Distributed Large-Scale Tensor Decomposition
André L. F. de Almeida, Alain Y. Kibangou

To cite this version:
André L. F. de Almeida, Alain Y. Kibangou. Distributed Large-Scale Tensor Decomposition. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2014), May 2014, Florence, Italy. hal-00958642

HAL Id: hal-00958642
https://hal.archives-ouvertes.fr/hal-00958642
Submitted on 13 Mar 2014

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.
DISTRIBUTED LARGE-SCALE TENSOR DECOMPOSITION

André L. F. de Almeida(1), Alain Y. Kibangou(2)

(1) Department of Teleinformatics Engineering, Federal University of Ceará, Fortaleza, Brazil
(2) GIPSA-Lab, University Joseph Fourier, CNRS, Saint Martin d’Hères, France
E-mails: andre.almeida@ieee.org, alain.kibangou@ujf-grenoble.fr

ABSTRACT

Canonical Polyadic Decomposition (CPD), also known as PARAFAC, is a useful tool for tensor factorization. It has found application in several domains including signal processing and data mining. With the deluge of data faced in our societies, large-scale matrix and tensor factorizations become a crucial issue. Few works have been devoted to large-scale tensor factorizations. In this paper, we introduce a fully distributed method to compute the CPD of a large-scale data tensor across a network of machines with limited computation resources. The proposed approach is based on collaboration between the machines in the network across the three modes of the data tensor. Such a multi-modal collaboration allows an essentially unique reconstruction of the factor matrices in an efficient way. We provide an analysis of the computation and communication cost of the proposed scheme and address the problem of minimizing communication costs while maximizing the use of available computation resources.

Index Terms— Tensor decompositions, large-scale data, distributed computation.

1. INTRODUCTION

From Internet to large research infrastructures, the volume of data generated by our societies is continuously increasing. A deluge faced by the producers of these data as well as their users. The big data issue is a significant scientific challenge that requires deep investigations in both engineering and fundamental science. Everyone is concerned and it is urgent to get answers to questions such as how to store these huge amount of data? How to process and analyze them?

Low-rank matrix factorization has received a particular attention in recent years, since it is fundamental to a variety of mining tasks that are increasingly being applied to massive datasets. In large applications, matrix factorizations can involve matrices with billions of entries. At this massive scale, distributed algorithms for matrix factorization are essential to achieve reasonable performance [2]. However, in many disciplines, data inherently has more than two axes of variation and can be arranged as tensors (i.e. multi-way arrays). Computing tensor decompositions of multi-way datasets is particularly useful to extract hidden patterns and structure in data analytics problems. Specifically, CPD (Canonical Polyadic Decomposition) also known as PARAFAC (Parallel factor analysis) is an extension of a low rank matrix decomposition to tensors.

In order to compute CPD, several algorithms have been proposed in the literature, which can be classified into three main categories: alternating algorithms, derivative based algorithms, and non-iterative algorithms (see e.g. [3, 4, 5]). Recently, considering that the data tensor can be serially acquired or the underlying process can be time-varying, adaptive algorithms have been proposed in [6]. And more recently, assuming that the data tensor is spatially spread out, distributed algorithms have been introduced in [7] and [8]. These algorithms are based on an alternating least square (ALS) algorithm that proceeds iteratively and minimizes a criterion (that is usually quadratic) with respect to individual factors one by one. Then, distributed algorithms are obtained using the concept of average consensus that has been extensively studied in computer science and is a central topic for load balancing (with divisible tasks) in parallel computers [9].

The need for large-scale tensor computations is increasing and there is a huge gap to be filled. In contrast to large-scale matrix factorization, very few works are devoted to large-scale tensors. Two ways have been recently considered in the literature. The first one consists in exploiting sparseness of tensors. In [10], the GigaTensor algorithm is designed in order to minimize the number of floating point operations and to handle the size of intermediate data in order to overcome the intermediate data explosion problem. For large-scale tensors the intermediate data explosion problem arises when Khatri-Rao matrix products are implemented in a naive way. In [10] a smart ordering of computations is proposed and explicit computation of Khatri-Rao matrix product is avoided. In addition, a way to implement the proposed scheme in the distributed computing frameworks MapReduce and Hadoop is proposed by the authors and then, from extensive simulations, linear scalability on the number of machines is claimed. Note that the data explosion problem can also be handled by exploiting sparseness of both data and latent factors as in [11, 12, 13].

