Fast Decomposition of Large Nonnegative Tensors

Jérémy E Cohen, Rodrigo Cabral Farias, Pierre Comon

Fast Decomposition of Large Nonnegative Tensors

Jeremy Cohen*, Rodrigo Cabral Farias, Pierre Comon, *Fellow, IEEE*

Abstract

In Signal processing, tensor decompositions have gained in popularity this last decade. In the meantime, the volume of data to be processed has drastically increased. This calls for novel methods to handle Big Data tensors. Since most of these huge data are issued from physical measurements, which are intrinsically real nonnegative, being able to compress nonnegative tensors has become mandatory. Following recent works on HOSVD compression for Big Data, we detail solutions to decompose a nonnegative tensor into decomposable terms in a compressed domain.

Index Terms

Big Data, Compression, CP decomposition, HOSVD, Non-negative, Parafac, Tensor.

EDICS Category: SAS-ICAB SAM-SAMC

J. Cohen, R. Cabral Farias and P. Comon are with CNRS, at GIPSA-lab, France (e-mail: {jeremy.cohen; rodrigo.cabral-farias; pierre.comon}@gipsa-lab.fr). This work has been supported in part by ERC AdG-2013-320594 grant "DECODA".

Manuscript submitted on 26 Sept. 2014.

Fast Decomposition of Large Nonnegative Tensors

I. Introduction

In the era of Big Data, the ability to handle huge data sets has become a key challenge. This fact hits the tensor decomposition domain as much as any other field, as proven by recent papers on the subject [1]–[3]. However, in the case where tensors solely contain nonnegative data, little has been made to improve computational speed by using compression. Still, tensor major applications like fluorescence spectroscopy or image processing (*e.g.* hyperspectral) do induce such positiveness.

In this letter, the focus will be on decomposing tensors into a sum of rank-1 terms. Since [4], such a decomposition is referred to as CP, which now smartly stands either for "Canonical Polyadic" or for "Candecomp/Parafac" [5]. More precisely, our concern will be to find a compressed version of the nonnegative CP decomposition. The well-known HOSVD will be used as an unconstrained compression method, as was suggested in R. Bro's thesis [6, p. 92] before the concept was formally stated in [7]. After unconstrained compression, we will perform a constrained CP decomposition. To our knowledge, no constructive algorithm has been proposed to date, even though the concept had already been evoked in Bro's thesis [6, p. 149-150] but considered as being too difficult to implement. A constrained compression algorithm has been developed in [3], but the compression in this case is based on the low rank approximation of a positive matrix and this hypothesis is already used in all CP algorithms. The latter algorithm also extends to Tucker3, but is not well designed for CP as it would require a full unconstrained CP as a first step [8]. On the other hand, the solution provided in this letter is specifically designed for CP.

Our approach reduces drastically the dimensions of the problem, in a similar fashion as the recent proposition of random compression [9]. The use of the HOSVD however enables a straightforward formulation of the problem, and we make use of the unitary structure of the transformation matrices to ease a fast compression-decompression scheme.

The letter is organized as follows. First, we formalize the problem and derive an objective function and constraints. In the second part, some algorithms are detailed and their convergence is discussed. In the last section, we show the efficiency and gain in computation speed on synthetic data.

II. Problem statement

Our first contribution is to provide a clear theoretical support to the nonnegative compression scheme. We consider 3-way tensors, but the generalization to n-way tensors is straightforward. Let $\boldsymbol{\mathcal{T}}$ be a $K \times L \times M$ nonnegative tensor. The main idea is to work on a smaller tensor $\boldsymbol{\mathcal{G}}$, which contains almost the same information as the original tensor. Because $\boldsymbol{\mathcal{G}}$ is meant much smaller than $\boldsymbol{\mathcal{T}}$, performing a CP on $\boldsymbol{\mathcal{G}}$ will of course be a way to speed up

the global tensor decomposition. However, positiveness constraints apply to $\boldsymbol{\mathcal{T}}$ and this must be considered when writing out the optimization problem.

