Singleshot: a scalable Tucker tensor decomposition
Abraham Traoré, Maxime Berar, Alain Rakotomamonjy

To cite this version:

HAL Id: hal-02288245
https://hal.archives-ouvertes.fr/hal-02288245
Submitted on 14 Sep 2019
Singleshot: a scalable Tucker tensor decomposition
Abraham Traoré, Maxime Berar, Alain Rakotomamonjy

To cite this version:

HAL Id: hal-02288245
https://hal.archives-ouvertes.fr/hal-02288245
Submitted on 14 Sep 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Singleshot: a scalable Tucker tensor decomposition

Abraham Traoré  Maxime Berar  Alain Rakotomamonjy

Abstract

This paper introduces a new approach for the scalable Tucker decomposition problem. Given a tensor $\mathcal{X}$, the algorithm proposed, named Singleshot, allows to perform the inference task by processing one subtensor drawn from $\mathcal{X}$ at a time. The key principle of our approach is based on the recursive computations of the gradient and on cyclic update of the latent factors involving only one single step of gradient descent. We further improve the computational efficiency of Singleshot by proposing an inexact gradient version named Singleshotinexact. The two algorithms are backed with theoretical guarantees of convergence and convergence rates under mild conditions. The scalability of the proposed approaches, which can be easily extended to handle some common constraints encountered in tensor decomposition (e.g. non-negativity), is proven via numerical experiments on both synthetic and real data sets.

1 Introduction

Tensor decomposition techniques are promising tools for exploratory analysis of multidimensional data in diverse disciplines including signal processing [11] and social networks analysis [25]. The two most common decompositions used for tensor analysis are the Tucker decomposition, introduced by [40] and the Canonical Polyadic Decomposition also named CPD introduced independently by [16] and by [6]. These decompositions are used to infer multilinear relationships from multidimensional datasets as they allow to extract hidden (latent) components and investigate the relationships among them.

In this paper, we focus on the Tucker decomposition motivated by the fact that this decomposition and its variants have been successfully used in many real applications [21, 19]. Our technical goal is to develop a scalable Tucker decomposition technique in a static setting (the tensor is fixed).

Such an objective is relevant in a situation where it is not possible to load in memory the tensor of interest or when the decomposition process may result in memory overflow generated by intermediate computations [20, 28].

1.1 Related work and main limitations

Divide-and-conquer type methods (i.e. which divide the data set into sub-parts) have already been proposed for the scalable Tucker decomposition problem, with the goal of efficiently decomposing a large fixed tensor (static setting). There are mainly three trends for these methods: distributed methods, sequential processing of small subsets of entries drawn from the tensor or the computation of the tensor-matrix product in a piecemeal fashion by adaptively selecting the order of the computations. A variant of the Tucker-ALS has been proposed in [28] and it solves each alternate step of the Tucker decomposition by processing on-the-fly intermediate data, reducing then the memory footprint of the algorithm. Several other approaches following the same principles are given in [5, 2, 4, 30] while others consider some sampling strategies [26, 33, 14, 36, 48, 18, 44, 32, 24, 10, 22] or distributed approaches [46, 7, 31]. One major limitation related to these algorithms is their lack of genericness (i.e. they cannot be extended to incorporate some constraints such as non-negativity).
Another set of techniques for large-scale Tucker decomposition in a static setting focuses on designing both deterministic and randomized algorithms in order to alleviate the computational burden of the decomposition. An approach proposed by [4] performs an alternate minimization and reduces the scale of the intermediate problems via the incorporation of sketching operators. In the same flavor, one can reduce the computational burden of the standard method HOSVD through randomization and by estimating the orthonormal basis via the so-called range finder algorithm [48]. This class of approaches encompasses other methods that can be either random [8], [27], [13], [34], [39], [43] or deterministic [37], [2], [33], [3], [47], [17], [23], [29]. The main limitation of these methods essentially stems from the fact that they use the whole data set at once (instead of dividing it), which makes them non-applicable when the tensor does not fit the available memory.

From a theoretical point of view, among all these works, some algorithms are backed up with convergence results [4] or have quality of approximation guarantees materialized by a recovery bound [1]. However, there is a lack of convergence rate analysis for the scalable Tucker problem.

