PARALLEL ALGORITHMS FOR LARGE SCALE CONSTRAINED TENSOR DECOMPOSITION

Athanasios P. Liavas [*]

Department of ECE
Technical University of Crete
liavas@telecom.tuc.gr

Nicholas D. Sidiropoulos [†]

Department of ECE
University of Minnesota
nikos@ece.umn.edu

ABSTRACT

Most tensor decomposition algorithms were developed for in-memory computation on a single machine. There are a few recent exceptions that were designed for parallel and distributed computation, but these cannot easily incorporate practically important constraints, such as nonnegativity. A new constrained tensor factorization framework is proposed in this paper, building upon the Alternating Direction method of Multipliers (ADMoM). It is shown that this simplifies computations, bypassing the need to solve constrained optimization problems in each iteration, yielding algorithms that are naturally amenable to parallel implementation. The methodology is exemplified using nonnegativity as a baseline constraint, but the proposed framework can incorporate many other types of constraints. Numerical experiments are encouraging, indicating that ADMoM-based nonnegative tensor factorization (NTF) has high potential as an alternative to state-of-the-art approaches.

Index Terms— Tensors, constrained optimization, parallel algorithms, nonnegative factorization, CANDECOMP, PARAFAC.

1. INTRODUCTION

Tensor decomposition has proven useful in a wide range of applications in signal processing and machine learning; see, for example, [1, 2, 3, 4, 5, 6, 7]. There exist two basic tensor factorization models: *parallel factor analysis* (PARAFAC) [8, 9] also known as *canonical decomposition* (CANDECOMP) [10], or CP (and CPD) for CANDECOMP-PARAFAC (Decomposition), or *canonical polyadic decomposition* (CPD, again); and the Tucker3 model. With few recent exceptions, all tensor factorization algorithms were originally developed for centralized, in-memory computation on a single machine. This model of computation is inadequate for emerging big data-enabled applications where the tensors to be analyzed cannot be loaded on a single machine, the data is more likely to reside in cloud storage, and cloud computing, or some other kind of high performance parallel architecture, must be used for the actual computation.

A carefully optimized Hadoop/MapReduce [11, 12] implementation of the basic ALS CP-decomposition algorithm was developed in [13], which reported 100-fold scaling improvements relative to the prior art. However, [13] is not designed for high performance computing architectures, and it does not incorporate constraints on the factor matrices. A random sampling approach has been proposed in [5], which creates and analyzes multiple randomly sub-sampled parts of the tensor, then combines the results using a common piece of data to anchor the constituent decompositions. The downside of [5] is that it only works for sparse tensors, and it offers no identifiability guarantees - although it usually works well for sparse tensors. A different approach based on generalized random sampling was recently proposed in [14, 15]. The idea is to create multiple randomly compressed mixtures (instead of sub-sampled parts) of the original tensor, analyze them all in parallel, and then combine the results. The main advantages of [14, 15] over [5] are that identifiability can be guaranteed, no sparsity is needed, and there are theoretical scalability guarantees. Distributed CP decomposition based on the ALS algorithm has been considered in [16, 17], which exploit the inherent parallelism in the matrix version of the linear least squares subproblems to split the computation in different ways, assuming an essentially 'flat' architecture for the computing nodes.

In this work, we develop efficient algorithms for constrained tensor factorization based on the Alternating Direction method of Multipliers (ADMoM). We focus on non-negative CP decomposition (sometimes referred to as non-negative tensor factorization) as a working problem, due to the importance of the CP model and nonnegativity constraints; but our approach can be readily generalized to many other types of constraints on the latent factors, as well as other tensor factorizations, such as Tucker3 (for a more detailed presentation of this work, see [18]).

The advantages of our approach are as follows. First, during each ADMoM iteration, we avoid the solution of constrained optimization problems, resulting in considerably smaller computational complexity per iteration compared to constrained least squares based algorithms, such as alternating non-negative least squares (NALS). Second, our approach leads naturally to distributed algorithms suitable for parallel implementation. Finally, it can easily incorporate many other types of constraints on the latent factors, such as sparsity. Numerical experiments are encouraging, indicating that the ADMoM-based NTF has high potential as an alternative to state-of-the-art approaches.

Notation: Vectors, matrices, and tensors are denoted by small, capital, and underlined capital bold letters, respectively; for example, $\mathbf{x}$, $\mathbf{X}$, and $\underline{\mathbf{X}}$. $\mathbb{R}_+^{I\times J\times K}$ denotes the set of $(I \times J \times K)$ real nonnegative tensors, while $\mathbb{R}_+^{I\times J}$ denotes the set of $(I \times J)$ real nonnegative matrices. $\|\cdot\|_F$ denotes the Frobenius norm of the tensor or matrix argument and $(\mathbf{A})_+$ denotes the projection of matrix $\mathbf{A}$ onto the set of element-wise nonnegative matrices. The outer product of vectors $\mathbf{a} \in \mathbb{R}^{I\times 1}$, $\mathbf{b} \in \mathbb{R}^{J\times 1}$, and $\mathbf{c} \in \mathbb{R}^{K\times 1}$ is the rank-one tensor $\mathbf{a} \circ \mathbf{b} \circ \mathbf{c} \in \mathbb{R}^{I\times J\times K}$ with elements $(\mathbf{a} \circ \mathbf{b} \circ \mathbf{c})(i, j, k) = \mathbf{a}(i)\mathbf{b}(j)\mathbf{c}(k)$. $\mathbf{A} \odot \mathbf{B}$ denotes the Khatri-Rao product. Finally,

[*]Supported in part by the EU and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: THALIS-UOA-SWINCOM.