A. HOSVD compression

The compression of $\boldsymbol{\mathcal{T}}$ into $\boldsymbol{\mathcal{G}}$ can be performed by applying the following unconstrained approximate HOSVD:

$$\boldsymbol{\mathcal{T}} \approx (\mathbf{U}, \mathbf{V}, \mathbf{W})\boldsymbol{\mathcal{G}} = \left[\!\!\left[\sum_{pqr}^{R_1 R_2 R_3} \mathbf{U}_{kp}\mathbf{V}_{\ell q}\mathbf{W}_{mr}\boldsymbol{\mathcal{G}}_{pqr} \right]\!\!\right]_{k\ell m}, \tag{1}$$

where $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{W}$ are unitary matrices with a truncated number of columns R_1, R_2 and R_3 to reduce the size of $\boldsymbol{\mathcal{G}}$, as recommended in [7], [10]. A naive attempt would be to impose the positiveness of the unitary matrices, but nonnegative unitary matrices mainly reduce to scaled permutations, which are useless in the present context. Moreover, truncating the HOSVD is a well founded operation, for which algorithms are available. Since HOSVD is implemented in practice through SVD of the unfolded data, the algorithm that carries out the SVD computation must be efficient for very large data sets. Therefore, we suggest to resort to randomly initialized Lanczos algorithms [11].

In this approach, compression parameters R_i are chosen to be as small as possible, such that no (or little) information is lost. Observe that compression does not only allow to speed up the algorithm but also allows denoising.

B. Compressed Nonnegative CP

We consider the freely compressed core tensor $\boldsymbol{\mathcal{G}}$ as an input of a CP decomposition:

$$\boldsymbol{\mathcal{G}} = (\mathbf{A}_c, \mathbf{B}_c, \mathbf{C}_c)\,\boldsymbol{\Lambda}, \tag{2}$$

where $\mathbf{A}_c$, $\mathbf{B}_c$ and $\mathbf{C}_c$ are called "compressed factor matrices", and $\boldsymbol{\Lambda}$ is a cubical diagonal tensor of dimension R; R is called the tensor rank of $\boldsymbol{\mathcal{G}}$, which approximately equals that of $\boldsymbol{\mathcal{T}}$, up to discarded information supposably due to noise [10]. This parameter R can be either inferred from physical modeling or it can be seen as another compression parameter.

From (1) and (2), it is clear that the full decomposition of $\boldsymbol{\mathcal{T}}$ is:

$$\boldsymbol{\mathcal{T}} \approx (\mathbf{U}\mathbf{A}_c, \mathbf{V}\mathbf{B}_c, \mathbf{W}\mathbf{C}_c)\,\boldsymbol{\Lambda}, \tag{3}$$

so that the positiveness of $\boldsymbol{\mathcal{T}}$ imposes that the matrices below have nonnegative entries:

$$\mathbf{U}\mathbf{A}_c \succeq 0,\ \mathbf{V}\mathbf{B}_c \succeq 0,\ \mathbf{W}\mathbf{C}_c \succeq 0,$$

where $\succeq$ means element-wise inequality. Decomposition (3) is a compressed version of the usual (uncompressed) CP decomposition:

$$\boldsymbol{\mathcal{T}} \approx (\mathbf{A}, \mathbf{B}, \mathbf{C})\,\boldsymbol{\Lambda}, \tag{4}$$

where factors $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ are constrained to be nonnegative. This leads to the following non-convex objective function:

$$\begin{aligned} \text{minimize} \quad & \Upsilon = \|\boldsymbol{\mathcal{G}} - (\mathbf{A}_c, \mathbf{B}_c, \mathbf{C}_c)\, \boldsymbol{\Lambda}\|_2^2, \\ \text{w.r.t.} \quad & \mathbf{A}_c, \mathbf{B}_c, \mathbf{C}_c, \boldsymbol{\Lambda} \\ \text{subject to} \quad & \mathbf{U}\mathbf{A}_c \succeq 0,\ \mathbf{V}\mathbf{B}_c \succeq 0,\ \mathbf{W}\mathbf{C}_c \succeq 0. \end{aligned} \tag{5}$$

The tricky question of *existence* of a best rank-R approximate is solved in our case:

Proposition 1: The infimum of the objective Υ in (5) is always reached.