1.2 Main contributions

In contrast to the works described above, our contributions are the following ones:

- We propose a new approach for the scalable Tucker decomposition problem, denoted as Singleshot leveraging on coordinate gradient descent [38] and sequential processing of data chunks amenable to constrained optimization.
- In order to improve the computational efficiency of Singleshot, we introduce an inexact gradient variant, denoted as Singleshotinexact. This inexact approach can be further extended so as to make it able to decompose a tensor growing in every mode and in an online fashion.
- From a theoretical standpoint, we establish for Singleshot an ergodic convergence rate of $O\left(\frac{1}{Kn}\right)$ ($K$: maximum number of iterations) to a stationary point and for Singleshotinexact, we establish a convergence rate of $O\left(\frac{1}{k}\right)$ ($k$ being the iteration number) to a minimizer.
- We provide experimental analyses showing that our approaches are able to decompose bigger tensors than competitors without compromising efficiency. From a streaming tensor decomposition point of view, our Singleshot extension is competitive with its competitor.

2 Notations & Definitions

A N-order tensor is denoted by a boldface Euler script letter $\mathbf{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$. The matrices are denoted by bold capital letters (e.g., $\mathbf{A}$). The identity matrix is denoted by $\mathbf{I}$. The $j^{th}$ row of a matrix $\mathbf{A} \in \mathbb{R}^{J \times L}$ is denoted by $\mathbf{A}_j$. The transpose of a matrix $\mathbf{A}$ is denoted by $\mathbf{A}^\top$.

Matricization is the process of reordering all the elements of a tensor into a matrix. The mode-$n$ matricization of a tensor $[\mathbf{X}]^{(n)}$ arranges the mode-$n$ fibers to be the columns of the resulting matrix $\mathbf{X}^{(n)} \in \mathbb{R}^{I_n \times \prod_{i \neq n} I_i}$. The mode-$n$ product of a tensor $\mathbf{G} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ with a matrix $\mathbf{A} \in \mathbb{R}^{I_n \times J_n}$ denoted by $\mathbf{G} \times_n \mathbf{A}$ yields a tensor of the same order whose mode-$n$ matricized form is defined by: $\mathbf{B}^{(n)} = \mathbf{A} \mathbf{G}^{(n)}$. For a tensor $\mathbf{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, its $i_n^{th}$ subtensor with respect to the mode $n$ is denoted by $\mathbf{X}^{(n)}_{i_n} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N}$. This subtensor is a $N$-order tensor defined via the mapping between its $n$-mode matricization $[\mathbf{X}_{i_n}^{(n)}]^{(n)}$ and the $i_n^{th}$ row of $\mathbf{X}^{(n)}$, i.e. the tensor $\mathbf{X}_{i_n}^{(n)}$ is obtained by reshaping the $i_n^{th}$ row of $\mathbf{X}^{(n)}$, with the target shape $(I_1, \ldots, I_{n-1}, 1, I_{n+1}, \ldots, I_N)$. The set of integers from $n$ to $N$ is denoted by $I_N = \{n, \ldots, N\}$: if $n = 1$, the set is simply denoted by $I_N$. The set of integers from 1 to $N$ with $n$ excluded is denoted by $I_{N \neq n} = \{1, \ldots, n-1, n+1, \ldots, N\}$. Let us define the tensor $\mathbf{G} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $N$ matrices $\{\mathbf{A}^{(m)} \in \mathbb{R}^{I_m \times J_m}\}_{m \in I_N}$.

The product of $\mathbf{G}$ with the matrices $\mathbf{A}^{(m)}$, $m \in I_N$ denoted by $\mathbf{G} \times_{m \in I_N} \mathbf{A}^{(m)}$ will be alternatively expressed by:

$$\mathbf{G} \times_{m \in I_{N \neq n}} \mathbf{A}^{(m)} = \mathbf{G} \times_{m \in I_{n-1}} \mathbf{A}^{(m)} \times_n \mathbf{A}^{(n)} \times_{q \in I_N^{n+1}} \mathbf{A}^{(q)} = \mathbf{G} \times_{m \in I_N \neq n} \mathbf{A}^{(m)} \times_n \mathbf{A}^{(n)}.$$ 

The Frobenius norm of a tensor $\mathbf{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, denoted by $\|\mathbf{X}\|_F$ is defined by:

$$\|\mathbf{X}\|_F = \left(\sum_{1 \leq i_1 \leq I_1, \ldots, 1 \leq n \leq N} \mathbf{X}_{i_1, \ldots, i_N}^2\right)^{\frac{1}{2}}.$$ 

The same definition holds for matrices.
3 Piecewise tensor decomposition: Singleshot

3.1 Tucker decomposition and problem statement

Given a tensor $X \in \mathbb{R}^{I_1 \times \ldots \times I_N}$, the Tucker decomposition aims at the following approximation:

$$X \approx \mathcal{G} \times_{m} A^{(m)}, \mathcal{G} \in \mathbb{R}^{J_1 \times \ldots \times J_N}, A^{(m)} \in \mathbb{R}^{I_m \times J_m}$$

