ \mathbf{D}_N \end{bmatrix}, \qquad \mathbf{D}_n = \mathtt{dvec}(1 \oslash \mathbf{C}_n) && (17) \\ \mathbf{F} &= \mathbf{P}_R\,\mathtt{dvec}(\boldsymbol{\Gamma}), \qquad \boldsymbol{\Gamma} = \circledast_{n=1}^N \mathbf{C}_n && (18) \end{aligned}$$

where $\oslash$ denotes the elementwise division. Then

$$\begin{aligned} \mathbf{B}_\mu &= \mathbf{K}^{-1} + \mathbf{Z}^T\mathbf{G}_\mu^{-1}\mathbf{Z} \\ &= (\mathbf{K}_0 + \mathbf{D}\mathbf{F}\mathbf{D}^T)^{-1} + \mathbf{Z}^T\mathbf{G}_\mu^{-1}\mathbf{Z} \\ &= \mathbf{K}_0^{-1} - \mathbf{K}_0^{-1}\mathbf{D}(\mathbf{F}^{-1} + \mathbf{D}^T\mathbf{K}_0^{-1}\mathbf{D})^{-1}\mathbf{D}^T\mathbf{K}_0^{-1} \\ &\quad + \mathbf{Z}^T\mathbf{G}_\mu^{-1}\mathbf{Z} \\ &= \mathbf{K}_0^{-1} + \mathbf{Z}^T\mathbf{G}_\mu^{-1}\mathbf{Z} - \mathbf{K}_0^{-1}\mathbf{D}\mathbf{Q}^{-1}\mathbf{D}^T\mathbf{K}_0^{-1} \\ &= \mathbf{J}_\mu - \mathbf{K}_0^{-1}\mathbf{D}\mathbf{Q}^{-1}\mathbf{D}^T\mathbf{K}_0^{-1} && (19) \end{aligned}$$

where

$$\begin{aligned} \mathbf{Q} &= \mathbf{F}^{-1} + \mathbf{D}^T\mathbf{K}_0^{-1}\mathbf{D} && (20) \\ \mathbf{J}_\mu &= \mathbf{K}_0^{-1} + \mathbf{Z}^T\mathbf{G}^{-1}\mathbf{Z} = \mathrm{bdiag}(\mathbf{J}_n)_{n=1}^N && (21) \\ \mathbf{J}_n &= \mathbf{K}_n^{-1} + (\mathbf{I}_R \otimes \mathbf{A}_n)^T \cdot \\ &\quad \cdot ((\boldsymbol{\Gamma}_{nn} + \mu\mathbf{I}_R)^{-1} \otimes \mathbf{I}_{I_n})(\mathbf{I}_R \otimes \mathbf{A}_n) \\ &= -\mathbf{P}_R\,\mathrm{dvec}(\mathbf{C}_n \oslash \boldsymbol{\Gamma}_{nn}) \\ &\quad + (\boldsymbol{\Gamma}_{nn} + \mu\mathbf{I}_R)^{-1} \otimes \mathbf{C}_n\,. && (22) \end{aligned}$$

Note that $\mathbf{Q}$ can be simplified to

$$\begin{aligned}\mathbf{Q} &= \mathbf{F}^{-1} + \sum_{n=1}^{N} \mathbf{D}_n \mathbf{K}_n^{-1} \mathbf{D}_n \\ &= \mathbf{P}_R \operatorname{dvec}(1 \oslash \mathbf{\Gamma}) - \sum_{n=1}^{N} \mathbf{P}_R \operatorname{dvec}(1 \oslash (\mathbf{\Gamma}_{nn} \circledast \mathbf{C}_n)) \\ &= -(N-1)\, \mathbf{P}_R \operatorname{dvec}(1 \oslash \mathbf{\Gamma}) \,. \end{aligned} \tag{23}$$