Proof: First, the best low multilinear rank approximation always exists as it is computed from low-rank matrix approximations [7]. Then the rest of the proof goes along the same lines as in [12]. In fact, on one hand the objective also writes $\|\boldsymbol{\mathcal{T}} - (\mathbf{U}\mathbf{A}_c, \mathbf{V}\mathbf{B}_c, \mathbf{W}\mathbf{C}_c)\, \boldsymbol{\Lambda}\|_2^2$ because the L^2 norm is invariant with respect to semi-unitary matrices, and matrices $\mathbf{U}\mathbf{A}_c$, $\mathbf{V}\mathbf{B}_c$ and $\mathbf{W}\mathbf{C}_c$ still belong to closed bounded subsets since columns of $\mathbf{A}_c$, $\mathbf{B}_c$ and $\mathbf{C}_c$ are normalized. And on the other hand the objective (5) is still coercive with respect to $\boldsymbol{\Lambda}$. ■

Now that the problem has been well-posed, we have to make sure it can be solved numerically. In the next section, we show how an ALS algorithm can be designed to answer the optimization problem. Next, a constrained descent algorithm is promoted in Section IV. Both perform well on synthetic data as shown in the last section.

III. Compressed and projected alternating least squares (CP-ALS)

The most widely used algorithm to solve (5) without constraints is the alternating least squares algorithm (ALS) [10]. At iterate k of ALS, two previous estimates of factor matrices, say $\hat{\mathbf{B}}_c^k$ and $\hat{\mathbf{C}}_c^k$ are fixed, and the objective $\Upsilon\left(\mathbf{A}_c, \hat{\mathbf{B}}_c^k, \hat{\mathbf{C}}_c^k\right)$ is minimized w.r.t. $\mathbf{A}_c$. This leads to a linear least squares problem with the following closed-form solution

$$\hat{\mathbf{A}}_c^{k+1} = \mathbf{G}_{(1)} \left(\hat{\mathbf{C}}_c^k \odot \hat{\mathbf{B}}_c^k\right)^{\dagger}, \tag{6}$$

$\mathbf{G}_{(1)}$ is the unfolding matrix of $\boldsymbol{\mathcal{G}}$ in the first way and $\mathbf{X}^{\dagger}$ denotes the pseudo-inverse of $\mathbf{X}$. Note that the scaling factor $\boldsymbol{\Lambda}$ has been pulled in the matrix factor $\mathbf{C}_c$, whereas matrices $\mathbf{B}_c$ and $\mathbf{A}_c$ still have normalized columns.

After solving for $\mathbf{A}_c$, the objective is minimized w.r.t. the other factor matrices, in turn. This alternating procedure is pursued until convergence. The popularity of this method does not come from its proved performances but rather from the fact that each of its step has an analytical solution that can be easily programmed.

A. Projected ALS for the uncompressed problem

In the original uncompressed space, projection onto the nonnegative orthant of the results given by each step of ALS allows to deal with the nonnegative tensor decomposition problem. For example, the update in the uncompressed space including a projection step is given by

$$\hat{\mathbf{A}}^{k+1} = \max\left[\mathbf{0}, \mathbf{T}_{(1)} \left(\mathbf{C}^k \odot \mathbf{B}^k\right)^{\dagger}\right], \tag{7}$$

where $\max[\mathbf{X}, \mathbf{Z}]$ is the maximum function applied entry-wise to $\mathbf{X}$ and $\mathbf{Z}$. This modified ALS algorithm is commonly known as projected ALS or alternating nonnegative least squares (ANLS) [2]. One question that arises is the following: can we apply a similar simple modification to ALS to find the solution to the nonnegative tensor decomposition problem in the compressed space?

B. Projected ALS for the compressed problem

Clearly this simple way of imposing nonnegativity within the ALS algorithm by clipping nonnegative values to zero cannot be directly applied to update (6) without care. Indeed, the nonnegative constraints need to apply in the original space and not in the compressed subspace. Should solution (7) be applied in the algorithm, then a simple solution to the following projection problem is needed:

$$\begin{array}{ll} \text{minimize} & \left\|\mathbf{A}_{cp} - \hat{\mathbf{A}}_c^{k+1}\right\|_2^2, \text{ w.r.t. } \mathbf{A}_{cp}, \\ \text{subject to} & \mathbf{U}\mathbf{A}_{cp} \succeq 0, \end{array} \tag{8}$$