The tensor $\mathcal{G}$ is generally named the core tensor and the matrices $\{A^{(m)}\}_{m \in I_N}$ the loading matrices. A natural way to tackle this problem is to infer the latent factors $\mathcal{G}$ and $\{A^{(m)}\}_{m \in I_N}$ in such a way that the discrepancy is low. Thus, the decomposition of $X$ is usually obtained by solving the following optimization problem:

$$\min_{\mathcal{G}, A^{(1)}, \ldots, A^{(N)}} \left\{ \frac{1}{2} \| X - \mathcal{G} \times_{m} A^{(m)} \|_F^2 \right\}$$ (1)

Our goal in this work is to solve the above problem, for large tensors, while addressing two potential issues: the processing of a tensor that does not fit into the available memory and avoiding memory overflow problem generated by intermediate operations during the decomposition process.

For this objective, we leverage on a reformulation of the problem (1) in terms of subtensors drawn from $X$ with respect to one mode (which we suppose to be predefined), the final objective being to address the case of subtensors drawn with respect to one mode and the general case can be derived following the same principle (see supplementary material, section 5).

3.2 Singleshot

Since the problem (1) does not admit any analytic solution, we propose a numerical resolution based on coordinate gradient descent [38]. The underlying idea is based on a cyclic update over each of the variables $\mathcal{G}, A^{(1)}, \ldots, A^{(N)}$ while fixing the others at their last updated values and each update being performed via a single iteration of gradient descent. More formally, given at iteration $k$, $\mathcal{G}_k, A^{(1)}_k, \ldots, A^{(N)}_k$ the value of the latent factors, the derivatives $\mathcal{D}_k^p$ and $\mathcal{D}_k^p$ of $f$ with respect to the core tensor and the $p^{th}$ loading matrix respectively evaluated at $(\mathcal{G}_k, A^{(1)}_k, \ldots, A^{(N)}_k)$ and $(\mathcal{G}_{k+1}, A^{(1)}_{k+1}, \ldots, A^{(p-1)}_k, A^{(p)}_k, A^{(N)}_k)$ are given by:

$$\mathcal{D}_k^p = \partial_{\mathcal{G}} f \left( \mathcal{G}_k, \left\{ A^{(m)}_k \right\}_{1}^{N} \right), \quad \mathcal{D}_k^p = \partial_{A^{(p)}} f \left( \mathcal{G}_{k+1}, \left\{ A^{(m)}_{k+1} \right\}_{1}^{p-1}, \left\{ A^{(q)}_k \right\}_{p}^{N} \right)$$ (3)

The resulting cyclic update algorithm, named Singleshot, is summarized in Algorithm 1. A naive implementation of the gradient computation would result in memory overflow problem. In what follows, we show that the derivatives $\mathcal{D}_k^p$ and $\mathcal{D}_k^p, 1 \leq p \leq N$ given by the equation (3) can be computed by processing a single subtensor at a time, making Algorithm 1 amenable to sequential processing of subtensors. Discussions on how the step sizes are obtained will be provide in Section 4.

Derivative with respect to $\mathcal{G}$. The derivative with respect to the core tensor is given by (details in Property 7 of supplementary material):

$$\mathcal{D}_k^p = \sum_{i_n=1}^{I_n} \mathcal{R}_{i_n} \times_m A^{(m)}_k \times n \left( \left( A^{(m)}_k \right)_{i_n,:} \right)^T, \quad \mathcal{R}_{i_n} = -X_{i_n} + \mathcal{G}_k \times_m A^{(m)}_k \times n \left( A^{(n)}_k \right)_{i_n,:} \left( A^{(m)}_k \right)_{i_n,:}^T$$ (4)
Algorithm 1 \textit{Singleshot}

\begin{itemize}
\item \textbf{Inputs:} \( \mathcal{X} \) tensor of interest, \( n \) splitting mode, \( \left\{ A^{(m)}_0 \right\}_{1 \leq m \leq N} \) initial loading matrices,
\item \textbf{Output:} \( G, \left\{ A^{(m)} \right\}_{1 \leq m \leq N} \)
\item \textbf{Initialization:} \( k = 0 \)
\end{itemize}

1: while a predefined stopping criterion is not met do
2: Compute optimal step \( \eta_k \)
3: \( G_{k+1} \leftarrow G_k - \eta_k \hat{D}_k^p \) with \( \hat{D}_k^p \) given by (4)
4: for \( p \) from 1 to \( N \) do
5: Compute optimal step \( \eta_k^p \)
6: \( A^{(p)}_{k+1} \leftarrow A^{(p)}_k - \eta_k^p \hat{D}_k^p \) with \( \hat{D}_k^p \) given by (5)
7: end for
8: end while

