Block-Randomized Stochastic Proximal Gradient for Low-Rank Tensor Factorization

Xiao Fu*, Cheng Cao*, Hoi-To Wai†, and Kejun Huang⋆

*School of Electrical Engineering and Computer Science
Oregon State University
†Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong
⋆Department of Computer and Information Science and Engineering
University of Florida

January 16, 2019

Abstract

This work considers the problem of computing the *canonical polyadic decomposition* (CPD) of large tensors. Prior works mostly leverage data sparsity to handle this problem, which are not suitable for handling dense tensors that often arise in applications such as medical imaging, computer vision, and remote sensing. Stochastic optimization is known for its low memory cost and per-iteration complexity when handling dense data. However, existing stochastic CPD algorithms are hard to incorporate a variety of constraints and regularizations that are of interest in signal and data analytics. Convergence properties of many such algorithms are also unclear. In this work, we propose a stochastic optimization framework for large-scale CPD with constraints/regularizations. The framework works under a doubly randomized fashion, and can be regarded as a judicious combination of *randomized block coordinate descent* (BCD) and *stochastic proximal gradient* (SPG). The algorithm enjoys lightweight updates and small memory footprint, and thus scales well. In addition, this framework entails considerable flexibility—many frequently used regularizers and constraints can be readily handled under the proposed scheme. The approach is also supported by convergence analysis. Numerical results on large-scale dense tensors are employed to showcase the effectiveness of the proposed approach.

1 Introduction

Canonical polyadic decomposition (CPD) [previously known as parallel factor analysis (PARAFAC)] [1–3] is arguably the most popular low-rank tensor decomposition model. CPD has successfully found many applications in various fields, such as analytical chemistry [4], social network mining [5], hyperspectral imaging [6], topic modeling [7], and time series analysis [8]; also see [9–11] for more classic applications in communications.

Computing the CPD of a tensor, however, is a quite challenging optimization problem [12]. Many algorithms have been proposed through the years [3, 13–15]. To keep pace with the ever

growing volume of available data, one pressing challenge is to compute CPD at scale. The classic alternating least squares (ALS) algorithm [3] has an elegant algorithmic structure, but also suffers from a number of numerical issues [16, 17] and is hardly scalable. In recent years, many new CPD algorithms have appeared, triggered by the advances in big data analytics and first-order optimization [13, 14, 18, 19]. Many of these algorithms leverage data sparsity to scale up CPD—by cleverly using the zero elements in huge tensors, computationally costly key operations in ALS (e.g., the *matricized tensor times Khatri-Rao product* (MTTKRP) operation) can be significantly simplified. Consequently, the classic ALS algorithm can be modified to handle CPD of huge and sparse tensors.

However, when the tensor to be factored is *dense*—i.e., when most entries of the tensor are nonzero—the sparsity-enabled efficient algorithms [6, 13, 14, 18, 19] are no longer applicable. Note that large and dense tensors arise in many timely and important applications such as medical imaging [20], hyperspectral imaging [6], and computer vision [21]. 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 into the RAM of laptops, desktops, or servers. This also raises serious challenges in the era of Internet of Things (IoT)—where edge computing on small devices is usually preferable.

Stochastic approximation is a powerful tool for handling optimization problems involving dense data, which is known for its low per-iteration memory and computational complexities [22]. A number of stochastic optimization based CPD algorithms have been proposed in the literature [23–25]. Specifically, The works in [23, 24] work in an iterative manner. In each iteration, the algorithm samples a random subset of the tensor entries and update the corresponding parts of the latent factors using the sampled data. The algorithms have proven quite effective in practice, and features distributed implementation [24]. 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. More importantly, this update strategy loses the opportunity to incorporate constraints/regularizations on the whole latent factors, since the sampled entries only contain partial information of them. This is undesired in practice, since prior information on the latent factors are critical for enhancing performance, especially in noisy cases.

Recently, a stochastic algorithm that ensures updating one entire latent factor in every iteration was proposed in [25]. Instead of sampling tensor entries, the algorithm works via sampling *tensor fibers* that contain information of the whole latent factors. However, this algorithm 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 [23, 24]. In addition, like those in [23, 24], the algorithm in [25] cannot handle constraints or regularizations on the latent factors, either. In addition, although empirically working well, convergence properties of many stochastic CPD algorithms such as those in [23, 25] are unclear.