$\mathbf{A}_{cp}$ stands for the factor matrix obtained after decompression-projection-re-compression. Now, the solution is harder to find because $\mathbf{U}\mathbf{U}^\top$ is an orthogonal projector onto a smaller dimensional subspace, hence $\mathbf{U}$ not invertible. The only way to solve it exactly is to resort to iterative optimization methods. Since this problem is convex, iterative algorithms could be used to find the projection. However, the large number of constraints makes it deterrent, bearing in mind that (8) is just one iteration of an ALS algorithm, and should not be too computationally heavy.

Due to the difficulties explained above, we propose a simple and approximate solution to (8), which is given in 3 steps (notation for factor $\mathbf{A}_c$ is used as an example and expressions are similar for the two other factors)

$$\begin{array}{lrcl} \textbf{1:} & \hat{\mathbf{A}}^{k+1} & = & \mathbf{U}\hat{\mathbf{A}}_c^{k+1}, \\ \textbf{2:} & \left[\hat{\mathbf{A}}^{k+1}\right]^+ & = & \max\left[\mathbf{0}, \hat{\mathbf{A}}^{k+1}\right], \\ \textbf{3:} & \hat{\mathbf{A}}_{cp}^{k+1} & = & \mathbf{U}^\top\left[\hat{\mathbf{A}}^{k+1}\right]^+. \end{array} \tag{9}$$

The first step corresponds to decompression, the second step forces the uncompressed factor to be nonnegative, and the third step is re-compression; note that $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$. The standard ALS in the compressed space with the additional steps (9) will be referred to as "compressed and projected ALS" (CP-ALS).

a) Nonincreasing cost function: note that this procedure is not the exact solution to (8). If we go back to the uncompressed space with $\hat{\mathbf{A}}^{k+1} = \mathbf{U}\hat{\mathbf{A}}_c^{k+1}$ after this approximate projection, we get

$$\mathbf{U}\mathbf{U}^\top\left[\hat{\mathbf{A}}^{k+1}\right]^+, \tag{10}$$

which has no reason to be nonnegative because $\mathbf{U}\mathbf{U}^\top \neq \mathbf{I}$. Yet, the approximate projection decreases the error on the compressed factor, as stated by the proposition below.

Lemma 1: Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two convex closed sets of $\mathbb{R}^N$, with a non empty intersection, $\boldsymbol{x_o}$ be a vector in $\mathcal{S}_1 \cap \mathcal{S}_2$, and p_i denote the projector onto $\mathcal{S}_i$. Then we have, $\forall \boldsymbol{x} \in \mathbb{R}^N$:

$$\|p_2 \circ p_1\boldsymbol{x} - \boldsymbol{x}_o\| \leq \|\boldsymbol{x} - \boldsymbol{x}_o\|,$$

where $\|\cdot\|$ denotes the euclidian norm.

Proof: The proof is straightforward: $\|p_2 \circ p_1\boldsymbol{x} - \boldsymbol{x}_o\| = \|p_2 \circ p_1\boldsymbol{x} - p_2 \circ p_1\boldsymbol{x}_o\|$ because $\boldsymbol{x}_o \in \mathcal{S}_1 \cap \mathcal{S}_2$, and since p_i are contracting $\|p_2 \circ p_1\boldsymbol{x} - p_2 \circ p_1\boldsymbol{x}_o\| \leq \|\boldsymbol{x} - \boldsymbol{x}_o\|$. ■

Proposition 2: In the compression-decompression iteration described above, it holds that

$$\left\|\mathbf{U}^\top\left[\hat{\mathbf{A}}^{k+1}\right]^+ - \mathbf{A}_c\right\|_F^2 \leq \left\|\hat{\mathbf{A}}_c^{k+1} - \mathbf{A}_c\right\|_F^2. \tag{11}$$