[†]Supported in part by NSF IIS-1247632.

for matrices $\mathbf{A}$ and $\mathbf{B}$ with equal dimensions, we define $\mathbf{A} * \mathbf{B} := \sum_{i,j} \mathbf{A}_{i,j}\mathbf{B}_{i,j}$.

2. NONNEGATIVE TENSOR FACTORIZATION

Let tensor $\underline{\mathbf{X}}^o \in \mathbb{R}_+^{I\times J\times K}$ admit a nonnegative[1] CP decomposition of order F, $\underline{\mathbf{X}}^o = [\mathbf{A}^o, \mathbf{B}^o, \mathbf{C}^o] = \sum_{f=1}^F \mathbf{a}_f^o \circ \mathbf{b}_f^o \circ \mathbf{c}_f^o$, where $\mathbf{A}^o = [\mathbf{a}_1^o \cdots \mathbf{a}_F^o] \in \mathbb{R}_+^{I\times F}$, $\mathbf{B}^o = [\mathbf{b}_1^o \cdots \mathbf{b}_F^o] \in \mathbb{R}_+^{J\times F}$, and $\mathbf{C}^o = [\mathbf{c}_1^o \cdots \mathbf{c}_F^o] \in \mathbb{R}_+^{K\times F}$. We observe the noise corrupted version of $\underline{\mathbf{X}}^o$, $\underline{\mathbf{X}} = \underline{\mathbf{X}}^o + \underline{\mathbf{E}}$. In order to estimate $\mathbf{A}^o$, $\mathbf{B}^o$, and $\mathbf{C}^o$, we compute matrices $\mathbf{A} \in \mathbb{R}_+^{I\times F}$, $\mathbf{B} \in \mathbb{R}_+^{J\times F}$, and $\mathbf{C} \in \mathbb{R}_+^{K\times F}$ that solve the optimization problem

$$\begin{array}{ll} \min\limits_{\mathbf{A},\mathbf{B},\mathbf{C}} & f_{\underline{\mathbf{X}}}(\mathbf{A},\mathbf{B},\mathbf{C}) = \frac{1}{2}\left\|\underline{\mathbf{X}} - [\mathbf{A},\mathbf{B},\mathbf{C}]\right\|_F^2 \\ \text{subject to} & \mathbf{A} \geq \mathbf{O}, \mathbf{B} \geq \mathbf{O}, \mathbf{C} \geq \mathbf{O}, \end{array} \tag{1}$$

where $\mathbf{O}$ is the zero matrix of appropriate dimensions, and the inequalities are element-wise. Let $\underline{\mathbf{Y}} = [\mathbf{A},\mathbf{B},\mathbf{C}]$ and, for every tensor $\underline{\mathbf{W}}$, let $\mathbf{W}^{(1)}$, $\mathbf{W}^{(2)}$, and $\mathbf{W}^{(3)}$ denote the matrix unfoldings of $\underline{\mathbf{W}}$, with respect to the first, second, and third mode, respectively. Then,

$$\mathbf{Y}^{(1)} = \mathbf{A}\left(\mathbf{C}\odot\mathbf{B}\right)^T,\ \mathbf{Y}^{(2)} = \mathbf{B}\left(\mathbf{C}\odot\mathbf{A}\right)^T, \mathbf{Y}^{(3)} = \mathbf{C}\left(\mathbf{B}\odot\mathbf{A}\right)^T,$$

and

$$\begin{aligned} f_{\underline{\mathbf{X}}}(\mathbf{A},\mathbf{B},\mathbf{C}) &= \frac{1}{2}\left\|\mathbf{X}^{(1)} - \mathbf{A}\left(\mathbf{C}\odot\mathbf{B}\right)^T\right\|_F^2 \\ &= \frac{1}{2}\left\|\mathbf{X}^{(2)} - \mathbf{B}\left(\mathbf{C}\odot\mathbf{A}\right)^T\right\|_F^2 \\ &= \frac{1}{2}\left\|\mathbf{X}^{(3)} - \mathbf{C}\left(\mathbf{B}\odot\mathbf{A}\right)^T\right\|_F^2. \end{aligned} \tag{2}$$

These expressions are the basis for alternating least squares (ALS)-type CP optimization. Nonnegative alternating least squares (NALS) is a popular approach for the solution of (1), but nonnegativity brings a significant computational burden relative to plain ALS, and it also complicates the development of parallel algorithms for NTF. The above expressions are also useful in the development of the ADMoM-based NTF algorithm.

3. ADMOM

ADMoM is a technique for the solution of optimization problems of the form [19]

$$\min_{\mathbf{x},\mathbf{z}}\ f(\mathbf{x}) + g(\mathbf{z}) \quad \text{subject to } \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c}, \tag{3}$$

where $\mathbf{x} \in \mathbb{R}^{n_1}$, $\mathbf{z} \in \mathbb{R}^{n_2}$, $\mathbf{A} \in \mathbb{R}^{m\times n_1}$, $\mathbf{B} \in \mathbb{R}^{m\times n_2}$, $\mathbf{c} \in \mathbb{R}^m$, $f : \mathbb{R}^{n_1} \to \mathbb{R}$, and $g : \mathbb{R}^{n_2} \to \mathbb{R}$.