The second class of methods consists in a subdivision of the large-scale tensor into smaller ones. Then factor matrices are reconstructed from the estimated sub-factors. The work in [14] is motivated by the success of random sampling-based matrix algorithms. The large-scale tensor is under-sampled several times, then the different sub-tensors are processed in parallel and eventually the results are combined in a clever way. In order to overcome permutation indeterminacies, all the different sub-tensors are enforced to overlap in a common set of indices in all the three modes of the tensor. The merging operation is guaranteed to be successful only if the random samples (sub-tensors) fulfill CPD identifiability conditions. In [1], sub-division of the original tensor is achieved in a deterministic way; a grid decomposition is proposed. CPD is first performed for each sub-tensor in order to get a an alternative representation that allows avoiding Khatri-Rao matrix products. Then, grid-CPD performs as a decentralized computation method where at each step parallel machines have to communicate with a central server. There is no direct collaboration between the involved machines. In [15], the authors have introduced the notion

This work is partially supported by the CNPq and the FUNCAP-INRIA project TeMP.
of collaboration in the grid. In this set-up, the computation is really distributed. The central server only initiates the process by sending data to the machines in the network and merges the results at the end of the computation process. During the computation process, the central server can allocate its own resources to other tasks. Moreover, thanks to this full collaboration and by introducing overlapping between sub-tensors, uniqueness of the decomposition with indeterminacies as in the standard CPD can be guaranteed. However, collaboration was restricted to two modes and overlapping induced loss of efficiency (increase of the dimensions and the number of sub-tensors).

This paper introduces a new way for computing a grid-CPD of a large-scale data tensor that ensures essential uniqueness and perfect reconstruction of the factor matrices. The main idea of the proposed approach is to allow collaboration between the machines in the network across the three modes of the tensor. To cope with the multi-way nature of the data, multi-graphs are used to model the network. Due to direct collaboration between machines, fulfilling CPD uniqueness conditions is not required for each sub-tensor. We provide an analysis of the computation and communication cost of the proposed scheme and address the problem of minimizing communication costs while maximizing the use of available computation resources.

Notations: Vectors are written as boldface lower-case letters \((a, b, \ldots)\), matrices as boldface upper-case letters \((A, B, \ldots)\), and tensor as calligraphic letters \((X, Y, \ldots)\). \(A^T\) stands for the transpose of \(A\) whereas \(A^H\) stands for its complex conjugate. The operator \(\circ\) denotes the outer product between vectors, while \(\odot\) stands for the Khatri-Rao (columnwise Kronecker) product. The Hadamard (element-wise) product is denoted by the \(\ast\) symbol.

### 2. PROBLEM SETTING

Consider a large-scale tensor \(X \in \mathbb{C}^{I \times J \times K}\), i.e. a tensor with a total number of entries \(IJK\) that can easily be of order of billions. The CPD of \(X\), denoted by \(\mathcal{X} = [A, B, C]\), is given by [16, 17]:

\[
\mathcal{X} = \sum_{r=1}^{R} a_r \circ b_r \circ c_r ,
\]

where \(A = [a_1, \ldots, a_R] \in \mathbb{C}^{I \times R}\), \(B = [b_1, \ldots, b_R] \in \mathbb{C}^{J \times R}\), and \(C = [c_1, \ldots, c_R] \in \mathbb{C}^{K \times R}\) are the factor matrices associated with the CPD of \(X\), and \(R\) denotes the tensor rank. A sufficient condition for the essential uniqueness of the CPD was first established by Kruskal in [18], and states that \(A, B\) and \(C\) can be uniquely estimated (up to column permutation and scaling) if \(k_A + k_B + k_C \geq 2R + 2\), where \(k_\cdot\) denotes the Kruskal-rank, or “\(k\)-rank”, of its matrix argument. Shorty, \(k_\cdot\) is equal to \(r\) if every set of \(r\) columns of \(A\) is linearly independent. In what follows, we work under the assumption that CPD uniqueness for \(X\) holds. Usually, the matrices \(A, B, C\) and \(C\) can be estimated by computing an approximation to the rank-\(R\) CPD of \(X\), i.e.:

\[
\min_{A, B, C} \left\| \mathcal{X} - \sum_{r=1}^{R} a_r \circ b_r \circ c_r \right\|_F^2 .
\]