Proof: Apply the lemma with $\mathcal{S}_1$ being the nonnegative orthant of $\mathbb{R}^N$, $N = K \times R$, and with $\mathcal{S}_2 = span\{\mathbf{U}\} \otimes \mathbb{R}^R$ being the subspace spanned by the R_1 columns of matrix $\mathbf{U}$. In that case, $p_2 = \mathbf{U}\mathbf{U}^T$. Then for any nonnegative matrix $\mathbf{X}_o$ of $\mathcal{S}_1 \cap \mathcal{S}_2$, and for any matrix $\mathbf{X}$ of $\mathbb{R}^{K \times R}$, we have:

$$\|\mathbf{U}\mathbf{U}^T p_1 \mathbf{X} - \mathbf{X}_o\|_F \leq \|\mathbf{X} - \mathbf{X}_o\|_F.$$

Yet, as any element of $\mathcal{S}_2$, $\mathbf{X}_o$ can be written as $\mathbf{X}_o = \mathbf{U}\mathbf{M}_o$, where $\mathbf{M}_o$ is a matrix of size $R_1 \times R$. Now apply this result to a general element of $\mathcal{S}_2$, $\mathbf{X} = \mathbf{U}\mathbf{M}_c$. We get the inequality:

$$\|\mathbf{U}^T p_1 \mathbf{U}\mathbf{M}_c - \mathbf{M}_o\|_F \leq \|\mathbf{M}_c - \mathbf{M}_o\|_F, \tag{12}$$

which holds true because $\mathbf{U}$ is an isometry, that is, because $\|\mathbf{U}\boldsymbol{y}\| = \|\boldsymbol{y}\|$. ■

b) Link with alternating direction method of multipliers: There is an interesting connection between the approximate projection method proposed above and the alternating direct method of multipliers (ADMM). ADMM has been recently popularized through its use in distributed estimation and optimization problems. The ADMM form of the solution to (8) is (see [13] for details on ADMM)

$$\begin{aligned}
\textbf{1:} \quad \hat{\mathbf{A}}_c^{i+1} &= \tfrac{\hat{\mathbf{A}}_c}{1+\rho} + \tfrac{\rho}{1+\rho}\mathbf{U}^\top \left(\hat{\mathbf{A}}_i - \mathbf{E}_{\mathbf{A}}^i\right), \\
\textbf{2:} \quad \hat{\mathbf{A}}^{i+1} &= \max\left(\mathbf{0}, \mathbf{U}\hat{\mathbf{A}}_c^{i+1} + \mathbf{E}_{\mathbf{A}}^i\right), \\
\textbf{3:} \quad \mathbf{E}_{\mathbf{A}}^{i+1} &= \mathbf{E}_{\mathbf{A}}^i + \mathbf{U}\hat{\mathbf{A}}_c^{i+1} - \hat{\mathbf{A}}^{i+1},
\end{aligned} \tag{13}$$

where $\hat{\mathbf{A}}_c$ is the point to be projected (the estimate given by the unconstrained ALS update), ρ is a penalty parameter and $\mathbf{E}_{\mathbf{A}}^k$ is a matrix of scaled dual variables. If we choose the previous estimate of the uncompressed factor $\hat{\mathbf{A}}_0$ to be $\mathbf{U}\hat{\mathbf{A}}_c$ and we set $\mathbf{E}_{\mathbf{A}}^0 = \mathbf{0}$, then 2 iterations of ADMM with $\rho = 1$ are equal to the approximate projection (9).

In the next section, another algorithm is described to solve (5), based on a conjugate gradient descent.

IV. Soft Penalization gradient-based algorithm

Although simple to implement, nothing guarantees that the number of iterates needed for the convergence of an alternating minimization approach, CP-ALS for example, will not make it slower than an all-at-once optimization method, which updates all factors at the same time. On the other hand, gradient-based all-at-once methods are widely used to compute the CP decomposition of data tensor [10], [14]. The Fletcher-Reeves non-linear conjugate gradient is especially efficient and makes convergence proofs possible for a backtracking step size satisfying some conditions [15] (which will be detailed later).

There exist in the literature various ways to ensure non-negativity of the factors in gradient-based approaches. Some authors consider the constraint explicitly by using barrier functions [14], dual formulations, or projection onto the positive orthant at each gradient iterate [16]. There are other techniques which are based on imposing nonnegativity through multiplicative updates [17] or re-parameterizations [18].