The augmented Lagrangian for problem (3) is

$$\begin{aligned} L_\rho(\mathbf{x},\mathbf{z},\mathbf{y}) = f(\mathbf{x}) + g(\mathbf{z}) &+ \mathbf{y}^T(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}) \\ &+ \frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|_2^2, \end{aligned} \tag{4}$$

where $\rho > 0$ is a penalty parameter. Assuming that at time instant k we have computed $\mathbf{z}^k$ and $\mathbf{y}^k$, which comprise the *state* of the algorithm, the $(k+1)$-st iteration of ADMoM is

$$\begin{aligned} \mathbf{x}^{k+1} &= \operatorname*{argmin}_{\mathbf{x}}\left(f(\mathbf{x}) + \mathbf{y}^{kT}\mathbf{A}\mathbf{x} + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z}^k - \mathbf{c}\|_2^2\right) \\ \mathbf{z}^{k+1} &= \operatorname*{argmin}_{\mathbf{z}}\left(g(\mathbf{z}) + \mathbf{y}^{kT}\mathbf{B}\mathbf{z} + \frac{\rho}{2}\|\mathbf{A}\mathbf{x}^{k+1} + \mathbf{B}\mathbf{z} - \mathbf{c}\|_2^2\right) \\ \mathbf{y}^{k+1} &= \mathbf{y}^k + \rho\left(\mathbf{A}\mathbf{x}^{k+1} + \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c}\right). \end{aligned}$$

4. ADMOM FOR NTF

In order to develop an ADMoM-based NTF algorithm we must put the NTF problem (1) into ADMoM form. Towards this end, we introduce auxiliary variables $\tilde{\mathbf{A}} \in \mathbb{R}^{I\times F}$, $\tilde{\mathbf{B}} \in \mathbb{R}^{J\times F}$, and $\tilde{\mathbf{C}} \in \mathbb{R}^{K\times F}$ and consider the equivalent problem

$$\begin{array}{ll} \min\limits_{\mathbf{A},\tilde{\mathbf{A}},\mathbf{B},\tilde{\mathbf{B}},\mathbf{C},\tilde{\mathbf{C}}} & f_{\underline{\mathbf{X}}}(\mathbf{A},\mathbf{B},\mathbf{C}) + g(\tilde{\mathbf{A}}) + g(\tilde{\mathbf{B}}) + g(\tilde{\mathbf{C}}) \\ \text{subject to} & \mathbf{A} - \tilde{\mathbf{A}} = \mathbf{O}, \mathbf{B} - \tilde{\mathbf{B}} = \mathbf{O}, \mathbf{C} - \tilde{\mathbf{C}} = \mathbf{O}, \end{array} \tag{5}$$

where, for any matrix argument $\mathbf{M}$,

$$g(\mathbf{M}) = \begin{cases} 0, & \text{if } \mathbf{M} \geq \mathbf{O}, \\ \infty, & \text{otherwise.} \end{cases} \tag{6}$$

We introduce dual variables $\mathbf{Y}_\mathbf{A} \in \mathbb{R}^{I\times F}$, $\mathbf{Y}_\mathbf{B} \in \mathbb{R}^{J\times F}$, and $\mathbf{Y}_\mathbf{C} \in \mathbb{R}^{K\times F}$, and the vector of penalty terms $\boldsymbol{\rho} := [\rho_\mathbf{A}\ \rho_\mathbf{B}\ \rho_\mathbf{C}]^T$. The augmented Lagrangian is given in (7), at the top of the next page.

The ADMoM for this problem is given in (4) in the next page (cf. the journal version, also available as an arxiv preprint [18], for the derivation). We observe that, during each ADMoM iteration, we avoid the solution of constrained optimization problems. This seems favorable, especially in the cases where the size of the problem is (very) large.

5. DISTRIBUTED ADMOM FOR LARGE NTF

In this section, we assume that all dimensions of tensor $\underline{\mathbf{X}}$ are large and derive an ADMoM-based NTF that is suitable for parallel implementation.[2] We first partition $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ as $\mathbf{A} = \left[\mathbf{A}_1^T \cdots \mathbf{A}_{N_A}^T\right]^T$, $\mathbf{B} = \left[\mathbf{B}_1^T \cdots \mathbf{B}_{N_B}^T\right]$, $\mathbf{C} = \left[\mathbf{C}_1^T \cdots \mathbf{C}_{N_C}^T\right]$, where $\mathbf{A}_{n_A} \in \mathbb{R}^{I_{n_A}\times F}$, for $n_A = 1,\dots,N_A$, $\sum_{n_A=1}^{N_A} I_{n_A} = I$, $\mathbf{B}_{n_B} \in \mathbb{R}^{J_{n_B}\times F}$, for $n_B = 1,\dots,N_B$, $\sum_{n_B=1}^{N_B} J_{n_B} = J$, and $\mathbf{C}_{n_C} \in \mathbb{R}^{K_{n_C}\times F}$, for $n_C = 1,\dots,N_C$, $\sum_{n_C=1}^{N_C} K_{n_C} = K$. Let $\underline{\mathbf{Y}} = [\mathbf{A},\mathbf{B},\mathbf{C}]$. It turns out that, in order to derive the distributed ADMoM for very large NTFs, we must derive partitionings of the matrix unfoldings, $\mathbf{Y}^{(1)}$, $\mathbf{Y}^{(2)}$, and $\mathbf{Y}^{(3)}$, in terms of (the blocks of) matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$. We start by partitioning $\mathbf{Y}^{(1)}$ as

