BLOCK-RANDOMIZED STOCHASTIC PROXIMAL GRADIENT FOR CONSTRAINED LOW-RANK TENSOR FACTORIZATION

Xiao Fu[†]*, Cheng Gao*[†]*, Hoi-To Wai*[⋆]*, and Kejun Huang*[∗]

[†]School of EECS, Oregon State University, Corvallis, OR, USA
[⋆]Department of SEEM, The Chinese University of Hong Kong, NT, Hong Kong
[∗] Department of CISE, University of Florida, Gainesville, FL, USA

ABSTRACT

This work focuses on *canonical polyadic decomposition* (CPD) for large-scale tensors. Many prior works rely on data sparsity to develop scalable CPD algorithms, which are not suitable for handling dense tensor, while dense tensors often arise in applications such as image and video processing. As an alternative, stochastic algorithms utilize data sampling to reduce per-iteration complexity and thus are very scalable, even when handling dense tensors. However, existing stochastic CPD algorithms are facing some challenges. For example, some algorithms are based on randomly sampled tensor entries, and thus each iteration can only updates a small portion of the latent factors. This may result in slow improvement of the estimation accuracy of the latent factors. In addition, the convergence properties of many stochastic CPD algorithms are unclear, perhaps because CPD poses a hard nonconvex problem and is challenging for analysis under stochastic settings. In this work, we propose a stochastic optimization strategy that can effectively circumvent the above challenges. The proposed algorithm updates a whole latent factor at each iteration using sampled fibers of a tensor, which can quickly increase the estimation accuracy. The algorithm is flexible—many commonly used regularizers and constraints can be easily incorporated in the computational framework. The algorithm is also backed by a rigorous convergence theory. Simulations on large-scale dense tensors are employed to showcase the effectiveness of the algorithm.

Index Terms— Canonical polyadic decomposition, PARAFAC, stochastic optimization

1. INTRODUCTION

Canonical polyadic decomposition (CPD) is arguably the most popular low-rank tensor factorization model. CPD has become a workhorse for tensor data analytics in many fields, such as computational chemistry, social network mining, computer vision, topic modeling, and hidden Markov model identification, just to name a few [1–3]. Computing the CPD, however, poses a very hard optimization problem. Many algorithms have been proposed through the years, e.g., those in [4–6].

In the era of *big data*, one critical challenge is how to scale up tensor factorization algorithms to keep pace with the overwhelmingly large volume of data. A number of scalable tensor factorization methods have been proposed in the literature [4, 5, 7, 8]. Many of these methods are specialized to handle big *sparse* data—the major insights there are to judiciously utilize zero elements in the data to compute key operations in the classical alternating least squares (ALS) algorithm such as the Khatri-Rao product-matrix multiplication. Another pressing challenges is to handle big *dense* tensors, which frequently arise in timely applications such as medical imaging, remote sensing, and computer vision. Under such circumstances, the sparsity-enabled efficient algorithms [4, 5, 7–9] are no longer scalable. In fact, since big dense tensors typically cost a lot of memory (e.g., a dense tensor with a size of $2,000 \times 2,000 \times 2,000$ occupies 57.52GB memory if saved as double-precision numbers), it is even hard to load them in RAM of desktops or even servers. Methods that can effectively handle such tensors with lightweight updates and small footprint in memory are highly desired.

Stochastic sampling/sketching based methods are suitable for handling large and dense tensors. These methods work in an iterative manner. In each iteration, a small portion of the tensor entries are sampled from the large tensor, and the latent factors are updated using information extracted from the sampled piece—thereby naturally admitting low per-iteration computational and memory complexities. For example, [10, 11] both propose algorithms under this framework, which scale very well. The challenge here is that every tensor entry only contains information of a certain row of the latent factors, and updating the entire latent factors may need a lot of iterations. This may lead to slow improvement of the latent factor estimation accuracy. A recent work in [12] takes a different sampling strategy—by sampling tensor fibers, an algorithm that ensures updating one entire latent factor in every iteration is proposed. However, the algorithm in [12] works with at least as many fibers as the tensor rank, which in some cases gives rise to much higher per-iteration complexity relative to the algorithms in [10, 11]. In addition, the algorithm in [12] cannot handle constraints or regularizations on the latent factors, which are important considerations in practice. Another challenge is that convergence properties of sampling based algorithms such as those in [10, 12] are often unclear.