The traditional way for carrying out such a computation is to resort to an alternating least squares algorithm (ALS) where estimations of \(A, B,\) and \(C\) are obtained by alternatively solving three different linear least squares (LS) problems. Despite its conceptual simplicity, ALS involved Khatri-Rao matrix products and multiplications of matrices that cannot be handled when the tensor exhibits large dimensions since computation and storage resource can easily become insufficient. Apart from complexity issues, ALS-based algorithms present a very slow convergence due to the huge number of unknowns involved in the estimation of the large factor matrices \(A, B\) and \(C\).

Consider a central server having at its disposal a large-scale data tensor \(X\) to be factorized. This server is connected to a dense network of \(L\) machines (cores or independent computers) with limited processing powers. Our goal is to perform the CPD of \(X\) across the network by using all the in-network computation resources in an optimal way while making the central server available to other tasks. The central server and the \(L\) machines, where computations are achieved, are nodes of a network connected through reliable high speed data links, so that data exchanges may be considered noise-free. The central server first subdivides the tensor and assigns each sub-tensor to a given machine. Once computation is achieved, the \(L\) machines send their results back to the central server where global factor matrices are built. In contrast to the grid-CPD in [1], the central server is not involved in the computation process and available to perform other tasks at the same time. The proposed set-up is therefore fully distributed while that of [1] is simply decentralized.

### 3. PROPOSED SOLUTION

The central server generates different (possibly overlapped) data sub-tensors \(X^{(I_1, I_2, I_3)} \in \mathbb{C}^{I_1 \times I_2 \times K_{I_3}}, I_1 = 1, \ldots, L_1, I_2 = 1, \ldots, L_2, I_3 = 1, \ldots, L_3,\) of much smaller dimensions, by partitioning \(X\) in a dense 3-D tensor grid. The dimensions of the sub-tensors are related as follows: \(I_1 + \ldots + I_{L_1} \geq I, J_1 + \ldots + J_{L_2} \geq J, K_1 + \ldots + K_{L_3} \geq K.\) Equality in a given mode occurs if and only if the sub-tensors do not overlap in that mode. The sub-tensors \(X^{(I_1, I_2, I_3)}\) can be concatenated along different pair of modes to form mode-1, mode-2, and mode-3 sub-tensors \(X^{(I_1)} \in \mathbb{C}^{I_1 \times J \times K}, X^{(I_2)} \in \mathbb{C}^{I \times I_2 \times K}, X^{(I_3)} \in \mathbb{C}^{I \times J \times K_{I_3}},\) which are defined, respectively, as

\[
X^{(I_1)} = [X^{(I_1, I_2, I_3)}]_{I_1}, \quad X^{(I_2)} = [X^{(I_1, I_2, I_3)}]_{I_2}, \quad X^{(I_3)} = [X^{(I_1, I_2, I_3)}]_{I_3} .
\]

An interesting fact resulting from the CPD trilinearity is as follows: if the CPD of \(X\) is essentially unique and provided that \(k_{A(I_i)} = k_A, k_{B(I_i)} = k_B, k_{C(I_i)} = k_C,\) each mode-\(i\) sub-tensor, \(i = 1, 2, 3,\) admits an exact CPD given by:

\[
X^{(I_i)} = [A^{(I_i)}, B^{(I_i)}, C^{(I_i)}] .
\]

Now, the central server assigns a sub-tensor \(X^{(I_1, I_2, I_3)}\) to each of the \(L\) available machines. To guarantee a one-to-one
mapping between the sub-tensors and the machines, each machine is uniquely labeled by the triad \((\ell_1, \ell_2, \ell_3)\) and is associated with the corresponding sub-tensor \(\chi^{(\ell_1, \ell_2, \ell_3)}\). Therefore the total number \(L\) of machines is equal to \(L_1 L_2 L_3\). According to (6), from their respective sub-tensors, the machines having the same \(i\)-th coordinate, \(i = 1, 2, 3\), can collaborate to estimate their common sub-factor matrix in mode-\(i\). Therefore the adjacency relations between the machines evolve according to the mode of interest.