$$\mathbf{Y}^{(1)} = \begin{bmatrix} \mathbf{Y}_{1,1}^{(1)} & \cdots & \mathbf{Y}_{1,N_C}^{(1)} \\ \vdots & \ddots & \vdots \\ \mathbf{Y}_{N_A,1}^{(1)} & \cdots & \mathbf{Y}_{N_A,N_C}^{(1)} \end{bmatrix},$$

where $\mathbf{Y}_{n_A,n_C}^{(1)} \in \mathbb{R}^{I_{n_A}\times(K_{n_C}J)}$, for $n_A = 1,\dots,N_A$ and $n_C = 1,\dots,N_C$. It is easy to show that $\mathbf{Y}_{n_A,n_C}^{(1)} = \mathbf{A}_{n_A}(\mathbf{C}_{n_C}\odot\mathbf{B})^T$ [18]. Similarly, $\mathbf{Y}^{(2)}$ can be partitioned into blocks $\mathbf{Y}_{n_B,n_C}^{(2)} = \mathbf{B}_{n_B}(\mathbf{C}_{n_C}\odot\mathbf{A})^T$, of dimensions $J_{n_B}\times(IK_{n_C})$, for $n_B = 1,\dots,N_B$ and $n_C = 1,\dots,N_C$, and $\mathbf{Y}^{(3)}$ can be partitioned into blocks $\mathbf{Y}_{n_C,n_B}^{(3)} = \mathbf{C}_{n_C}(\mathbf{B}_{n_B}\odot\mathbf{A})^T$, of dimensions $K_{n_C}\times(IJ_{n_B})$, for $n_C = 1,\dots,N_C$ and $n_B = 1,\dots,N_B$. If we partition

[1]Note that due to the nonnegativity constraints on the latent factors, F can be higher than the rank of $\underline{\mathbf{X}}^o$.

[2]Of course, the cases where only one or two of the dimensions of $\underline{\mathbf{X}}$ are large can be handled similarly.

$$L_{\boldsymbol{\rho}}(\mathbf{A},\mathbf{B},\mathbf{C},\tilde{\mathbf{A}},\tilde{\mathbf{B}},\tilde{\mathbf{C}},\mathbf{Y}_{\mathbf{A}},\mathbf{Y}_{\mathbf{B}},\mathbf{Y}_{\mathbf{C}}) = f_{\underline{\mathbf{X}}}(\mathbf{A},\mathbf{B},\mathbf{C}) + g(\tilde{\mathbf{A}}) + g(\tilde{\mathbf{B}}) + g(\tilde{\mathbf{C}}) + \mathbf{Y}_{\mathbf{A}} * (\mathbf{A}-\tilde{\mathbf{A}}) + \frac{\rho_{\mathbf{A}}}{2}\|\mathbf{A}-\tilde{\mathbf{A}}\|_F^2 \\ + \mathbf{Y}_{\mathbf{B}} * (\mathbf{B}-\tilde{\mathbf{B}}) + \frac{\rho_{\mathbf{B}}}{2}\|\mathbf{B}-\tilde{\mathbf{B}}\|_F^2 + \mathbf{Y}_{\mathbf{C}} * (\mathbf{C}-\tilde{\mathbf{C}}) + \frac{\rho_{\mathbf{C}}}{2}\|\mathbf{C}-\tilde{\mathbf{C}}\|_F^2. \quad (7)$$

$$\begin{aligned}
\mathbf{A}^{k+1} &= \left(\mathbf{X}^{(1)}\,(\mathbf{C}^k \odot \mathbf{B}^k) + \rho_{\mathbf{A}}\tilde{\mathbf{A}}^k - \mathbf{Y}_{\mathbf{A}}^k\right)\left((\mathbf{C}^k \odot \mathbf{B}^k)^T(\mathbf{C}^k \odot \mathbf{B}^k) + \rho_{\mathbf{A}}\mathbf{I}_F\right)^{-1}\\
\mathbf{B}^{k+1} &= \left(\mathbf{X}^{(2)}\,(\mathbf{C}^k \odot \mathbf{A}^{k+1}) + \rho_{\mathbf{B}}\tilde{\mathbf{B}}^k - \mathbf{Y}_{\mathbf{B}}^k\right)\left((\mathbf{C}^k \odot \mathbf{A}^{k+1})^T(\mathbf{C}^k \odot \mathbf{A}^{k+1}) + \rho_{\mathbf{B}}\mathbf{I}_F\right)^{-1}\\
\mathbf{C}^{k+1} &= \left(\mathbf{X}^{(3)}\,(\mathbf{B}^{k+1} \odot \mathbf{A}^{k+1}) + \rho_{\mathbf{C}}\tilde{\mathbf{C}}^k - \mathbf{Y}_{\mathbf{C}}^k\right)\left((\mathbf{B}^{k+1} \odot \mathbf{A}^{k+1})^T(\mathbf{B}^{k+1} \odot \mathbf{A}^{k+1}) + \rho_{\mathbf{C}}\mathbf{I}_F\right)^{-1} \qquad (8)\\
\tilde{\mathbf{A}}^{k+1} &= \left(\mathbf{A}^{k+1} + \frac{1}{\rho_{\mathbf{A}}}\mathbf{Y}_{\mathbf{A}}^k\right)_+,\ \tilde{\mathbf{B}}^{k+1} = \left(\mathbf{B}^{k+1} + \frac{1}{\rho_{\mathbf{B}}}\mathbf{Y}_{\mathbf{B}}^k\right)_+,\ \tilde{\mathbf{C}}^{k+1} = \left(\mathbf{C}^{k+1} + \frac{1}{\rho_{\mathbf{C}}}\mathbf{Y}_{\mathbf{C}}^k\right)_+\\
\mathbf{Y}_{\mathbf{A}}^{k+1} &= \mathbf{Y}_{\mathbf{A}}^k + \rho_{\mathbf{A}}\left(\mathbf{A}^{k+1} - \tilde{\mathbf{A}}^{k+1}\right),\ \mathbf{Y}_{\mathbf{B}}^{k+1} = \mathbf{Y}_{\mathbf{B}}^k + \rho_{\mathbf{B}}\left(\mathbf{B}^{k+1} - \tilde{\mathbf{B}}^{k+1}\right),\ \mathbf{Y}_{\mathbf{C}}^{k+1} = \mathbf{Y}_{\mathbf{C}}^k + \rho_{\mathbf{C}}\left(\mathbf{C}^{k+1} - \tilde{\mathbf{C}}^{k+1}\right).
\end{aligned}$$