Now, applying (32) to (19) we get

$$\begin{aligned}\mathbf{B}_\mu^{-1} &= \mathbf{J}_\mu^{-1} + \mathbf{J}_\mu^{-1} \mathbf{K}_0^{-1} \mathbf{D} \mathbf{S}_\mu^{-1} \mathbf{D}^T \mathbf{K}_0^{-1} \mathbf{J}_\mu^{-1} \\ &= \mathbf{J}_\mu^{-1} + \mathbf{L}_\mu \mathbf{S}_\mu^{-1} \mathbf{L}_\mu^T \end{aligned} \tag{24}$$

where

$$\begin{aligned}\mathbf{S}_\mu &= \mathbf{Q} - \mathbf{D}^T \mathbf{K}_0^{-1} \mathbf{J}_\mu^{-1} \mathbf{K}_0^{-1} \mathbf{D} \\ &= \mathbf{Q} - \sum_{n=1}^{N} \mathbf{D}_n \mathbf{K}_n^{-1} \mathbf{J}_n^{-1} \mathbf{K}_n^{-1} \mathbf{D}_n \end{aligned} \tag{25}$$

and

$$\mathbf{L}_\mu = \mathbf{J}_\mu^{-1} \mathbf{K}_0^{-1} \mathbf{D} \,. \tag{26}$$

We can see that the inverse of $\mathbf{B}_\mu$, which normally needs $O(N^3 R^6)$ operations, can be computed through inversions of smaller matrices $\mathbf{J}_1, \ldots, \mathbf{J}_N$ and $\mathbf{S}_\mu$ of the size $R^2 \times R^2$ in $O(NR^6)$ operations.

The inverse of $\mathbf{H}_\mu$ can be written also in a block form as

$$(\mathbf{H}_\mu)^{-1} = \widetilde{\mathbf{H}}_\mu = \begin{bmatrix} \widetilde{\mathbf{H}}_\mu^{(1,1)} & \cdots & \widetilde{\mathbf{H}}_\mu^{(1,N)} \\ \vdots & \ddots & \vdots \\ \widetilde{\mathbf{H}}_\mu^{(N,1)} & \cdots & \widetilde{\mathbf{H}}_\mu^{(N,N)} \end{bmatrix} \tag{27}$$

where

$$\begin{aligned}\widetilde{\mathbf{H}}_\mu^{(n,m)} &= \delta_{n,m} \left(\widetilde{\mathbf{\Gamma}}_\mu^{(n,n)} \otimes \mathbf{I}_{I_n} \right) \\ &\quad + \left(\mathbf{I}_R \otimes \mathbf{A}^{(n)} \right) \widetilde{\mathbf{B}}_\mu^{(n,m)} \left(\mathbf{I}_R \otimes \mathbf{A}^{(m)\,T} \right) \end{aligned} \tag{28}$$

$$\widetilde{\mathbf{\Gamma}}_\mu^{(n,n)} = (\mathbf{\Gamma}_{nn} + \mu \mathbf{I}_R)^{-1} \tag{29}$$

and $\widetilde{\mathbf{B}}_\mu^{(n,m)}$ is the $(n,m)-$th block of $\mathbf{B}_\mu^{-1}$, in particular

$$\widetilde{\mathbf{B}}_\mu^{(n,m)} = \delta_{n,m} \mathbf{J}_n^{-1} + \mathbf{L}_n \mathbf{S}_\mu^{-1} \mathbf{L}_m \tag{30}$$

$$\begin{aligned}\mathbf{L}_n &= \mathbf{J}_n^{-1} \mathbf{K}_n^{-1} \mathbf{D}_n \\ &= -\mathbf{J}_n^{-1} \mathbf{P}_R \operatorname{dvec}(1 \oslash \mathbf{\Gamma}_{nn}) \end{aligned} \tag{31}$$

for $m, n = 1, \ldots, N$. Note that the inversion of $\mathbf{H}_\mu$ in the block form is in memory saving format. It requires only saving the matrices $\widetilde{\mathbf{\Gamma}}_\mu^{(n,n)}$, $\mathbf{\Gamma}_{nn}$, $\mathbf{J}_n^{-1}$, $n = 1, \ldots, N$, and $\mathbf{S}_\mu^{-1}$.