It is straightforward to see that \( \hat{D}_k^p \) (given by the equation (4)) is the last term of the recursive sequence \( \left\{ (\hat{D}_k^p)^j \right\}_{1 \leq j \leq j_n} \) defined as \( (\hat{D}_k^p)^j = (\hat{D}_k^p)^{j-1} + \theta_j \), with \( (\hat{D}_k^p)^0 \) being the null tensor. An important observation is that the additive term \( \theta_j \) (given by the equation (4)) depends only on one single subtensor \( \mathcal{X}_i^n \). This is the key of our approach since it allows the computation of \( \hat{D}_k^p \) through the sequential processing of a single subtensor \( \mathcal{X}_i^n \) at a time.

Derivatives with respect to \( A^{(p)}, p \neq n \) (\( n \) being the splitting mode). For those derivatives, we can exactly follow the same reasoning, given in detail in Property 9 of the Supplementary material, and obtain for \( p < n \) (the case \( p > n \) yields a similar formula):

\[
\hat{D}_k^p = \sum_{i_{n-1}=1}^{I_{n-1}} \left( - (X_{i_n}^{(p)})^T + A^{(p)}_k B^{(p)}_{i_n} \right) (B^{(p)}_{i_n})^T
\]  

(5)

The matrices \( (X_{i_n}^{(p)})^T \) and \( B^{(p)}_{i_n} \) represent respectively the mode-\( p \) matricized forms of the \( i_{n-1} \)th subtensor \( \mathcal{X}_i^n \) and the tensor \( B^{(p)}_{i_n} \) is defined by:

\[
B_{i_n} = G_{k+1} \times_m A^{(m)}_{k+1} \times_p \text{Id} \times_q A^{(q)}_k \times_n (A^{(n)}_k)_{i_{n-1}}
\]

with \( \text{Id} \in \mathbb{R}^{J_p \times J_p} \) being the identity matrix. With a similar reasoning as for the derivative with respect to the core, it is straightforward to see that \( \hat{D}_k^p \) can be computed by processing a single subtensor at a time.

Derivative with respect to \( A^{(n)} \) (\( n \) being the splitting mode). The derivative with respect to the matrix \( A^{(n)} \) can be computed via the row-wise stacking of independent terms, that are, the derivatives with respect to the rows \( A^{(n)}_j \) and the derivative of \( f \) with respect to \( A^{(n)}_j \) depends only on \( \mathcal{X}_j^n \). Indeed, let’s consider \( 1 \leq j \leq I_n \). In the expression of the objective function \( f \) given by the equation (2), the only term that depends on \( A^{(n)}_j \) is \( \| \mathcal{X}_j^n - G \times_m A^{(m)}_j \times_p B \times_q A^{(q)} \|_F^2 \), thus the derivative of \( f \) with respect to \( A^{(n)}_j \) depends only on \( \mathcal{X}_j^n \) and is given by (see property 8 in the supplementary material):

\[
\partial_{A^{(n)}_j} f (G, \left\{ A^{(m)} \right\}_1^N) = - \left( (X_{i_n}^{(n)})_j - A^{(n)}_j B^{(n)} \right) B^{(n) \top}
\]

The tensors \( (X_{i_n}^{(n)})_j \in \mathbb{R}^{1 \times \prod_{k \neq n} I_k} \) and \( B^{(n)} \) respectively represent the mode-n matricized form of the tensors \( \mathcal{X}_j^n \) and \( B \) with \( B = G \times_p A^{(p)} \times_n \text{Id} \times_q A^{(q)} \times_k Id \in \mathbb{R}^{J_n \times J_n} \); identity matrix.

Remark 1. We stress that all entries of the tensor \( \mathcal{X} \) have been entirely processed when running Algorithm \( \text{[7]} \) and our key action is the sequential processing of subtensors \( \mathcal{X}_{i_n}^n \). In addition, if
one sub-tensor does not fit in the available memory, the recursion, as shown in section 5 of the supplementary material, can still be applied to sub-tensors of the form $X_{\theta_1, \ldots, \theta_N}$, $\theta_m \in \{1, 2, \ldots, I_m\}$ with $(X_{\theta_1, \ldots, \theta_N})_{i_1, \ldots, i_N} = X_{i_1, \ldots, i_N} \in \theta_1 \times \ldots \times \theta_N$, $\times$ referring to the Cartesian product.