To capture this behavior, we adopt multi-layer graphs, or simply multi-graphs, for representing the network [19, 20]. For this purpose, let \(G\) be a multi-layer graph which contains 3 individual graph layers \(G^{(i)}, i = 1, 2, 3\), where each layer \(G^{(i)}\) is an undirected graph consisting of a common vertex set \(V\), with cardinality \(L_i\), and a specific edge set \(E^{(i)}\). Each layer is associated with a mode of the tensor to be decomposed. In addition, each layer is constituted with \(L_i\) connected components of \(L_i/L_i\) vertices, each connected component being a star graph. More precisely, the nodes labeled \((l_1, l_2, l_3)\) and \((l'_1, l'_2, l'_3)\) belongs to the same component of \(G^{(i)}\) if and only if \(l_i = l'_i\). They are adjacent only if one of them is an internal node and the other is a leaf of a tree. Therefore, each layer can be viewed as a set of parallel star sub-networks, each star sub-network constituting a connected component of the graph. As it is common for tensor decomposition, data processing is to be carried out in an alternating way. The layers of the multi-graph are to be considered one after the other. In the time, the network topology switches periodically between parallel star subnetworks (see Fig. 1).

![Fig. 1](image)

By exploiting the degrees of freedom provided by the three layers \(G^{(i)}, i = 1, 2, 3\), of the multi-graph \(G\), the problem consists in finding \(A^{(\ell_1)}\), \(A^{(\ell_2)}\), \(A^{(\ell_3)}\), \(B^{(\ell_2)}\), and \(C^{(\ell_3)}\), \(\ell_3 = 1, \ldots, L_3\) by solving the three following sets of LS problems:

\[
\begin{align*}
\min_{A^{(\ell_1)}} \& \|X^{(\ell_1)} - [A^{(\ell_1)}, B, C]\|_F, \ & \ell_1 = 1, \ldots, L_1, \\
\min_{B^{(\ell_2)}} \& \|X^{(\ell_2)} - [A, B^{(\ell_2)}, C]\|_F, \ & \ell_2 = 1, \ldots, L_2, \\
\min_{C^{(\ell_3)}} \& \|X^{(\ell_3)} - [A, B, C^{(\ell_3)}]\|_F, \ & \ell_3 = 1, \ldots, L_3.
\end{align*}
\]

Each set of LS problems is associated with disjoint subsets of machines and therefore can be solved in parallel. For instance, the \(L_1\) LS problems in mode-1 are solved independently by \(L_1\) subsets of machines working in parallel. The same applies to the two other

The solution that minimizes \(J_{A^{(\ell_1)}}\), \(J_{B^{(\ell_2)}}\), and \(J_{C^{(\ell_3)}}\) can then be written as:

\[
\begin{align*}
A^{(\ell_1)}T &= \left( \frac{1}{L_2 L_3} \sum_{\ell_2, \ell_3} \Gamma^{(\ell_2, \ell_3)} A^{(\ell_1)} \right) \left( \frac{1}{L_2 L_3} \sum_{\ell_2, \ell_3} \Psi^{(\ell_2, \ell_3)} A^{(\ell_1)} \right)^{-1} \\
B^{(\ell_2)}T &= \left( \frac{1}{L_1 L_3} \sum_{\ell_3} \Gamma^{(\ell_2, \ell_3)} B^{(\ell_2)} \right) \left( \frac{1}{L_1 L_3} \sum_{\ell_3} \Psi^{(\ell_2, \ell_3)} B^{(\ell_2)} \right)^{-1} \\
C^{(\ell_3)}T &= \left( \frac{1}{L_1 L_2} \sum_{\ell_2} \Gamma^{(\ell_2, \ell_3)} C^{(\ell_3)} \right) \left( \frac{1}{L_1 L_2} \sum_{\ell_2} \Psi^{(\ell_2, \ell_3)} C^{(\ell_3)} \right)^{-1}
\end{align*}
\]