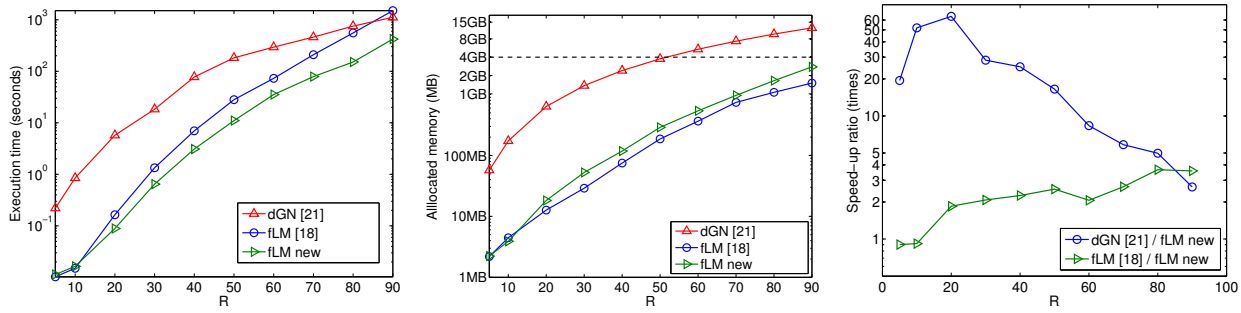
Remark. Unlike the fast inversion of the Hessian through (10) with direct inversion of $\mathbf{B}_\mu$, the novel fast inversion through (24) or (27)-(31) requires the assumption that no element of the matrices $\mathbf{\Gamma}_{nn}$ is zero. It means that the latter method is numerically less stable than the former method, when the factor matrices have orthogonal or nearly orthogonal columns. In that case, the nearly zero elements of $\mathbf{\Gamma}_{nn}$ have to be replaced by a suitable small constant. If the Hessian remains positive definite, the convergence of the whole dGN method is not affected at all or is affected only a little. In other words, the convergence can be a little slower, but it is guaranteed, to the same terminating point with zero gradient. Note that for tensors coming from the real-world it is rare to get factors with nearly orthogonal columns. A more problematic case (and also more frequent) is to have nearly co-linear columns.

4. EXAMPLES

The Levenberg-Marquart algorithm has been proved to outperform existing other CP algorithms [17, 18], and to be particularly efficient in some difficult CP decompositions such as decomposition of tensors with (nearly) linear dependency among components of factor matrices, or large difference in magnitude between components [20, 17, 18]. Hence, in this section, we only compare execution times of the fLM algorithms including the standard dGN algorithm in the Matlab routines PARAFAC3W [17, 21], the fLM algorithm in [18] and the new fLM algorithm. The standard dGN algorithm in PARAFAC3W [21] employs Cholesky decomposition and back substitution to solve the inverse problems $\mathbf{H}^{-1}\mathbf{g}$, where $\mathbf{g}$ is the CP gradient (MTTKRP). Since PARAFAC3W supports only order-3 tensors, the dGN algorithm does not participate in simulations for higher order tensors. The computational costs to inverse the approximate Hessian $\mathbf{H}$ of the new fLM, fLM[18] and dGN[17] are $\mathcal{O}(NR^6)$, $\mathcal{O}(N^3R^6)$ and $\mathcal{O}(R^3T^3)$, respectively.

We generated order-3 and order-4 dense random tensors of size $I_n = 100$ for all n, whose entries are normally distributed random numbers. The random tensors were decomposed into R rank-one tensors with R = 5, 10 , 20, ..., 90. Algorithms factorized the same data tensors using the same initialization values. There was not any stopping criterion applied to the three algorithms, except the maximum number of iterations was 100 for low R, and 20 for high $R \geq 60$. Execution time for each algorithm was measured using the stopwatch command: "tic" "toc" of MATLAB release 2011a on a laptop which had 1.8 GHz i7 processor and 4 GB memory.

Fig. 2 shows the average execution times per iteration of the three LM algorithms. The fLM and new fLM algorithms are significantly faster than dGN. For example, dGN took 78 seconds/iteration on average to factorize a $100 \times 100 \times 100$ dimensional tensor with $R = 40$, while fLM[18] took 7 seconds/iteration, and new fLM need only 3.1 seconds/iteration. It means that for this test case, the new fLM was approximately 2.3 and 25 times faster than fLM[18] and dGN[17], respectively. The speed-up ratios between dGN and new fLM