In this work, we propose a new stochastic algorithm for computing the CPD of large-scale dense tensors. Our sampling strategy is similar to that in [12]—we sample tensor fibers in each iteration. Our contributions consist of a number of key differences relative to the prior work in [12]. First, unlike the method in [12], our method allows the number of sampled fibers to be much fewer than that of the tensor rank—which substantially reduces the per-iteration complexity. Second, the proposed method can easily handle a series of regularizations and constraints (e.g., nonnegativity and sparsity) that are frequently used in practice. Third, by judiciously combining the sampling strategy with a randomized block variable updating rule, we show that the algorithm converges to a stationary point of the problem of interest. Simulation results using large and dense tensors show that the proposed method is very promising.

This work was supported in part by National Science Foundation under Project NSF ECCS-1808159

2. PRELIMINARIES

An Nth order tensor is a multiway array whose elements are indexed by N indices, i.e., $\underline{\boldsymbol{X}}(i_1, \ldots, i_N)$ denotes one element of the tensor $\underline{\boldsymbol{X}}$ with a size of $I_1 \times I_2 \times \ldots \times I_N$. Our interest lies in the CPD of an Nth order tensor, which refers to the following rank-F parametrization of a tensor. Let $\boldsymbol{A}_{(n)}$ be an $I_n \times F$ matrix,

$$\underline{\boldsymbol{X}} = \sum_{f=1}^{F} \boldsymbol{A}_{(1)}(:, f) \circ \boldsymbol{A}_{(2)}(:, f) \circ \ldots \circ \boldsymbol{A}_{(N)}(:, f), \quad (1)$$

where "$\circ$" denotes the outer product of vectors. An other representation for CPD is

$$\underline{\boldsymbol{X}}(i_1, \ldots, i_N) = \sum_{f=1}^{F} \prod_{n=1}^{N} \boldsymbol{A}_{(n)}(i_n, f) \quad (2)$$

for $i_n \in \{1, \ldots, I_n\}$. When F is the minimal integer that satisfies the expression in (1), the right hand side in (1) is called the *canonical* polyadic decomposition of the tensor $\underline{\boldsymbol{X}}$. The CPD of a tensor is essentially unique (meaning that the latent factors $\boldsymbol{A}_{(n)}$'s that constitute the data $\underline{\boldsymbol{X}}$ are unique up to some trivial ambiguities [1]). This interesting and important property has enabled a plethora of applications—also see [1] for a recent overview. CPD of a tensor is often realized by optimizing a certain decomposition criterion, e.g.,

$$\underset{\{\boldsymbol{A}_{(n)}\}_{n=1}^{N}}{\text{minimize}} \; f(\{\boldsymbol{A}_{(n)}\}) = \Big\| \underline{\boldsymbol{X}} - \sum_{f=1}^{F} \boldsymbol{A}_{(1)}(:, f) \circ \ldots \circ \boldsymbol{A}_{(N)}(:, f) \Big\|_F^2. \quad (3)$$

2.1. Unfolding, ALS and Variants

An important operation in tensor algebra is the so-called matricization, or unfolding. The mode-n unfolding of a tensor [2], denoted by $\boldsymbol{X}_{(n)}$, is an $J_n \times I_n$ matrix where we have the relation $\underline{\boldsymbol{X}}(i_1, \ldots, i_N) = \boldsymbol{X}_{(n)}(j, i_n)$, and we have $j = 1 + \sum_{k=1, k\neq n}^{N} (i_k - 1) J_k$ and $J_k = \prod_{m=1, m\neq n}^{k-1} I_m$. Equivalently, the mode-n unfolding can be represented as

$$\boldsymbol{X}_{(n)} = \boldsymbol{H}_{(n)} \boldsymbol{A}_{(n)}^{\top}, \quad (4)$$

where the $J_n \times F$ matrix $\boldsymbol{H}_{(n)}$ is defined as $\boldsymbol{H}_{(n)} = \boldsymbol{A}_{(1)} \odot \boldsymbol{A}_{(n-1)} \odot \boldsymbol{A}_{(n+1)} \odot \ldots \odot \boldsymbol{A}_{(N)} = \odot_{i=1, i\neq n}^{N} \boldsymbol{A}_{(i)}$ and $\odot$ denotes the Khatri-Rao product.