### 3.3 Singleshotinexact

While all of the sub-tensors $X^n_{i_n}$, $1 \leq i_n \leq I_n$ are considered in the Singleshot algorithm for the computation of the gradient, in Singleshotinexact, we propose to use only a subset of them for the sake of reducing computational time. The principle is to use for the gradient computation only $B_k < I_n$ sub-tensors. Let’s consider the set $SE\tilde{T}_k$ (of cardinality $B_k$) composed of the integers representing the indexes of the sub-tensors that are going to be used at iteration $k$. The numerical resolution scheme is identical to the one described by Algorithm 1 except for the definition of $D_k^q$ and $D_k^p$ which are respectively replaced by $\tilde{D}_k^q$ and $\tilde{D}_k^p$, $p \neq n$ defined by:

$$\tilde{D}_k^q = \sum_{i_n \in SE\tilde{T}_k} \mathcal{R}_{i_n \times m} A_k^{(n)} \mathcal{I}_{\tilde{S} \setminus n} \left( A_k^{(n)}(i_n, \cdot) \right)^\top, \tilde{D}_k^p = \sum_{i_n \in SE\tilde{T}_k} \left( -(X_n^{(p)}) + A^{(p)} B_n^{(p)}(i_n) \right) \left( B_n^{(p)}(i_n) \right)^\top \ \ (6)$$

For the theoretical convergence, the descent steps are defined as $\frac{q}{\tilde{D}_k^q}$ and $\frac{p}{\tilde{D}_k^p}$, $1 \leq p \leq N$. It is worth to highlight that the derivative $\tilde{D}_k^n$ ($n$ being the mode with respect to which the sub-tensors are drawn) is sparse: Singleshotinexact amounts to minimize $f$ defined by (2) by dropping the terms

$$\left\{ \|X_j^n - G \cdot A^{(m)} \times_n A^{(n)} \|_F^2 \right\}_{m \in SE\tilde{T}_k}$$

with $j \notin SE\tilde{T}_k$, thus, the rows $\left( \tilde{D}_k^p \right)_{j, j \notin SE\tilde{T}_k}$ are all equal to zero.

### 3.4 Discussions

First, we discuss the space complexity needed by our algorithms supposing that the sub-tensors are drawn with respect to one mode. Let’s denote by $n$ the splitting mode. For Singleshot and Singleshot-inexact, at a same mode, we only need to have in memory the tensor $X^n_j$ of size $\prod_{m \in I_N \setminus n} I_m = I_{n-1} I_n + 1 \ldots I_N$, the matrices $\{A^{(m)}\}_{m \in I_N \setminus n}$ and the previous iterate of the gradient. Thus, the complexity in space is $\prod_{m \in I_N \setminus n} I_m + \sum_{m \in I_N \setminus n} J_m + J_n + A T$ with $A T$ being the space complexity of the previous gradient iterate: for the core update, $A T = \prod_{m \in I_N} J_m$ and for a matrix $A^{(m)}$, $A T = I_m J_m$. If the recursion used for the derivatives computation is applied to sub-tensors of the form $X_{\theta_1, \ldots, \theta_N}$, the space complexity is smaller than these complexities.

Another variant of Singleshotinexact can be derived to address an interesting problem that has received little attention so far [4], that is the decomposition of a tensor streaming in every mode with a single pass constraint (i.e., each chunk of data is processed only once) named Singleshotonline. This is enabled by the inexact gradient computation which uses only sub-tensors that are needed. In the streaming context, the gradient is computed based only on the available sub-tensor. Positivity constraints is one of the most encountered constraints in tensor computation and we can simply handle those constraints via the so-called projected gradient descent [42]. This operation does not alter the space complexity with respect to the unconstrained case, since no addition storage is required but increases the complexity in time. For more details, see the section 3 in the supplementary material for the algorithmic details for the proposed variants.

### 4 Theoretical result

Let’s consider the minimization problem (1):

$$\min_{\mathcal{G}, \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}} f \left( \mathcal{G}, \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)} \right)$$

By denoting the block-wise derivative by $\partial_k f$, the derivative of $f$, denoted $\nabla f$ and defined by $(\partial_1 f, \partial_2 f, \ldots, \partial_N f, f)$, is an element of $\mathbb{R}^{J_1 \times J_2 \times \ldots \times J_N \times J_1 \times J_2 \times \ldots \times J_N \times J_N}$ endowed with the norm $\|\cdot\|_F$ defined as the sum of the Frobenius norms. Besides, let’s consider, for writing simplicity, the alternative notations of $f(\mathcal{G}, \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)})$ given by: $f(\mathcal{G}, \{\mathbf{A}^{(m)}\}_{m=1}^{N})$, $f(\mathcal{G}, \{\mathbf{A}^{(m)}\}_{m=1}^{N})^{p}$, $\{\mathbf{A}^{(q)}\}_{q=p+1}^{N}$.
For the theoretical guarantees which details have been reported in the supplementary material, we consider the following assumptions:

**Assumption 1.** Uniform boundedness. The $n^{th}$ subtensors are bounded: $\|X^n_n\|_F \leq \rho$.

**Assumption 2.** Boundedness of factors. We consider the domain $G \in D_g, A^{(m)} \in D_m$ with:

$$D_g = \{\|G_n\|_F \leq \alpha\}, D_m = \{\|A^{(m)}\|_F \leq \alpha\}$$

### 4.1 Convergence result of Singleshot

For the convergence analysis, we consider the following definitions of the descent steps $\eta^G_k$ and $\eta^p_k$ at the $(k+1)^{th}$ iteration:

$$\eta^G_k = \arg \min_{\eta \in [\frac{\delta_1}{\sqrt{K}}, \frac{\delta_2}{\sqrt{K}}]} (\eta - \frac{\delta_1}{\sqrt{K}}) \phi_\eta(\eta), \eta^p_k = \arg \min_{\eta \in [\frac{\delta_1}{\sqrt{K}}, \frac{\delta_2}{\sqrt{K}}]} (\eta - \frac{\delta_1}{\sqrt{K}}) \phi_p(\eta)$$

(7)

$$\phi_\eta(\eta) = f \left( G_k - \eta D^G, \{A^{(m)}_k\}^N_{1} \right) - f \left( G_k, \{A^{(m)}_k\}^N_{1} \right)$$

$$\phi_p(\eta) = f \left( G_{k+1}, \{A^{(m)}_{k+1}\}^{p-1}_{1}, A^{(p)}_k - \eta D^p, \{A^{(q)}_k\}^N_{p+1} \right) - f \left( G_{k+1}, \{A^{(m)}_{k+1}\}^{p-1}_{1}, \{A^{(q)}_k\}^N_{p} \right)$$

and $\delta_2 > \delta_1 > 0$ being user-defined parameters. The motivation of the problems given by the equation (7) is to ensure a decreasing of the objective function after each update. Also note that, the minimization problems related to $\eta^G_k$ and $\eta^p_k$ are well defined since all the factors involved in their definitions are known at the computation stage of $G_{k+1}$ and $A^{(p)}_k$ and correspond to the minimization of a continuous function on a compact set.

Along with Assumption[1] and Assumption[2] as well as the definitions given by (7), we assume that:

$$\frac{\delta_1}{\sqrt{K}} < \eta^G_k, \eta^p_k \leq \frac{\delta_2}{\sqrt{K}}$$

(8)

This simply amounts to consider that the solutions of the minimization problems defined by the equation (7) are not attained at the lower bound of the interval. Under this framework, we establish, as in standard non-convex settings [12], an ergodic convergence rate. Precisely, we prove that:

$$\exists K_0 \geq 1, \forall K \geq K_0, \frac{1}{K} \sum_{k=0}^{K-1} \|\nabla f \left( G_k, \{A^{(m)}_k\}^N_{1} \right) \|_*^2 \leq \frac{(N+1)\Delta}{\sqrt{K}}$$

(9)

with $\|\nabla f \left( G_k, \{A^{(m)}_k\}^N_{1} \right) \|_*^2 = \|\partial G f(G_k, \{A^{(m)}_k\}^N_{1})\|_F + \sum_{p=1}^{N} \|\partial A^{(p)} f(G_k, \{A^{(m)}_k\}^N_{1})\|_F$, $
\Delta = \frac{\delta_2}{\delta_1^2} \left( 2\Gamma + \alpha^2 \Gamma g^2 \delta_1^2 + 2 \sum_{p=1}^{N} (1 + 2\Gamma + \alpha^2 \Gamma g^2 \delta_1^2) \right)$, $\Gamma, \Gamma_g, \Gamma_p \geq 0$ being respectively the supremum of $f, \|\partial A^{(p)} f(G, \{A^{(m)}\})\|_F, \|\partial G f(G, \{A^{(m)}\})\|_F$ on the compact set $D_g \times D_1 \times \ldots \times D_N$

This result proves that Singleshot converges ergodically to a point where the gradient is equal to 0 at the rate $O\left(\frac{1}{\sqrt{K}}\right)$.