$\mathbf{X}^{(1)}$, $\mathbf{X}^{(2)}$, and $\mathbf{X}^{(3)}$ accordingly, then $f_{\underline{\mathbf{X}}}$ can be written as

$$\begin{aligned}
f_{\underline{\mathbf{X}}}(\mathbf{A},\mathbf{B},\mathbf{C}) &= \sum_{n_A=1}^{N_A}\sum_{n_C=1}^{N_C}\frac{1}{2}\|\mathbf{X}^{(1)}_{n_A,n_C} - \mathbf{A}_{n_A}(\mathbf{C}_{n_C}\odot\mathbf{B})^T\|_F^2\\
&= \sum_{n_B=1}^{N_B}\sum_{n_C=1}^{N_C}\frac{1}{2}\|\mathbf{X}^{(2)}_{n_B,n_C} - \mathbf{B}_{n_B}(\mathbf{C}_{n_C}\odot\mathbf{A})^T\|_F^2\\
&= \sum_{n_C=1}^{N_C}\sum_{n_B=1}^{N_B}\frac{1}{2}\|\mathbf{X}^{(3)}_{n_C,n_B} - \mathbf{C}_{n_C}(\mathbf{B}_{n_B}\odot\mathbf{A})^T\|_F^2.
\end{aligned}$$

These expressions are fundamental for the development of the distributed ADMoM for NTF.

5.1. Distributed ADMoM for large NTF

In order to put the very large NTF problem into a form suitable for the ADMoM, we introduce auxuliary variables $\tilde{\mathbf{A}} = [\tilde{\mathbf{A}}_1^T \cdots \tilde{\mathbf{A}}_{N_A}^T]^T$, with $\tilde{\mathbf{A}}_{n_A} \in \mathbb{R}^{I_{n_A}\times F}$, for $n_A = 1,\ldots,N_A$, $\tilde{\mathbf{B}} = [\tilde{\mathbf{B}}_1^T \cdots \tilde{\mathbf{B}}_{N_B}^T]^T$, with $\tilde{\mathbf{B}}_{n_B} \in \mathbb{R}^{J_{n_B}\times F}$, for $n_B = 1,\ldots,N_B$, $\tilde{\mathbf{C}} = [\tilde{\mathbf{C}}_1^T \cdots \tilde{\mathbf{C}}_{N_C}^T]^T$, with $\tilde{\mathbf{C}}_{n_C} \in \mathbb{R}^{K_{n_C}\times F}$, for $n_C = 1,\ldots,N_C$, and consider the equivalent problem

$$\begin{aligned}
\min_{\mathbf{A},\tilde{\mathbf{A}},\mathbf{B},\tilde{\mathbf{B}},\mathbf{C},\tilde{\mathbf{C}}} \quad & f_{\underline{\mathbf{X}}}(\mathbf{A},\mathbf{B},\mathbf{C}) + \textstyle\sum_{n_A=1}^{N_A} g(\tilde{\mathbf{A}}_{n_A})\\
& + \textstyle\sum_{n_B=1}^{N_B} g(\tilde{\mathbf{B}}_{n_B}) + \sum_{n_C=1}^{N_C} g(\tilde{\mathbf{C}}_{n_C})\\
\text{subject to} \quad & \mathbf{A}_{n_A} - \tilde{\mathbf{A}}_{n_A} = \mathbf{O}, \quad n_A = 1,\ldots,N_A,\\
& \mathbf{B}_{n_B} - \tilde{\mathbf{B}}_{n_B} = \mathbf{O}, \quad n_B = 1,\ldots,N_B,\\
& \mathbf{C}_{n_C} - \tilde{\mathbf{C}}_{n_C} = \mathbf{O}, \quad n_C = 1,\ldots,N_C.
\end{aligned} \qquad (10)$$