A. Soft Penalization

Unlike a barrier function which renders negative factors impossible, we chose a soft penalization function. This choice seems somewhat arbitrary considering the wide panel of possibilities. However, this choice presents some

advantages: (i) if the trajectory is allowed to cross non admissible regions, convergence may be faster, (ii) it is easy to implement in large dimensions.

In the following, we use a general sigmoid function f_α, but other functions may be used such as the hyperbolic tangent or the arc-tangent functions. It takes the form

$$f_\alpha(x) = \delta\left(1 - \frac{1}{1+e^{-(\alpha\sqrt{d})x}}\right) \tag{14}$$

where δ is the penalization amplitude, α is the stiffness and d is the dimension of the penalized subspace. The rescaling by $\sqrt{d}$ is meant for normalization. Parameters d and α may be different in each mode, but we use only one global amplitude parameter δ to control the weight of penalization; in particular, it can be decreased when approaching convergence, if constraints are inactive. Some numerical observations with proper initializations tend to suggest that the constraints were indeed not active most of the time, so that for simulations, the penalization was rarely needed.

B. Conjugate gradient computation

As explained previously, we use the non-linear conjugate gradient as a descent method, subsequently called "compressed conjugate gradient" (CCG). Some implementation details are now given.

1) Normalization choices: because the Big Data setting induces important variations in the dimensions of the input tensor, we need to build an algorithm that is robust to these variations. That is, we want the tuning of parameters of the algorithm to depend as little as possible on input dimensions. This requires a careful normalization of the data when computing the positive compressed CP. Therefore, we normalize the stiffness and amplitude in the penalization.

2) Objective function: taking in consideration all previous observations, we get the following objective function to minimize:

$$S(\boldsymbol{\theta}) = \Upsilon(\boldsymbol{\theta}) + \frac{1}{R(K+L+M)}\left(\sum_{ij} f_{\alpha_1}\left([\mathbf{U}\hat{\mathbf{A}}_c]_{ij}\right) + \dots\right)$$

where $\boldsymbol{\theta}$ is a $R_1R_2R_3R\times 1$ parameter vector containing the entries of $\hat{\mathbf{A}}_c$, $\hat{\mathbf{B}}_c$ and $\hat{\mathbf{C}}_c$, that is $\boldsymbol{\theta} = \mathbf{vec}\left(\hat{\mathbf{A}}_c^T, \hat{\mathbf{B}}_c^T, \hat{\mathbf{C}}_c^T\right)$.

3) Gradient computation: the computation of the gradient of $S(\boldsymbol{\theta})$ does not raise any difficulty. If needed, compact expressions can be obtained with a similar approach as in [18].

C. CCG update rule

The CCG algorithm iterates as follows:

$$\begin{array}{lrcl} \textbf{1:} & \mathbf{p}^k & = & -\nabla S(\hat{\boldsymbol{\theta}}^k), \\ \textbf{2:} & \beta^k & = & \frac{\|\mathbf{p}^k\|_2^2}{\|\mathbf{p}^{k-1}\|_2^2}, \\ \textbf{3:} & \mathbf{s}^k & = & \mathbf{p}^k + \beta^k\mathbf{s}^{k-1}, \\ \textbf{4:} & \hat{\boldsymbol{\theta}}^{k+1} & = & \hat{\boldsymbol{\theta}}^k + \mu_k\mathbf{s}^k. \end{array} \tag{15}$$

The step size μ_k is chosen at the end of each iteration using Armijo's backtracking method. Under this backtracking strategy two conditions are sufficient to prove convergence to a local minimum of the conjugate gradient method