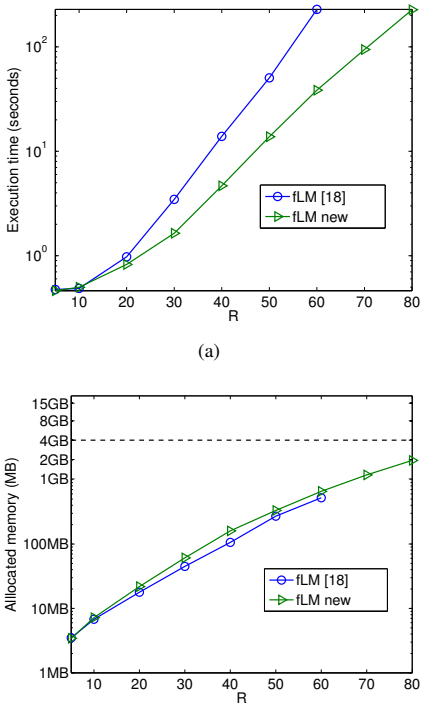
(a) (b) (c)

Fig. 1. Comparison of memory requirements and execution time per iteration of fLM algorithms and their speed-up ratio in approximation of $100\times100\times100$ dimensional tensors by rank-R tensors where $R = 5, 10, 20, \ldots, 90$.

were approximately 19.4, 51.9, 64.3, 28.4, 25.1, 16.5, 8.3, 5.8, 5 and 2.6 times for $R = 5$, 10, 20, ..., 90, respectively as seen in Fig. 1(c). The ratio was relatively high (> 30 times) for low ranks $R \leq 30$, and gradually reduced to 2 times when R increased and approached $I_n = 100$. Nevertheless, the new fLM algorithm saved at least 12 minutes for each iteration compared with the dGN algorithm when $R = 90$.

In addition to having low computational cost, the new fLM algorithm was also considerably less memory consuming than the dGN algorithm[21]. The amounts of allocated memory per iteration of the three fLM algorithms are compared in Fig. 1(b). It would be important to note that the machine had only 4 GB RAM, and the algorithm cannot use the whole memory. The standard dGN algorithm was extremely space consuming. When $R \geq 50$, this algorithm exceeded the memory limit; hence it became relatively slow. The new fLM algorithm required much less RAM than dGN.

Comparison between fLM and new fLM in decomposition of order-4 tensors of size $I_n = 100$ is illustrated in Fig. 2. The new fLM algorithm was approximately 6 times faster than fLM[18] when the two algorithms decomposed the tensors into $R = 60$ rank-one tensors.



(a)

(b)

Fig. 2. Comparison of execution time per iteration and of memory requirements of the fLM algorithms and their speed-up ratio in approximation of $100\times100\times100\times100$ dimensional tensors by rank-R tensors where $R = 5, 10, 20, \ldots, 80$.

5. CONCLUSIONS

We have derived explicit forms of inversion of the Hessian matrix of the multilinear mapping that describes the CP factorization, suitable for application in the fast damped Gauss-Newton method. The inversion is both fast and memory saving. In future we wish to extend the method to some new emerging models such as Kronecker product tensor decomposition and block tensor decomposition.

Appendix A: Matrix Inversion Lemma

Let $\mathbf{A}$, $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{R}$ are matrices of compatible dimensions such that the following products and inverses exist.

Then

$$(\mathbf{A} + \mathbf{XRY})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{X}(\mathbf{R}^{-1} + \mathbf{YA}^{-1}\mathbf{X})^{-1}\mathbf{YA}^{-1} \tag{32}$$

6. REFERENCES

[1] C.M. Andersson and R. Bro, "Practical aspects of PARAFAC modelling of fluorescence excitation-emission data," *Journal of Chemometrics*, vol. 17, pp. 200–215, 2003.

[2] R. Bro, *Multi-way Analysis in the Food Industry - Models, Algorithms, and Applications*, Ph.D. thesis, University of Amsterdam, Holland, 1998.

[3] G. Tomasi, *Practical and Computational Aspects in Chemometric Data Analysis*, Ph.D. thesis, Frederiksberg, Denmark, 2006.

[4] L. De Lathauwer and J. Castaing, "Tensor-based techniques for the blind separation of DS-CDMA signals," *Signal Processing*, vol. 87, no. 2, pp. 322–336, feb 2007.