We introduce dual variables $\mathbf{Y}_{\mathbf{A}} = [\mathbf{Y}_{\mathbf{A}_1}^T \cdots \mathbf{Y}_{\mathbf{A}_{N_A}}^T]^T$, $\mathbf{Y}_{\mathbf{A}_{n_A}} \in \mathbb{R}^{I_{n_A}\times F}$, $n_A = 1,\ldots,N_A$, $\mathbf{Y}_{\mathbf{B}} = [\mathbf{Y}_{\mathbf{B}_1}^T \cdots \mathbf{Y}_{\mathbf{B}_{N_B}}^T]^T$, $\mathbf{Y}_{\mathbf{B}_{n_B}} \in \mathbb{R}^{J_{n_B}\times F}$, $n_B = 1,\ldots,N_B$, $\mathbf{Y}_{\mathbf{C}} = [\mathbf{Y}_{\mathbf{C}_1}^T \cdots \mathbf{Y}_{\mathbf{C}_{N_A}}^T]^T$, $\mathbf{Y}_{\mathbf{C}_{n_C}} \in \mathbb{R}^{K_{n_C}\times F}$, $n_C = 1,\ldots,N_C$. The ADMoM for this problem is given in (9) (a proof is given in the corresponding journal submission [18]).

We observe that, during each ADMoM iteration, we avoid the solution of constrained optimization problems. Furthermore, and, perhaps, more importantly, having computed all algorithm quantities at iteration k, the updates of $\mathbf{A}_{n_A}^k$, for $n_A = 1,\ldots,N_A$, are *independent* and can be implemented in *parallel*. After the computation of $\mathbf{A}^{k+1}$, the updates of $\mathbf{B}_{n_B}^k$, for $n_B = 1,\ldots,N_B$, are also independent and can be implemented in parallel as well. Finally, after the computation of $\mathbf{B}^{k+1}$, we can compute in parallel the updates of $\mathbf{C}_{n_C}^k$, for $n_C = 1,\ldots,N_C$. A description of a mesh-type architecture for the implementation of the distributed ADMoM is provided in [18].

We note that we can solve problem (10) using the 'centralized' ADMoM of Section 4. The difference is that, using the distributed ADMoM developed in this section, we uncover the *inherent parallelism* in the updates of the blocks of $\mathbf{A}^k$, $\mathbf{B}^k$, and $\mathbf{C}^k$. If we initialize the corresponding quantities of the two algorithms with the same values, then both algorithms evolve in *exactly* the same way, see [18] for a proof. Thus, the study (convergence analysis and/or numerical behavior) of one of them is sufficient for the characterization of both.

6. NUMERICAL EXPERIMENTS

In our numerical experiments, we compare ADMoM NTF with (1) NALS NTF, as implemented in the `parafac` routine of the *N-way toolbox* for Matlab [20] and (2) NTF using the nonlinear least-squares solvers (NLS), as implemented in the `sdf_nls` routine of tensorlab [21] (with the non-negativity option turned on in both cases). In all cases, we use random initialization.

In extensive numerical experiments, we have observed that the relative performance of the algorithms depends on the size and rank of the tensor as well as the additive noise power. Thus, we consider 12 different scenarios, corresponding to the combinations of the following cases (1) one, two, or three tensor dimensions are large, (2) rank F is small or large, and (3) additive noise is 'weak' or 'strong'. For each scenario, we generate $R = 50$ realizations of tensor $\underline{\mathbf{X}}$ as follows. We generate random matrices $\mathbf{A}^o$, $\mathbf{B}^o$, and $\mathbf{C}^o$ with i.i.d. $\mathcal{U}[0,1]$ elements (using the `rand` command of Matlab) and construct $\underline{\mathbf{X}} = [\mathbf{A}^o, \mathbf{B}^o, \mathbf{C}^o] + \underline{\mathbf{N}}$, where $\underline{\mathbf{N}}$ consists of i.i.d. $\mathcal{N}(0,\sigma_N^2)$ elements. For each realization, we solve the NTF problem with (1) NALS (`parafac`), (2) NLS (`sdf_nls`), and (3) ADMoM.

We designed our experiments so that, upon convergence, all algorithms achieve practically the same relative factorization error. Towards this end, we set the values of the stopping parameters as follows: the parameter `Options(1)` of `parafac` is set to `Options(1)` $= 10^{-5}$, the parameter `TolFun` of `sdf_nls` is set to `TolFun` $= 10^{-8}$, and the ADMoM stopping parameters are set to