### 4.2 Convergence result for Singleshotinexact

Let us consider that $f_j(A^{(N)}) \triangleq \frac{1}{2} \|X_j - G_{k+1} \times_{m \in \mathcal{K}_n} A^{(m)}_{k+1} \times_{n \in \mathcal{N}} A^{(n)}_{k+1} \times_{g \in \mathcal{G}_N} A^{(g)}_{k+1} \times_{q \in \mathcal{Q}_N} A^{(q)}_{k+1} \times_{N} A^{(N)}\|_F^2$

and that the step $\eta^N_k$ for $A^{(N)}$ is defined by the following minimization problem:

$$\eta^N_k = \arg \min_{\eta \in [\frac{\delta_1}{\sqrt{K}}, \frac{\delta_2}{\sqrt{K}}]} \left( \eta - \frac{1}{4K^2} \right) \phi(\eta)$$

(10)

$$\phi(\eta) = f \left( G_{k+1}, \{A^{(m)}_{k+1}\}^{N-1}_{1}, A^{(N)}_k \right) - f \left( G_k, \{A^{(m)}_k\}^N_{1} \right) + \lambda f \left( G_{k+1}, \{A^{(m)}_{k+1}\}^{N-1}_{1}, \phi(\eta) \right)$$
and \( \tilde{\phi}(\eta) = A_k^{(N)} - \frac{\eta}{\eta_k} \sum_{j \in S \mathcal{E} T_k} \partial A_k^{(N)} \ell_j \), \( \partial A_k^{(N)} \ell_j \) being the derivative of \( \ell_j \) evaluated at \( A_k^{(N)} \).

The parameters \( \lambda > 0, \gamma > 1 \) represent user-defined parameters. In addition to Assumption 1 and Assumption 2, we consider the following three additional assumptions:

1. **Descent step related to the update of** \( A_k^{(N)} \)

We assume that \( \frac{1}{\lambda} < \eta_k \leq \frac{1}{\lambda \gamma^2} \), which means that the solution of problem (10) is not attained at the lower bound of the interval.

2. **Non-vanishing gradient with respect to** \( A_k^{(N)} \)

\[ \partial A_k^{(N)} f \left( G_k^{(1)}, \left\{ A_k^{(m)} \right\}_{1}^{N-1}, A_k^{(N)} \right) \neq 0; \] this condition ensures the existence of a set \( \mathcal{S} \mathcal{E} T_k \) such that \( \sum_{j \in \mathcal{S} \mathcal{E} T_k} \partial A_k^{(N)} \ell_j \neq 0 \) and the set considered for the problem (10) is one of such sets.

3. **Choice of the number of sub-tensors** \( B_k \)

\[ I_n \times \sqrt{\frac{1}{2} \frac{1}{\Delta_k}} \leq B_k, I_n > 2; \] the condition \( I_n > 2 \) ensures \( I_n \sqrt{\frac{1}{2} \frac{1}{\Delta_k}} < I_n \).

With these assumptions at hand, the sequence \( \Delta_k = f \left( G_k^{(1)}, \left\{ A_k^{(m)} \right\}_{1}^{N} \right) - f_{\text{min}} \) verifies:

\[ \forall k > k_0 = 1 + \frac{1}{\log(1 + \lambda)} \log \left( \frac{1}{\log(1 + \lambda)} \right), \Delta_k \leq \frac{\Delta_1 + \zeta(\lambda, \rho, \alpha, I_n)}{k - k_0} \]

(11)

with \( \log \) being the logarithmic function, \( f_{\text{min}} \) representing the minimizer of \( f \), a continuous function defined on the compact set \( \mathcal{S} \mathcal{E} T_k \) and \( \zeta \) a function of \( \lambda, \rho, \alpha, I_n \). The parameter \( k_0 \) is well-defined since \( \lambda > 0 \). This result ensures that the sequence \( \left\{ G_k, A_k^{(1)}, .., A_k^{(N)} \right\} \) generated by **Singleshotinexact** converges to the set of minimizers of \( f \) at the rate \( O \left( \frac{1}{k} \right) \)

**Remark 2.** The problems defined by the equations (7) and (10), which solutions are global and can be solved by simple heuristics (e.g. **Golden section**), are not in contradiction with our approach since they can be solved by processing a single sub-tensor at a time due to the expression of \( f \) given by (2).

5 **Numerical experiments**

Our goal here is to illustrate that for small tensors, our algorithms **Singleshot** and **Singleshotinexact** and their positive variants, are comparable to some state-the-art decomposition methods. Then as the tensor size grows, we show that they are the only ones that are scalable. The competitors we have considered include SVD-based iterative algorithms [41] (denoted **GreedyHOOI**), a very recent alternate minimization approach based on sketching [44] (named **Tensorsketch** and randomization-based methods [48] (Algorithm 2 in [48] named **Sclera** and Algorithm 1 in [48] with positivity constraints named **posSclera**). Other materials related to the numerical experiments are provided in the supplementary material in the sections 4.1, 4.2, 4.3, 4.4.