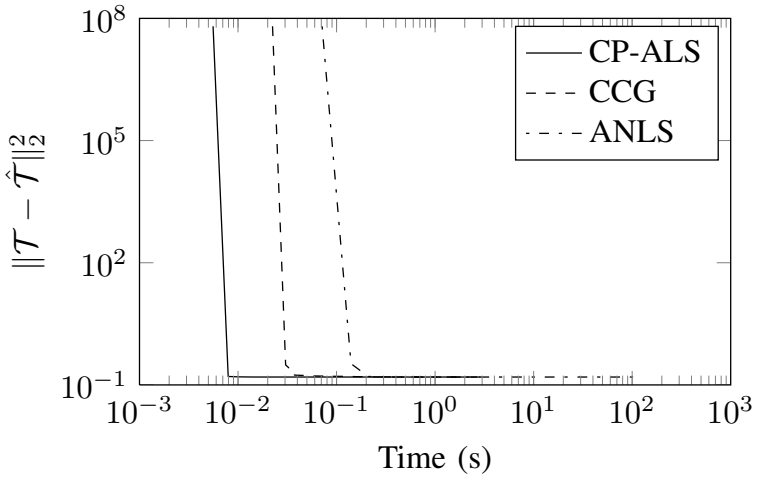

Fig. 1. Reconstruction error by computation speed of CP decomposition for $\mathcal{T}$ in $\mathbb{R}_+^{250\times250\times250}$. Compressed space is of size $20 \times 20 \times 20$.

[15, p. 49]: the objective function must be Lipschitz continuous and the second Wolfe condition on curvature must be satisfied (since the first is already satisfied by Armijo's rule). The gradient of our objective function clearly satisfies the Lipschitz condition because the CP objective (5) is a polynomial in the factors. Thus, by choosing a step size that satisfies the curvature condition, convergence of the CCG to a local minimum is guaranteed.

Computational complexities of CCG and CPALS are of same order $\mathcal{O}(R_1R_2R_3R)$. We thus expect both algorithms to be of similar running time. The main advantage of CCG is that it is less sensitive to swamps encountered with ALS, which have been extensively studied by the community [19], [20]. The next section compares the two algorithms in terms of speed and performance on different data sets.

V. Results on simulated data

For the following simulations, tensor $\mathcal{T}$ is generated randomly by drawing coefficients from a centered normalized Gaussian distribution and taking their absolute value. White Gaussian noise is added with standard deviation $\sigma = 10^{-4}$. The tensor rank R of $\mathcal{T}$ is set to 6. Fletcher-Reeves and Pollack-Ribiere implementations of the CCG yield similar results.

We first check the computation speed of CCG and CP-ALS compared with a ANLS without compression. One typical realization of the nonnegative tensor decomposition is given in Fig. 1. It appears that CP-ALS is faster than CCG, even though it will fail in some rare cases where CCG does not. Both proposed algorithms are faster than ANLS as the tensor dimensions increase. Note that the computation of the SVD through the randomized method [11] for $\mathcal{T}$ costs as much as a few steps of ANLS, and compression becomes worthwhile when the factorization of $\mathcal{T}$ is to be repeated.

Next, the compression error is reported in Fig. 2 for various compression rates, and is averaged over 20 trials. Because important information in $\mathcal{T}$ is contained in $R = 6$ rank-1 factors, compressing the data further than $(6, 6, 6)$ results in reconstruction error.

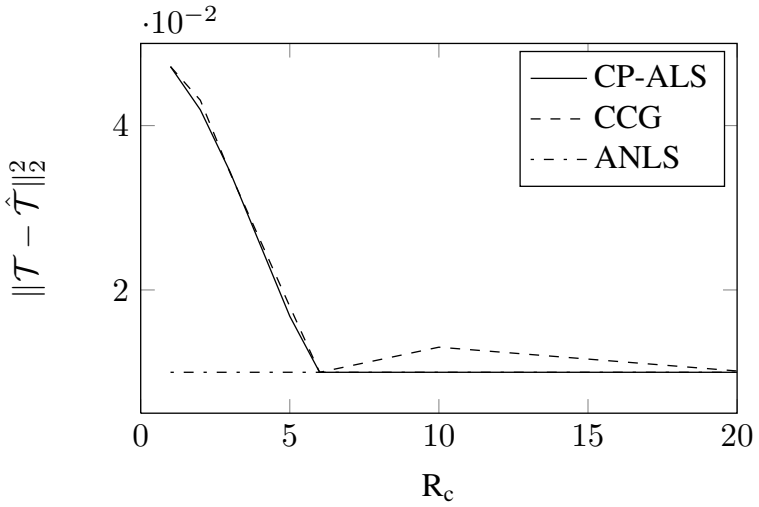

Fig. 2. Reconstruction error as a function of compressed dimensions $\boldsymbol{\mathcal{T}}$ in $\mathbb{R}_+^{100\times100\times100}$. Compressed space is of size $R_c \times R_c \times R_c$.