$$
\begin{aligned}
\mathbf{A}_{n_A}^{k+1} &= \left(\left(\sum_{n_C=1}^{N_C} \mathbf{X}^{(1)}_{n_A,n_C} (\mathbf{C}^k_{n_C} \odot \mathbf{B}^k) \right) + \rho_{\mathbf{A}} \tilde{\mathbf{A}}^k_{n_A} - \mathbf{Y}^k_{\mathbf{A}_{n_A}} \right) \left(\left(\sum_{n_C=1}^{N_C} (\mathbf{C}^k_{n_C} \odot \mathbf{B}^k)^T (\mathbf{C}^k_{n_C} \odot \mathbf{B}^k) \right) + \rho_{\mathbf{A}} \mathbf{I}_F \right)^{-1} \\
\mathbf{B}_{n_B}^{k+1} &= \left(\left(\sum_{n_C=1}^{N_C} \mathbf{X}^{(2)}_{n_B,n_C} (\mathbf{C}^k_{n_C} \odot \mathbf{A}^{k+1}) \right) + \rho_{\mathbf{B}} \tilde{\mathbf{B}}^k_{n_B} - \mathbf{Y}^k_{\mathbf{B}_{n_B}} \right) \left(\left(\sum_{n_C=1}^{N_C} (\mathbf{C}^k_{n_C} \odot \mathbf{A}^{k+1})^T (\mathbf{C}^k_{n_C} \odot \mathbf{A}^{k+1}) \right) + \rho_{\mathbf{B}} \mathbf{I}_F \right)^{-1} \\
\mathbf{C}_{n_C}^{k+1} &= \left(\left(\sum_{n_B=1}^{N_B} \mathbf{X}^{(3)}_{n_C,n_B} (\mathbf{B}^{k+1}_{n_B} \odot \mathbf{A}^{k+1}) \right) + \rho_{\mathbf{C}} \tilde{\mathbf{C}}^k_{n_C} - \mathbf{Y}^k_{\mathbf{C}_{n_C}} \right) \left(\left(\sum_{n_B=1}^{N_B} (\mathbf{B}^{k+1}_{n_B} \odot \mathbf{A}^{k+1})^T (\mathbf{B}^{k+1}_{n_B} \odot \mathbf{A}^{k+1}) \right) + \rho_{\mathbf{C}} \mathbf{I}_F \right)^{-1} \\
\tilde{\mathbf{A}}_{n_A}^{k+1} &= \left(\mathbf{A}^{k+1}_{n_A} + \frac{1}{\rho_{\mathbf{A}}} \mathbf{Y}^k_{\mathbf{A}_{n_A}} \right)_+ ,\ \tilde{\mathbf{B}}^{k+1}_{n_B} = \left(\mathbf{B}^{k+1}_{n_B} + \frac{1}{\rho_{\mathbf{A}}} \mathbf{Y}^k_{\mathbf{B}_{n_B}} \right)_+ ,\ \tilde{\mathbf{C}}^{k+1}_{n_C} = \left(\mathbf{C}^{k+1}_{n_C} + \frac{1}{\rho_{\mathbf{A}}} \mathbf{Y}^k_{\mathbf{C}_{n_C}} \right)_+ \\
\mathbf{Y}^{k+1}_{\mathbf{A}_{n_A}} &= \mathbf{Y}^k_{\mathbf{A}_{n_A}} + \rho_{\mathbf{A}} \left(\mathbf{A}^{k+1}_{n_A} - \tilde{\mathbf{A}}^{k+1}_{n_A} \right),\ \mathbf{Y}^{k+1}_{\mathbf{B}_{N_B}} = \mathbf{Y}^k_{\mathbf{B}_{n_B}} + \rho_{\mathbf{B}} \left(\mathbf{B}^{k+1}_{n_B} - \tilde{\mathbf{B}}^{k+1}_{n_B} \right),\ \mathbf{Y}^{k+1}_{\mathbf{C}_{n_C}} = \mathbf{Y}^k_{\mathbf{C}_{n_C}} + \rho_{\mathbf{C}} \left(\mathbf{C}^{k+1}_{n_C} - \tilde{\mathbf{C}}^{k+1}_{n_C} \right)
\end{aligned} \tag{9}
$$

Size	F	σ_N^2	mean(RFE)	NALS mean(t)	NLS mean(t)	ADMoM mean(t)	NALS std(t)	NLS std(t)	ADMoM std(t)
$3000 \times 50 \times 50$	3	10^{-2}	0.2156	12.4010	17.2584	7.1806	1.4770	6.9348	3.8647
		10^{-4}	0.0221	16.6500	16.5962	7.5098	1.9625	3.3589	4.2499
	30	10^{-2}	0.0260	212.0598	115.7030	110.4128	11.4579	8.0409	91.4882
		10^{-4}	0.0026	270.6674	117.2152	148.5052	11.5324	11.3154	149.0542
$400 \times 400 \times 50$	3	10^{-2}	0.2175	8.6174	4.7124	8.0710	0.9264	1.4596	4.4952
		10^{-4}	0.0222	11.0670	4.8916	8.3614	1.2075	1.6443	3.1833
	30	10^{-2}	0.0260	71.1950	31.7546	106.1362	8.2119	3.2162	76.5743
		10^{-4}	0.0026	92.4838	31.1690	94.5734	7.9280	3.7062	88.1671
$200 \times 200 \times 200$	5	10^{-2}	0.1400	10.9142	4.1756	12.5190	1.2783	0.7915	2.1334
		10^{-4}	0.0143	14.2882	4.1070	12.6184	2.5817	0.9059	2.1616
	30	10^{-2}	0.0260	55.0806	16.1838	33.9268	4.4886	1.3649	12.2687
		10^{-4}	0.0026	70.0238	16.7624	32.1670	6.5737	1.4152	10.0182

Table 1. Mean relative factorization error and mean and standard deviation of `cputime`, in sec, for NALS, NLS and ADMoM NTF.

$\epsilon^{\text{abs}} = 10^{-4}$ and $\epsilon^{\text{rel}} = 10^{-4}$.

In practice, convergence properties of ADMoM NTF depend on the (random) initialization point. In some cases, convergence may be quite fast while, in others, it may be quite slow. As we shall see in the sequel, this phenomenon seems more prominent in the cases where rank F is 'large.' In order to overcome this problem, we adopted the following strategy. We execute ADMoM NTF for up to $n_{\max} = 400$ iterations (we have observed that, in the great majority of the cases in the scenarios we examined, this number of iterations is sufficient for convergence when we start from a 'good' initial point). If ADMoM does not converge within $n_{\max}$ iterations, then we restart it from another random initial point; we repeat this procedure until ADMoM converges.