Contributions In this work, we propose a new stochastic algorithmic framework for computing the CPD of large-scale dense tensors. Specifically, our contributions include:

• **A Doubly Randomized Computational Framework for Large-Scale CPD**. Our first contribution lies in proposing an efficient and flexible computational framework for CPD of large dense tensors. Our method is a judicious combination of randomized block coordinate descent (BCD) [26, 27] and stochastic proximal gradient (SPG) [28, 29]. Specifically, in each iteration, our method first samples a mode from all modes of the tensor. Then, the algorithm samples some fibers

of this mode and updates the corresponding latent factor via stochastic proximal operations. Such a combination exhibits an array of attractive features: It admits much smaller per-iteration memory and computational complexities relative to the existing fiber sampling based method in [25]. More importantly, it is very flexible in terms of incorporating regualrizations and constraints on the latent factors.

• **Rigorous Convergence Analysis**. Both BCD and SPG are well studied topics in the optimization literature [26, 27, 30]. However, convergence properties of the proposed framework is not immediately clear, due to the nonconvex nature of CPD. The existing block-randomized SGD (BR-SGD) framework [31] only considers convex optimization. A related work in [32] deals with nonconvex problems via block stochastic gradient, but their *Gauss-Seidel* type BCD strategy (without block randomization) makes the convergence analysis inapplicable to our case. Hence, we offer tailored convergence analyses for our proposed CPD algorithms.

• **Implementation-friendly Adaptive Stepsize Scheduling**. In practice, one of the most challenging aspects in stochastic optimization is selecting a proper stepsize schedule. To make the proposed algorithms friendly to use by practitioners, we propose a practical and adaptive stepsize schedule that is based on the celebrated `Adagrad` algorithm [33]. `Adagrad` is an adaptive stepsize selection method that was devised for single-block gradient descent. Nonetheless, we find through extensive simulations that it largely helps reduce the agonizing pain of tuning stepsize when implementing our multi-block algorithm for CPD. In addition, we also show that the adaptive stepsize-based algorithm converges to a stationary problem almost surely under some conditions.

A quick demonstration of the effectiveness of the proposed algorithms is shown in Fig. 1, where the average mean squared error (MSE) of the estimated latent factors [cf. Eq. (21)] against the number of MTTKRP computed (which dominates the complexity) is plotted. One can see that the proposed algorithm largely outperforms a couple of state-of–the-art algorithms for constrained CPD. More thorough numerical results can be seen in Sec. 6.

Part of the work was submitted to ICASSP 2019 [34]. In this new version, we have additionally included detailed convergence proofs and the new adaptive stepsize based algorithm. More extensive simulations and real-data experiments are also included.

Notation. We follow the established conventions in signal processing. x, $\boldsymbol{x}$, $\boldsymbol{X}$, and $\underline{\boldsymbol{X}}$ denote scaler, vector, matrix, and tensor, respectively; $\|\cdot\|$ denotes the Euclidean norm, i.e., $\|\boldsymbol{x}\|_2$ and $\|\boldsymbol{X}\|_F$, respectively; $\circ$, $\odot$, and $\circledast$ denote outer product, Khatri-Rao product, and Hadamard product, respectively; $\text{vec}(\boldsymbol{X})$ denotes the vectorization operator that concatenates the columns of $\boldsymbol{X}$; $\boldsymbol{X} \geq \boldsymbol{0}$ means that all the entries of $\boldsymbol{X}$ are nonnegative; $^\top$ and † denote transpose and pseudo-inverse, respectively; $|\mathcal{C}|$ denotes the cardinality of of set $\mathcal{C}$; $\lambda_{\max}(\cdot)$ denotes the largest eigenvalue of a matrix.

2 Background

We first introduce some notions that are heavily used in tensor algebra.

2.1 Tensors and CPD

An Nth order tensor is an array whose entries are indexed by N coordinates; i.e., $\underline{\boldsymbol{X}}(i_1, \ldots, i_N)$ denotes an element of the tensor $\underline{\boldsymbol{X}}$ with a size of $I_1 \times I_2 \times \ldots \times I_N$. Like matrices, tensors can be