The above unfolding representations have enabled the famous ALS algorithm [13], where one finds the CPD of a tensor by solving the following cyclically:

$$\boldsymbol{A}_{(n)} \leftarrow \arg\min_{\boldsymbol{A}} \; \big\| \boldsymbol{X}_{(n)} - \boldsymbol{H}_{(n)} \boldsymbol{A}^{\top} \big\|_F^2. \quad (5)$$

When the problem dimension is large (which often happens in applications such as medical imaging, remote sensing, and computer vision), solving the above problem can be computationally prohibitive. For example, the product $\boldsymbol{H}_{(n)}^{\top} \boldsymbol{X}_{(n)}$ that happens in every iteration of ALS costs $\mathcal{O}(\prod_{n=1}^{N} I_n F)$ flops. Exploiting sparsity in the tensor data, many works have considered fast algorithms for computing this product in judicious ways [7, 14]. Some recent methods combine first-order optimization and ALS [4, 5] to make the algorithms more flexible (in terms of incorporating constraints and regularizations) and scalable—but the complexity orders of those algorithms often scale similarly as that of ALS.

Fig. 1. From left to right: mode-$1, 2, 3$ tensor fibers of a third-order tensor, respectively.

2.2. Sampling/Sketching and Stochastic Optimization

When the tensor is large and dense, working with the entire dataset could be computationally and memory-wise expensive. A popular workaround is to apply *stochastic approximation* (SA)—i.e., sampling parts of the data at random and use the sampled piece to update the latent factors. By (2), Eq. (3) can be rewritten as

$$\underset{\{\boldsymbol{A}_{(n)}\}}{\text{minimize}} \; (1/T) \textstyle\sum_{i_1, \ldots, i_N} f_{i_1, \ldots, i_N} \left(\{\boldsymbol{A}_{(n)}\} \right), \quad (6)$$

where $T = \prod_{n=1}^{N} I_n$ and

$$f_{i_1, \ldots, i_N}(\{\boldsymbol{A}_{(n)}\}) = \Big(\underline{\boldsymbol{X}}(i_1, \ldots, i_N) - \sum_{f=1}^{F} \prod_{n=1}^{N} \boldsymbol{A}_{(n)}(i_n, f) \Big)^2.$$

The objective function in (6) can be understood as the empirical risk of SA. Using this observation, the algorithms in [10, 11] randomly sample a subset set of $\{(i_1, \ldots, i_N)\}$ and update the pertinent parts of the latent factors (note that the $(i_1, \ldots, i_N)$th entry of tensor contains the information of $\boldsymbol{A}_{(n)}(i_n, :)$ for $n = 1, \ldots, N$) using the sampled entries of the tensor. For example, [11] uses a stochastic gradient (SG) based approach and update $\boldsymbol{A}_{(n)}(i_n, :)$. The sampling method in [10] is similar, while the update is not gradient-based but Gauss-Newton or ALS applied to the sampled set of entries (or, subtensors, to be precise). The upshot of this line of work is that the per-iteration complexity can be quite low. The downside, however, is that in every iteration only a small part of the $\boldsymbol{A}_{(n)}$'s (i.e., some rows) are updated—which may result in slow improvement of estimation accuracy of the latent factors. In addition, the convergence properties of these algorithms are unclear.

An alternative [12] to the SA based methods above is to leverage the tensor data structure by considering randomly sampled *fibers* of tensors. Note that a mode-n fiber of $\underline{\boldsymbol{X}}$ (cf. Fig. 1) is a row of the mode-n unfolding $\boldsymbol{X}_{(n)}$ [2]. Now, assuming that one samples a set of mode-n fibers indexed by $\mathcal{F}_n \subset \{1, ..., J_n\}$, then $\boldsymbol{A}_{(n)}$ can be updated by solving a 'sketched version' of Problem (5):

$$\boldsymbol{A}_{(n)} \leftarrow \arg\min_{\boldsymbol{A}} \; \big\| \boldsymbol{X}_{(n)}(\mathcal{F}_n, :) - \boldsymbol{H}_{(n)}(\mathcal{F}_n, :) \boldsymbol{A}^{\top} \big\|_F^2, \quad (7)$$