Since an accurate statement about the computational complexity per iteration of `parafac` is not easy, the metric we used for comparison of the algorithms is the `cputime` of Matlab. Despite the fact that `cputime` is strongly depended on computer hardware and the actual algorithm implementation, we feel that it is a useful metric for the assessment of the relative efficiency of the algorithms.[3]

In Table 1, we present the mean and standard deviation of `cputime`, in seconds, denoted as `mean(t)` and `std(t)`, respectively, for NALS, NLS, and ADMoM. We also present the mean relative factorization error (which is common to all algorithms), denoted as `mean(RFE)`. We observe that there is no clear winner. Certainly, for high ranks, NLS has very good behavior. In general, both NALS and NLS have more 'predictable' behavior than ADMoM. Especially for high ranks, the `cputime` of our implementation of ADMoM has very high variance. For small ranks, ADMoM looks more competitive and, in the cases where one dimension is much larger than the other two, it behaves very well.

7. CONCLUSION

Motivated by emerging big data applications, we developed a new constrained tensor factorization framework based on ADMoM. We used non-negative factorization of third order tensors as an example to work out the main ideas, but our approach can be generalized to higher order tensors, many other types of constraints on the latent factors, as well as other tensor factorizations and tensor completion. Our numerical experiments were encouraging, indicating that, in many cases, ADMoM-based NTF has high potential as an alternative to the state-of-the-art and, in some cases, it may become state-of-the-art. The fact that it is naturally amenable to parallel implementation can only increase its potential. We are currently working towards further improvements of the core algorithm.

[3]For our experiments, we run Matlab 2014a on a MacBook Pro with a 2.5 GHz Intel Core i7 Intel processor and 16 GB RAM.

8. REFERENCES

[1] N. Sidiropoulos, R. Bro, and G. Giannakis, "Parallel factor analysis in sensor array processing," *IEEE Transactions on Signal Processing*, vol. 48, no. 8, pp. 2377–2388, 2000.

[2] N. Sidiropoulos, G. Giannakis, and R. Bro, "Blind PARAFAC receivers for DS-CDMA systems," *IEEE Transactions on Signal Processing*, vol. 48, no. 3, pp. 810–823, 2000.

[3] D. Nion, K. Mokios, N. Sidiropoulos, and A. Potamianos, "Batch and adaptive PARAFAC-based blind separation of convolutive speech mixtures," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1193–1207, 2010.

[4] C. Fevotte and A. Ozerov, "Notes on nonnegative tensor factorization of the spectrogram for audio source separation: Statistical insights and towards self-clustering of the spatial cues," in *Exploring Music Contents*, ser. Lecture Notes in Computer Science, S. Ystad, M. Aramaki, R. Kronland-Martinet, and K. Jensen, Eds. Springer Berlin, 2011, vol. 6684, pp. 102–115.

[5] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos, "Parcube: Sparse parallelizable tensor decompositions." in *ECML/PKDD (1)*, ser. Lecture Notes in Computer Science, P. A. Flach, T. D. Bie, and N. Cristianini, Eds., vol. 7523. Springer, 2012, pp. 521–536.

[6] R. Bro and N. Sidiropoulos, "Least squares regression under unimodality and non-negativity constraints," *Journal of Chemometrics*, vol. 12, pp. 223–247, 1998.

[7] A. Cichocki, D. Mandic, C. Caiafa, A.-H. Phan, G. Zhou, Q. Zhao, and L. De Lathauwer, "Multiway Component Analysis: Tensor Decompositions for Signal Processing Applications," *IEEE Signal Processing Magazine*, 2014 (to appear).

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

[9] ——, "Determination and proof of minimum uniqueness conditions for PARAFAC-1," *UCLA Working Papers in Phonetics*, vol. 22, pp. 111–117, 1972.

[10] J. Carroll and J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of Eckart-Young decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.

[11] Apache, "Hadoop." [Online]. Available: http://hadoop.apache.org/

[12] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[13] U. Kang, E. Papalexakis, A. Harpale, and C. Faloutsos, "Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 316–324.

[14] N. Sidiropoulos, E. Papalexakis, and C. Faloutsos, "A Parallel Algorithm for Big Tensor Decomposition Using Randomly Compressed Cubes (PARACOMP)," in *Proc. IEEE ICASSP 2014,* May 4-9, Florence, Italy.

[15] ——, "Parallel Randomly Compressed Cubes: A Scalable Distributed Architecture for Big Tensor Decomposition," *IEEE Signal Processing Magazine*, Sep. 2014.

[16] A. de Almeida and A. Kibangou, "Distributed computation of tensor decompositions in collaborative networks," in *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2013 IEEE 5th International Workshop on*, Dec 2013, pp. 232–235.

[17] ——, "Distributed large-scale tensor decomposition," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, May 2014.

[18] A. P. Liavas and N. D. Sidiropoulos, "Parallel algorithms for constrained tensor decomposition via the alternating direction method of multipliers," *IEEE Trans. on Signal Processing*, submitted Aug. 30, 2014. [Online]. Available: http://arxiv.org/abs/1409.2383

[19] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011. [Online]. Available: http://dx.doi.org/10.1561/2200000016

[20] C. A. Andersson and R. Bro, "The n-way toolbox for matlab." [Online]. Available: http://www.models.life.ku.dk/source/nwaytoolbox

[21] L. Sorber, M. Van Barel, and L. De Lathauwer, "Tensorlab v2.0." [Online]. Available: http://www.tensorlab.net/