[5] N.D. Sidiropoulos and R. Bro, "PARAFAC techniques for signal separation," in *Signal Processing Advances in Communications*, P. Stoica, G. Giannakis, Y. Hua, and L. Tong, Eds., vol. 2, chapter 4. Prentice-Hall, Upper Saddle River, NJ, USA, 2000.

[6] M. Mørup, L. K. Hansen, C. S. Herrmann, J. Parnas, and S. M. Arnfred, "Parallel factor analysis as an exploratory tool for wavelet transformed event-related EEG," *NeuroImage*, vol. 29, no. 3, pp. 938–947, 2006.

[7] H. Becker, P. Comon, L. Albera, M. Haardt, and I. Merlet, "Multi-way space-time-wave-vector analysis for EEG source separation," *Signal Processing*, vol. 92, no. 4, pp. 1021–1031, 2012.

[8] B. W. Bader, M. W. Berry, and M. Browne, "Discussion tracking in Enron email using PARAFAC," in *Survey of Text Mining II*, M. W. Berry and M. Castellanos, Eds., pp. 147–163. Springer London, 2008.

[9] D. M. Dunlavy, T. G. Kolda, and E. Acar, "Temporal link prediction using matrix and tensor factorizations," *ACM Transactions on Knowledge Discovery from Data*, vol. 5, no. 2, pp. Article 10, 27 pages, February 2011.

[10] D. González, A. Ammar, F. Chinesta, and E. Cueto, "Recent advances on the use of separated representations," *International Journal for Numerical Methods in Engineering*, vol. 81, no. 5, pp. 637–659, 2010.

[11] A. Shashua and T. Hazan, "Non-negative tensor factorization with applications to statistics and computer vision," in *Proc. of the 22-th International Conference on Machine Learning (ICML)*, Bonn, Germany, 2005, pp. 792–799, ICML.

[12] I. Ibraghimov, "Application of the three-way decomposition for matrix compression," *Numerical Linear Algebra with Applications*, vol. 9, no. 6-7, pp. 551–565, 2002.

[13] T.G. Kolda and B.W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, September 2009.

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

[15] C. Hayashi and F. Hayashi, "A new algorithm to solve PARAFAC-model," *Behaviormetrika*, vol. 11, pp. 49–60, 1982.

[16] P. Paatero, "Least-squares formulation of robust non-negative factor analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 37, pp. 23–35, 1997.

[17] G. Tomasi and R. Bro, "A comparison of algorithms for fitting the PARAFAC model", *Computational Statistics and Data Analysis*, vol. 50, pp. 1700-1734, 2006.

[18] A.-H. Phan, P. Tichavský, and A. Cichocki, "Low complexity damped Gauss-Newton algorithms for CANDECOMP /PARAFAC," http://arxiv.org/abs/1205.2584, *SIAM Journal on Linear algebra and Application*, accepted for publication.

[19] B. C. Mitchell and D. S. Burdick, "Slowly converging PARAFAC sequences: Swamps and two-factor degeneracies," *Journal of Chemometrics*, vol. 8, pp. 155168, 1994.

[20] P. Paatero, "A weighted non-negative least squares algorithm for three-way 'PARAFAC' factor analysis", *Chemometrics and Intelligent Laboratory Systems*, vol. 38, pp. 223–242, 1997.

[21] G. Tomasi, *INDAFAC and PARAFAC3W*. `http://www.models.kvl/dk/source/indafac/index.asp`, 2003.

[22] G. Tomasi, "Recent developments in fast algorithms for fitting the PARAFAC model," Greece, 2006, TRICAP.

[23] P. Tichavský, Z. Koldovský, "Simultaneous search for all modes in multilinear models", *Proc. ICASSP 2010*, Dallas, TX, March 14-19, 2010, pp. 4114-4117.

[24] C. A. Anderson and R. Bro, "The n-way toolbox for MATLAB", *Chemometrics and Intelligent Laboratory Systems*, vol. 52, pp. 1-4, 2000.

[25] K. Madsen, H. B. Nielsen, O. Tingleff, "Methods for nonlinear least squares problems, second ed.", Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 2004.

[26] A.-H. Phan, P. Tichavský, and A. Cichocki, "On fast computation of gradients for CANDECOMP/PARAFAC algorithms," CoRR, vol. abs/1204.1586, 2012, submitted for publication.