where \(\Gamma^{(\ell_2, \ell_3)} = B^{(\ell_2)}B^{(\ell_2)}C^{(\ell_2)}H C^{(\ell_3)}\in C^{R \times R}\), \(\Gamma^{(\ell_2, \ell_3)} = C^{(\ell_2)}H C^{(\ell_2)}H A^{(\ell_1)}\in C^{R \times R}\), \(\Gamma^{(\ell_2, \ell_3)} = A^{(\ell_1)}H A^{(\ell_1)}H B^{(\ell_2)}\in C^{R \times R}\), \(\Gamma^{(\ell_2, \ell_3)} = B^{(\ell_2)}H X^{(\ell_2, \ell_3)}\in C^{R \times R}\), \(\Gamma^{(\ell_2, \ell_3)} = X^{(\ell_2, \ell_3)}H X^{(\ell_2, \ell_3)}\in C^{R \times R}\).

The estimation of \(\{A^{(1)}, \ldots, A^{(L_1)}\}\), \(\{B^{(1)}, \ldots, B^{(L_2)}\}\), and \(\{C^{(1)}, \ldots, C^{(L_3)}\}\) is carried out in three steps. First, the \(L_1\) connected components of \(G^{(1)}\) operate in parallel to estimate \(A^{(1)}, \ldots, A^{(L_1)}\). Second, the \(L_2\) connected components of \(G^{(2)}\) operate in parallel to estimate \(B^{(1)}, \ldots, B^{(L_2)}\). Finally, the \(L_3\) connected components of \(G^{(3)}\) operate in parallel to estimate \(C^{(1)}, \ldots, C^{(L_3)}\). Thanks to the star topology adopted herein, the averaging operations involved in (10), (11), and (12) can be carried
Computation cost
950
Sub-tensor size
(90
(90
100
100
C
Ψ
334
Communication cost
3
-1468
×
Table 1 gives the optimal sub-tensor size for different numbers of available machines for a 103 \times 103 \times 103 tensor (C_m= communication cost per machine; R= rate of machines used).

<table>
<thead>
<tr>
<th>Machines</th>
<th>Sub-tensor size</th>
<th>(I_1, J_2, L_3)</th>
<th>C_m</th>
<th>R (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>500 \times 334 \times 334</td>
<td>(2, 3, 3)</td>
<td>1468</td>
<td>90</td>
</tr>
<tr>
<td>40</td>
<td>334 \times 334 \times 250</td>
<td>(3, 4, 5)</td>
<td>1218</td>
<td>90</td>
</tr>
<tr>
<td>60</td>
<td>334 \times 250 \times 200</td>
<td>(3, 4, 5)</td>
<td>1084</td>
<td>100</td>
</tr>
<tr>
<td>80</td>
<td>250 \times 250 \times 200</td>
<td>(4, 4, 5)</td>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>250 \times 250 \times 200</td>
<td>(4, 4, 5)</td>
<td>950</td>
<td>100</td>
</tr>
</tbody>
</table>

Note that as the number of available machines increases a more efficient use of the resources is obtained and the communication cost per node is reduced, corroborating the relevance of optimizing the sampling of the data tensor across the three modes.

4. CONCLUSION

Performing CPD for large-scale tensors is particularly challenging. In this paper, we have proposed a fully distributed method that allows computing CPD across a network of machines with limited computation resources. In the proposed scheme, the central server does not interfere in the computation process and can achieve other tasks at the same time. The machines in the network collaborate with each other according to the three modes of the tensor. Such a multi-modal collaboration is captured by a multi-graph whose layers have several connected components. Due to this full collaboration, improved necessary identifiability conditions are achieved and the global factor matrices can be reconstructed in an efficient way. However, a communication overhead is introduced. Hopefully, the communication cost grows linearly with the number of machines and the dimensions of the sub-tensors. Additionally, optimizing the sampling of the tensor across the three modes yields a significant reduction of this overhead while maximizing the use of the available computation resources. Extension to tensors of higher orders is straightforward although the proposed optimization problem becomes more challenging. Future works include resources optimization for heterogeneous networks of machines having different computation capabilities, extension to other tensor factorization, and exploitation of sparsity and structural properties of the factor matrices.
5. REFERENCES