If $|\mathcal{F}_n| \geq F$, then $\boldsymbol{H}_{(n)}(\mathcal{F}_n, :)$ can be invertible and the sketched system of linear equations $\boldsymbol{X}_{(n)}(\mathcal{F}_n, :) \approx \boldsymbol{H}_{(n)}(\mathcal{F}_n, :) \boldsymbol{A}_{(n)}^{\top}$ is over-determined. Hence, one can update $\boldsymbol{A}_{(n)}$ by solving the $|\mathcal{F}_n|$ dimensional linear system $\boldsymbol{A}_{(n)}^{\top} \leftarrow \boldsymbol{H}_{(n)}(\mathcal{F}_n, :)^{\dagger} \boldsymbol{X}_{(n)}(\mathcal{F}_n, :)$. Similar to the ALS algorithm, after updating $\boldsymbol{A}_{(n)}$, the algorithm moves to mode-$(n+1)$ fibers and repeats the same for updating $\boldsymbol{A}_{(n+1)}$. Intuitively, the method in [12] can be more efficient than SG based methods in terms of estimating the $\boldsymbol{A}_{(n)}$'s. The downside is that it needs to sample at least F fibers for each update, and F can be large. In addition, the update in (5) can only handle unconstrained/unregularized tensor decomposition, while incorporating constraints/regularizations is often critical in practice. Convergence properties of the method are also unclear.

3. PROPOSED ALGORITHM

In this work, we combine ALS and fiber sampling, but allow $|\mathcal{F}_n| \ll F$—which keeps the per-iteration complexity low. As we show below, the proposed algorithm can easily handle a variety of constraints and regularizations that are commonly used in signal processing and data analytics. In addition, we provide convergence analysis to back up the proposed algorithm.

3.1. Basic Idea

We propose to tackle (3) by combining SA and exploiting the tensor fiber structure. Instead of exactly solving the least squares subproblems (5) in each iteration, we update $\boldsymbol{A}_{(n)}$ using a doubly stochastic procedure—specifically, at iteration r, we first sample uniformly at random a mode index $n \in \{1, ..., N\}$, then we sample uniformly a set of mode-n fibers that is indexed by $\mathcal{F}_n \subset \{1, ..., J_n\}$, with $|\mathcal{F}_n| = \overline{F}$. Let $\boldsymbol{G}^{(r)} \in \mathbb{R}^{I_1 \times F} \times \cdots \times \mathbb{R}^{I_N \times F}$ be a collection of matrices, representing the stochastic gradient as:

$$\boldsymbol{G}_{(n)}^{(r)} = \frac{1}{|\mathcal{F}_n|}\Big(\boldsymbol{A}_{(n)}^{(r)}\boldsymbol{H}_{(n)}^{\top}(\mathcal{F}_n)\boldsymbol{H}_{(n)}(\mathcal{F}_n) - \boldsymbol{X}_{(n)}^{\top}(\mathcal{F}_n)\boldsymbol{H}_{(n)}(\mathcal{F}_n)\Big)$$
$$\boldsymbol{G}_{(n')}^{(r)} = \boldsymbol{0}, \;\; n' \neq n, \tag{8}$$

where $\boldsymbol{G}_{(n)}^{(r)}$ denotes the nth block of $\boldsymbol{G}^{(r)}$, and we used the shorthand notations $\boldsymbol{X}_{(n)}(\mathcal{F}_n) = \boldsymbol{X}_{(n)}(\mathcal{F}_n, :)$ and $\boldsymbol{H}_{(n)}(\mathcal{F}_n) = \boldsymbol{H}_{(n)}(\mathcal{F}_n, :)$ The latent variables are updated by

$$\boldsymbol{A}_{(n)}^{(r+1)} \leftarrow \boldsymbol{A}_{(n)}^{(r)} - \alpha^{(r)}\boldsymbol{G}_{(n)}^{(r)}, \;\; n = 1, ..., N. \tag{9}$$

One observation is that $\boldsymbol{G}_{(n)}^{(r)}$ is simply an SA applied to the full gradient of Problem (3) w.r.t. the chosen mode-n variable $\boldsymbol{A}_{(n)}$, and the update is an iteration of the classical SG algorithm (with a minibatch size $|\mathcal{F}_n|$) for solving the problem in (5).

The proposed update is very efficient, since the most resource-consuming update $\boldsymbol{H}_{(n)}^{\top}\boldsymbol{X}_{(n)}$ in algorithms such as those in [4, 5] is avoided. The corresponding part $\boldsymbol{X}_{(n)}^{\top}(\mathcal{F}_n, :)\boldsymbol{H}_{(n)}(\mathcal{F}_n, :)$ costs only $\mathcal{O}(|\mathcal{F}_n|FI_n)$ flops—and $|\mathcal{F}_n| = \overline{F}$ is under our control. Note that the first step in this procedure is different from standard ALS-type algorithms that update the block variables $\boldsymbol{A}_{(n)}$ cyclically instead of updating a randomly sampled block. As we show next, this modification will greatly simplify our convergence analysis.