VI. Conclusion

Up to now, nonnegative tensors have not been decomposed in a compressed domain because the HOSVD cannot handle nonnegativity. In this letter, we suggest to solve this problem as a low dimensional constrained tensor decomposition. Reproducible algorithms derived from workhorse methods are proposed and simulation results show, without any substantial increase in the tensor reconstruction error, that such an approach allows a major speed up in the computation of the decomposition. This letter serves the purpose of introducing compression in constrained multilinear data; other types of constraints or compressions could be handled in a similar way, and give a full picture of the possibilities to deal with large tensor datasets.

Acknowledgment

We would like to thank M.-A. Veganzones for his helpful proofreading.

References

[1] N. D. Sidiropoulos, R. Bro, and G. B. Giannakis, "Parallel factor analysis in sensor array processing," *IEEE Trans. Sig. Proc.*, vol. 48, no. 8, pp. 2377–2388, Aug. 2000.

[2] A. Cichocki, R. Zdunek, A. H. Phan, and S.-I. Amari, *Nonnegative Matrix and Tensor Factorization*. Wiley, 2009.

[3] G. Zhou, A. Cichocki, Q. Zhao, and S. Xie, "Nonnegative matrix and tensor factorizations : An algorithmic perspective," *IEEE Sig. Proc. Magazine*, vol. 31, no. 3, pp. 54–65, 2014.

[4] H. A. L. Kiers, "Towards a standardized notation and terminology in multiway analysis," *Jour. Chemometrics*, pp. 105–122, 2000.

[5] P. Comon, "Tensors: a brief introduction," *IEEE Sig. Proc. Magazine*, vol. 31, no. 3, pp. 44–53, May 2014, hal-00923279.

[6] R. Bro, "Multi-way analysis in the food industry : Models, algorithms, and applications," Ph.D. dissertation, University of Amsterdam, Amsterdam, 1998.

[7] L. D. Lathauwer, B. D. Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM Jour. Matrix Ana. Appl.*, vol. 21, no. 4, pp. 1253–1278, Apr. 2000.

[8] G. Zhou, A. Cichocki, and S. Xie, "Fast nonnegative matrix/tensor factorization based on low-rank approximation," *IEEE Trans. Sig. Proc.*, vol. 60, no. 6, pp. 2928–2940, 2012.

[9] N. Sidiropoulos, E. E. Papalexakis, and C. Faloutsos, "Parallel randomly compressed cubes," *IEEE Sig. Proc. Magazine*, vol. 31, no. 5, pp. 57–70, Sep. 2014, special issue on Big data.

[10] P. Comon, X. Luciani, and A. L. F. De Almeida, "Tensor decompositions, alternating least squares and other tales," *Jour. Chemometrics*, vol. 23, no. 7-8, pp. 393–405, Aug. 2009, hal-00410057.

[11] N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Review*, vol. 53, no. 2, pp. 217–288, 2011.

[12] L.-H. Lim and P. Comon, "Nonnegative approximations of nonnegative tensors," *Jour. Chemometrics*, vol. 23, pp. 432–441, Aug. 2009.

[13] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

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

[15] C. T. Kelley, *Iterative methods for optimization*. Siam, 1999, vol. 18.

[16] C.-J. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural computation*, vol. 19, no. 10, pp. 2756–2779, 2007.

[17] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in neural information processing systems*, 2001, pp. 556–562.

[18] J.-P. Royer, N. Thirion-Moreau, and P. Comon, "Computing the polyadic decomposition of nonnegative third order tensors," *Signal Processing*, vol. 91, no. 9, pp. 2159–2171, Sep. 2011, hal-00618729.

[19] P. Paatero, "The multilinear engine: A table-driven, least squares program for solving multilinear problems, including the n-way parallel factor analysis model," *Journal of Computational and Graphical Statistics*, vol. 8, no. 4, pp. 854–888, Dec. 1999.

[20] M. Rajih, P. Comon, and R. Harshman, "Enhanced line search : A novel method to accelerate PARAFAC," *SIAM Jour. Matrix Ana. Appl.*, vol. 30, no. 3, pp. 1148–1171, Sep. 2008.