The experiments are performed on the **Movielens** dataset [15], from which we construct a 3-order tensor whose modes represent test-samples, movies and users and on the **Enron** email dataset, from which a 3-order tensor is constructed, the first and second modes representing the sender and recipients of the emails and the third one denoting the most frequent words used in the miscellaneous emails. For Movielens, we set up a recommender system for which we report a mean average precision (MAP) obtained over a test set (averaged over five 50 – 50 train-test random splits) and for Enron, we report an error (AE) on a test set (with the same size as the training one) for a regression problem. As our goal is to analyze the scalability of the different methods as the tensor to decompose grows, we have arbitrarily set the size of the Movielens and Enron tensors to \( M \times M \times 200 \) and \( M \times M \times 610 \), \( M \) being a user-defined parameter. Experiments have been run on MacOzs with 32 Gb of memory.

Another important objective is to prove the robustness of our approach with respect to the assumptions and the definitions related to the descent steps laid out in the section 4 which is of primary importance since the minimization problems defining these steps can be time-consuming in practice for large tensors. This motivates the fact that for our algorithms, the descent steps are fixed in advance. For **Singleshot**, the steps are fixed to \( 10^{-6} \). For **Singleshotinexact**, the steps are respectively fixed to \( 10^{-6} \) and \( 10^{-8} \) for **Enron** and **Movielens**. Regarding the computing of the inexact gradient for **Singleshotinexact**, the elements in \( \mathcal{S} \mathcal{E} T_k \) are generated randomly without replacement with the same cardinality for any \( k \). The number of slices is chosen according to the third assumption in section 4.2.

For the charts, the unconstrained versions of **Singleshot** and **Singleshotinexact** will be followed by the term “unconstrained” and “positive” for the positive variants.
Figure 1: Approximation error and running time for the unconstrained decompositions. From left to right: first and second figures represent Movielens, third and fourth represent Enron. As $M$ grows, missing markers for a given algorithm means that it ran out of memory. The core $G$ rank is $(5, 5, 5)$.

Figure 2: Approximation error and running time for the non-negative decompositions. From left to right: first and second figures represent Movielens, third and fourth represent Enron. As $M$ grows, missing markers, for a given algorithm means that it ran out of memory. The core $G$ rank is $(5, 5, 5)$.

Figure 1 presents the results we obtain for these two datasets. At first, we can note that performance, in terms of MAP or AE, are rather equivalent for the different methods. Regarding the running time, the Scalrandomtucker is the best performing algorithm being an order of magnitude more efficient than other approaches. However, all competing methods struggle to decompose tensors with dimension $M = 4000$ and $M \geq 2800$ respectively for Enron and Movielens due to memory error. Instead, our Singleshot methods are still able to decompose those tensors, although the running time is large. As expected, Singleshotinexact is more computationally efficient than Singleshot.

Figure 2 displays the approximation error and the running time for Singleshot and singleshotinexact with positivity constraints and a randomized decomposition approach with non-negativity constraints denoted here as PosScalrandomtucker for Enron and Movielens. Quality of the decomposition is in favor of Singleshotpositive for both Movielens and Enron. In addition, when the tensor size is small enough, PosScalrandomtucker is very computationally efficient, being one order of magnitude faster than our Singleshot approaches on Enron. However, PosScalrandomtucker is not able to decompose very large tensors and ran out of memory contrarily to Singleshot.

For illustrating the online capability of our algorithm, we have considered a tensor of size $20000 \times 2000 \times 50$ constructed from Enron which is artificially divided into slices drawn with respect to the first and the second modes. The core rank is $(R, R, R)$. We compare the online variant of our approach associated to positivity constraints named Singleshotonline to the online version of Tensorsketch denoted Tensorsketchonline. Figure 3 shows running time for both algorithms. While of equivalent performance (see section 4.3.1 in the supplementary for the approximation error (AE) on a test set), our method is faster as our proposed update schemes, based on one single step of gradient descent, are more computationally efficient than a full alternate minimization.

6 Conclusion

We have introduced two new algorithms for scalable Tucker decomposition for which we established a convergence rate. Besides, we have proposed a new approach for a problem that has drawn little attention so far, that is, the Tucker decomposition under the single pass constraint (with no need to resort to the past data) of a tensor streaming with respect to every mode. In future works, we aim at applying the principle of Singleshot to other decomposition problems different from Tucker.
References


[34] Nicholas D. Sidiropoulos, Evangelos E. Papalexakis, and Christos Faloutsos. Parallel randomly compressed cubes (parcomp): A scalable distributed architecture for big tensor decomposition. 2014.