3.2. Extension

In many applications the following regularized version of tensor decomposition is of interest:

$$\underset{\{\boldsymbol{A}_{(n)}\}_{n=1}^{N}}{\text{minimize}} \;\; f(\{\boldsymbol{A}_{(n)}\}) + \sum_{n=1}^{N} r_n(\boldsymbol{A}_{(n)}) \;\; \text{s.t.} \;\; \boldsymbol{A}_{(n)} \in \mathcal{A}_n, \tag{10}$$

where $f(\{\boldsymbol{A}_{(n)}\})$ is the objective function of (3), $r_n(\boldsymbol{A}_{(n)})$ denotes a structure-promoting regularizer on $\boldsymbol{A}_{(n)}$, e.g., $r_n(\boldsymbol{A}_{(n)}) = \|\text{vec}(\boldsymbol{A}_{(n)})\|_1$ for promoting sparsity of $\boldsymbol{A}_{(n)}$, and $\mathcal{A}_n$ denotes a constraint set of $\boldsymbol{A}_{(n)}$ (e.g., $\mathcal{A}_n = \mathbb{R}_{+}^{I_n \times F}$). Note that $\boldsymbol{A}_{(n)} \in \mathcal{A}_n$ can also be written as a regularization $r_n\left(\boldsymbol{A}_{(n)}\right)$ if $r_n(\cdot)$ is defined as the indicator function of set $\mathcal{A}_n$. Using the same fiber sampling strategy as in the previous subsection, we update $\boldsymbol{A}_{(n)}$ by

$$\boldsymbol{A}_{(n)}^{(r+1)} \leftarrow \arg\min_{\boldsymbol{A}_{(n)}} \; \big\|\boldsymbol{A}_{(n)} - \big(\boldsymbol{A}_{(n)}^{(r)} - \alpha^{(r)}\boldsymbol{G}_{(n)}^{(r)}\big)\big\|_F^2 + r_n\left(\boldsymbol{A}_{(n)}\right) \tag{11a}$$

$$\boldsymbol{A}_{(n')}^{(r+1)} \leftarrow \boldsymbol{A}_{(n')}^{(r)}, \quad n' \neq n \tag{11b}$$

Algorithm 1: BR-SGD

input : N-way tensor $\underline{\boldsymbol{X}} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$; rank F; sample size $\overline{F}$, initialization $\{\boldsymbol{A}_{(n)}^{(0)}\}$

$r \leftarrow 0$;
repeat
  uniformly sample n from $\{1, \ldots, N\}$, then sample $\mathcal{F}_n$ from $\{1, \ldots, J_n\}$ with $|\mathcal{F}_n| = \overline{F}$;
  form the stochastic gradient $\boldsymbol{G}^{(r)} \leftarrow$ (8);
  update $\boldsymbol{A}_{(n)}^{(r+1)} \leftarrow$ (11a), $\boldsymbol{A}_{(n')}^{(r+1)} \leftarrow \boldsymbol{A}_{(n')}^{(r)}$ for $n' \neq n$;
  $r \leftarrow r + 1$;
until *some stopping criterion is reached*;

output: $\{\boldsymbol{A}_{(n)}^{(r)}\}_{n=1}^{N}$

Problem (11a) is also known as the proximal operator—and many $r_n(\cdot)$'s admit very simple closed-form solutions for their respective proximal operators, e.g., when $r_n(\cdot)$ is the indicator function of the nonnegative orthant and $r_n(\cdot) = \|\cdot\|_1$; see more details in [4, 15]. The complexity of computing the above is similar to that of the plain update in (9), and thus is also computationally efficient. An overview of the proposed algorithm can be found in algorithm 1, which we named as *Block-Randomized SGD* (BR-SGD).

3.3. Convergence Properties

The proposed algorithm admits a low per-iteration complexity and is flexible in handling constraints and regularizations. These are important practical considerations. On the other hand, since we tackle the CPD problem via sample approximations, a natural question is does the algorithm even converge? The answer is affirmative, under some conditions and carefully selected step size $\alpha^{(r)}$.

We first show that the stochastic gradient oracle in (8) is *unbiased*. Let $\mathcal{B}^{(r)}$ be the filtration of random variables $\{\{\boldsymbol{A}_{(n)}^{(r')}\}_{n=1}^{N}\}_{r'=0}^{(r)}$ during the iterations of BR-SGD, for any $n = 1, ..., N$, we have

$$\begin{aligned}
\overline{\boldsymbol{G}}_{(n)}^{(r)} &= \mathbb{E}\big[\boldsymbol{G}_{(n)}^{(r)} | \mathcal{B}^{(r)}\big] = \mathbb{E}\big[\boldsymbol{G}_{(n)}^{(r)} | \{\boldsymbol{A}_{(n)}^{(r)}\}_{n=1}^{N}\big] \\
&= \mathbb{E}_{n'}\left[\frac{1}{\binom{J_{n'}}{\overline{F}}}\big(\boldsymbol{A}_{(n')}^{(r)}\boldsymbol{H}_{(n')}^{\top}\boldsymbol{H}_{(n')} - \boldsymbol{X}_{(n')}^{\top}\boldsymbol{H}_{(n')}\right] \\
&\overset{(a)}{=} \sum_{n'=1}^{N} \frac{\delta(n'-n)}{N\binom{J_{n'}}{\overline{F}}}\big(\boldsymbol{A}_{(n')}^{(r)}\boldsymbol{H}_{(n')}^{\top}\boldsymbol{H}_{(n')} - \boldsymbol{X}_{(n')}^{\top}\boldsymbol{H}_{(n')}\big) \\
&= \frac{1}{N\binom{J_n}{\overline{F}}}\big(\boldsymbol{A}_{(n)}^{(r)}\boldsymbol{H}_{(n)}^{\top}\boldsymbol{H}_{(n)} - \boldsymbol{X}_{(n)}^{\top}\boldsymbol{H}_{(n)}\big)
\end{aligned} \tag{12}$$

where $\delta(\cdot)$ is the Dirac function and the expectation in *(a)* is taken over the possible modes n'. The last equality shows that $\overline{\boldsymbol{G}}_{(n)}^{(r)}$ is a scaled version of the gradient of the objective function of (3) taken w.r.t. $\boldsymbol{A}_{(n)}^{(r)}$. Hence, the block sampling step together with fiber sampling entails us an easy way to estimate the *full gradient w.r.t. all the latent factors* in an unbiased manner. Based on this observation, we first have the following convergence property:

Proposition 1 *Consider the case with $r_n(\cdot) = 0$. Assume that $\boldsymbol{A}_{(n)}^{(r)}$ is bounded for all r, n; in particular, we have $\sigma_{\max}(\boldsymbol{H}_{(n)}^{(r)})^2 \leq L_{(n)}^{(r)} \leq L$ where $\boldsymbol{H}_{(n)}^{(r)} = \odot_{n'=1, n' \neq n}^{N}\boldsymbol{A}_{(n')}^{(r)}$. If we set $\{\alpha^{(r)}\}$ as a diminishing sequence such that $\sum_r \alpha^{(r)} = \infty$ and $\sum_r (\alpha^{(r)})^2 < \infty$, the solution sequence produced by BR-SGD satisfies:*

$$\lim_{r \to \infty} \inf \mathbb{E}\left[\left\|\nabla_{\boldsymbol{A}_{(n)}} f\left(\{\boldsymbol{A}^{(r)}\}\right)\right\|^2\right] \to 0, \; \forall n.$$

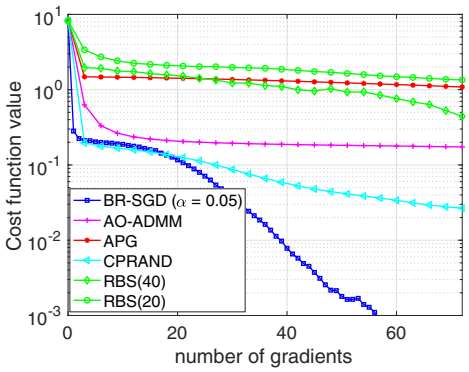

Fig. 2. Cost value against flops needed (measured by number of gradients in (12)) for the algorithms under test; $F = 200$.

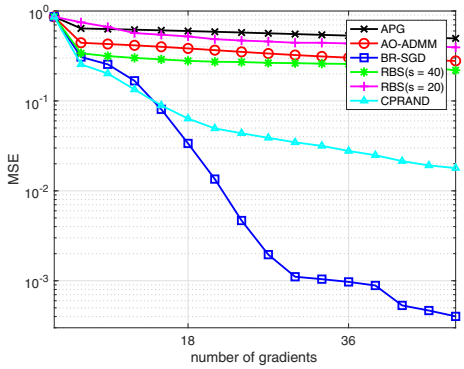

Fig. 3. Average MSE of the latent factors against flops needed for the algorithms under test. $F = 200$;

Table 1. MSE of the latent factors under different ranks after 60 full gradients of w.r.t. $\boldsymbol{A}_{(n)}$; $I_n = 300, N = 3$.

F	200	300	400	500
AO-ADMM	0.2669	0.2753	0.2798	0.2849
APG	0.4629	0.4942	0.5207	0.5433
BR-SGD (Proposed)	$\mathbf{3.9089 \times 10^{-4}}$	**0.0013**	**0.0019**	**0.0075**
RBS(s =40)	0.1388	0.2664	0.2799	0.2841
RBS(s =20)	0.3510	0.3856	0.6814	0.7350

The proof is omitted due to page limitations. The above proposition implies that there exists a subsequence of solution that converges to a stationary point in expectation. We should mention that the SGD/stochastic proximal gradient type update and the block sampling step are essential for establishing convergence—and using the exact solution to (7) as in [12] may not have such convergence properties. Another remark is that the assumption $\sigma_{\max}(\boldsymbol{H}_{(n)}^{(r)})$ being bounded through all the iterations may be hard to check or guarantee in theory, but imposing constraints or regularizations on $\boldsymbol{A}_{(n)}$'s can prevent unboundedness from happening in practice. The case for a variety of $r_n(\cdot) \neq 0$ can be shown in a similar way, with more cumbersome specifications for the optimality conditions, and thus is skipped for saving space.

4. SIMULATION AND CONCLUSION

In this section, we use simulations to showcase the effectiveness of the proposed algorithm. Throughout this section, we use synthetic tensors whose latent factors are drawn from i.i.d. uniform distribution in between 0 and 1. This way, large and *dense* tensors can be created. We test the algorithms on tensors having a size of $300 \times 300 \times 300$, which means that the tensors admit 9×10^6 real-valued elements. Different F's are tested. As baselines, a number of effective CPD algorithms are also tested on the same tensors, namely, the AO-ADMM algorithm [16], the APG algorithm [5], the CPRAND algorithm [12], and the RBS algorithm [10]. Note that BR-SGD, CPRAND and RBS are sampling based stochastic algorithms, while AO-ADMM and APG work with the whole data. For BR-SGD, we use $|\mathcal{F}_n| = 9$ and set the step size to be β/r^α, where r is the number of iterations, $\alpha = 10^{-6}$ and β typically takes a value in between 0.001 and 0.1. For CPRAND, we follow the instruction in the original paper [12] and sample $10F \log_2 F$ fibers for each update. For RBS, we use the implementation in Tensorlab [17]. RBS samples $s \times s \times s$ sub-tensors from the original tensor in each iteration. We set $s = 20$ and 40 respectively, so that the numbers of sampled entries by RBS in the two cases are roughly 3 and 23 times of that sampled by BR-SGD (i.e., 9×300), respectively. All the algorithms above work under nonnegativity constraints (except CPRAND and RBS which cannot handle constraints) .

To measure performance, we employ two metrics. The first one is cost value—since all the algorithms work with the same objective function. The second one is the estimation accuracy of the latent factors, $\boldsymbol{A}_{(n)}$ for $n = 1, \ldots, N$. The accuracy is measured by the mean squared error (MSE) which is as defined in [18, 19]. All the simulations are conducted in Matlab. All the results are averaged from 10 random trials with different tensors.

Fig. 2 shows the cost value against the number of flops used (measured by the number of flops needed to compute a full gradient w.r.t. $\boldsymbol{A}_{(n)}$ in (12)). In this case, we set $F = 200$—which is considered a relatively easy case since $F \leq I_n$. Nevertheless, since the size of the tensor is large, such cases are of interest. One can see that the number of flops that the proposed method needs to decrease the cost value to a satisfactory level is substantially smaller than those needed by the baseline algorithms. The CPRAND algorithm also decreases the cost value faster than other baselines, possibly because it uses a similar sampling strategy as that of BR-SGD.

Fig. 3 shows the corresponding MSE performance in the same simulation. The proposed BR-SGD and CPRAND are still the best and second best-performing algorithms. This echoes our comment that fiber sampling may help improve the estimation accuracy of $\boldsymbol{A}_{(n)}$'s more quickly compared to entry sampling. In particular, BR-SGD outputs an MSE that is at the order of 10^{-3} after using flops equivalent to that of 18 gradients w.r.t. a single $\boldsymbol{A}_{(n)}$, while the best baseline's (i.e., CPRAND) MSE is still above 10^{-2} using the same amount of flops.

Table 1 presents the MSE performance of the algorithms when F changes. All the algorithms are stopped when the number of flops reaches the amount of flops needed for computing 60 gradients w.r.t. a single latent factor $\boldsymbol{A}_{(n)}$. Note that we dropped CPRAND in this simulation since it is too slow when F becomes large. One can see that the proposed method works remarkably well under all the ranks under test, even when F largely exceeds I_n.

To conclude, we proposed a block-randomized stochastic gradient based CPD algorithm for large-scale dense tensors. The algorithm works under a doubly stochastic manner, which randomly samples a mode and then a set of fibers for updating the latent factors. The algorithm has a series of nice features including being able to quickly improve estimation accuracy of the latent factors and having rigorous convergence guarantees. Simulation results show that the algorithm outperforms a number of state-of-art CPD algorithms when dealing with large dense tensors.

5. REFERENCES

[1] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3551–3582.

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

[3] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos, "Tensors for data mining and data fusion: Models, applications, and scalable algorithms," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, no. 2, p. 16, 2017.

[4] K. Huang, N. D. Sidiropoulos, and A. P. Liavas, "A flexible and efficient algorithmic framework for constrained matrix and tensor factorization," *IEEE Trans. Signal Process.*, vol. 64, no. 19, pp. 5052–5065, 2016.

[5] Y. Xu and W. Yin, "A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion," *SIAM Journal on imaging sciences*, vol. 6, no. 3, pp. 1758–1789, 2013.

[6] A. P. Liavas and N. D. Sidiropoulos, "Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers," *IEEE Trans. Signal Process.*, vol. 63, no. 20, pp. 5450–5463, 2015.

[7] U. Kang, E. Papalexakis, A. Harpale, and C. Faloutsos, "Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries," in *Proc. ACM SIGKDD 2012*, 2012, pp. 316–324.

[8] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos, "Parcube: Sparse parallelizable tensor decompositions," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2012, pp. 521–536.

[9] C. I. Kanatsoulis, X. Fu, N. D. Sidiropoulos, and W.-K. Ma, "Hyperspectral super-resolution: A coupled tensor factorization approach," *IEEE Trans. Signal Process.* to appear, 2018.

[10] N. Vervliet and L. De Lathauwer, "A randomized block sampling approach to canonical polyadic decomposition of large-scale tensors," *IEEE J. Sel. Topics Signal Process.*, vol. 10, no. 2, pp. 284–295, 2016.

[11] A. Beutel, P. P. Talukdar, A. Kumar, C. Faloutsos, E. E. Papalexakis, and E. P. Xing, "Flexifact: Scalable flexible factorization of coupled tensors on hadoop," in *Proc. SIAM SDM 2014*. SIAM, 2014, pp. 109–117.

[12] C. Battaglino, G. Ballard, and T. G. Kolda, "A practical randomized cp tensor decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 39, no. 2, pp. 876–901, 2018.

[13] J. D. Carroll and J.-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.

[14] E. E. Papalexakis, U. Kang, C. Faloutsos, N. D. Sidiropoulos, and A. Harpale, "Large Scale Tensor Decompositions: Algorithmic Developments and Applications," *IEEE Data Engineering Bulletin, Special Issue on Social Media and Data Analysis*, vol. 36, no. 3, pp. 59–66, Sep. 2013.

[15] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in optimization*, vol. 1, no. 3, pp. 123–231, 2013.

[16] K. Huang, N. Sidiropoulos, E. Papalexakis, C. Faloutsos, P. Talukdar, and T. Mitchell, "Principled neuro-functional connectivity discovery," in *Proc. SIAM SDM 2015*, 2015.

[17] N. Vervliet, O. Debals, and L. De Lathauwer, "Tensorlab 3.0–numerical optimization strategies for large-scale constrained and coupled matrix/tensor factorization," in *Proc. Asilomar 2016*, 2016, pp. 1733–1738.

[18] X. Fu, W.-K. Ma, K. Huang, and N. D. Sidiropoulos, "Blind separation of quasi-stationary sources: Exploiting convex geometry in covariance domain," *IEEE Trans. Signal Process.*, vol. 63, no. 9, pp. 2306–2320, May 2015.

[19] L. D. Lathauwer and J. Castaing, "Blind identification of underdetermined mixtures by simultaneous matrix diagonalization," *IEEE Trans. Signal Process.*, vol. 56, no. 3, pp. 1096 –1105, Mar. 2008.