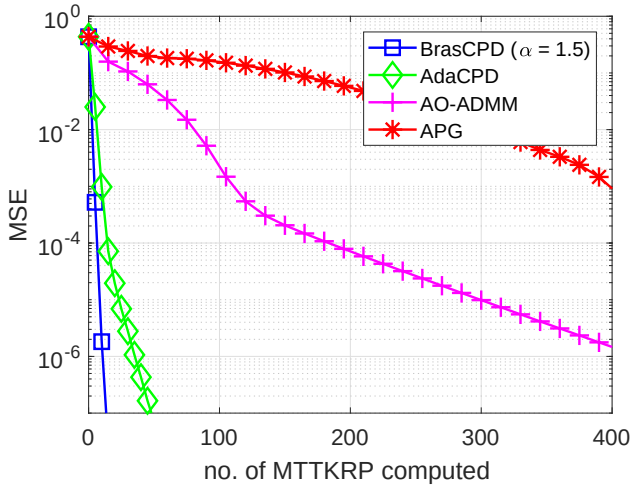

Figure 1: The proposed algorithms (`AdaCPD` and `BrasCPD`) exhibit low complexity for achieving high accuracy of the estimated latent factors. The tensor under test has a size of $100 \times 100 \times 100$ and the rank is 10. The latent factors are constrained to be nonnegative. The baselines are two state-of-art constrained CPD algorithms `AO-ADMM` [13] and `APG` [14].

represented as sum of rank-one components:

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

where "$\circ$" denotes the outer product of vectors, and $\boldsymbol{A}_{(n)}$ an $I_n \times F$ matrix that is often referred to as the *mode-n latent factor*. 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 $\underline{\boldsymbol{X}}$. At the entry level, the CPD can be expressed as

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

for $i_n \in \{1, \ldots, I_n\}$. The CPD of a tensor is essentially unique under mild conditions (meaning that the latent factors $\boldsymbol{A}_{(n)}$'s that constitute the data $\underline{\boldsymbol{X}}$ are unique up to some trivial ambiguities like column permutations and scalings [2]). In practice, the CPD of a tensor is often obtained via solving a certain optimization criterion

$$\underset{\{\boldsymbol{A}_{(n)}\}_{n=1}^{N}}{\text{minimize}} \;\; f(\boldsymbol{A}_{(1)}, \ldots, \boldsymbol{A}_{(N)}). \tag{3}$$

One of most commonly seen optimization surrogate for CPD in the literature is the *least squares* (LS) fitting criterion [3, 13, 14]:

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

In the sequel, we will often use the shorthand notation $f(\boldsymbol{\theta})$ to denote $f(\boldsymbol{A}_{(1)}, \ldots, \boldsymbol{A}_{(N)})$, where

$$\boldsymbol{\theta} = [\mathrm{vec}(\boldsymbol{A}_{(1)})^\top, \ldots, \mathrm{vec}(\boldsymbol{A}_{(N)}^\top)]^\top.$$

Note that many other criteria have also been considered, e.g., the Kullback-Leibler (KL) divergence [35] and robust fitting [36, 37] criteria—which serve for different purposes. Nevertheless, the LS fitting criterion is arguably the most popular one.

2.2 Unfolding, ALS and MTTKRP

The matricization operation, or *matrix unfolding* of a tensor, has proven very useful in designing tensor factorization algorithms. The mode-n unfolding of a tensor is a $J_n \times I_n$ matrix where

$$\underline{\boldsymbol{X}}(i_1, \ldots, i_N) = \boldsymbol{X}_{(n)}(j, i_n),$$

and we have $j = 1 + \sum_{k=1, k\neq n}^{N} (i_k - 1) J_k$ and $J_k = \prod_{m=1, m\neq n}^{k-1} I_m$ [1]. The CPD representation in Eq. (1) can be expressed as

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

where the $J_n \times F$ matrix $\boldsymbol{H}_{(n)}$ is defined as

$$\boldsymbol{H}_{(n)} = \boldsymbol{A}_{(1)} \odot \boldsymbol{A}_{(n-1)} \odot \boldsymbol{A}_{(n+1)} \odot \ldots \odot \boldsymbol{A}_{(N)} = \odot_{i=1, i\neq n}^{N} \boldsymbol{A}_{(i)}.$$

The elegant form of the unfoldings has enabled the famous *alternating least squares* (ALS) algorithm [3] for handling Problem (3) with the LS objective. Specifically, ALS solves the following cyclically for $n = 1, \ldots, N$:

$$\boldsymbol{A}_{(n)} \leftarrow \arg\min_{\boldsymbol{A}} \ \left\| \boldsymbol{X}_{(n)} - \boldsymbol{H}_{(n)} \boldsymbol{A}^{\top} \right\|_F^2. \tag{5}$$

Problem (5) is nothing but a least squares problem that admits the following closed-form solution:

$$\boldsymbol{A}_{(n)} \leftarrow \left((\boldsymbol{H}_{(n)}^{\top} \boldsymbol{H}_{(n)})^{-1} \boldsymbol{H}_{(n)}^{\top} \boldsymbol{X}_{(n)} \right)^{\top},$$

if $\mathrm{rank}(\boldsymbol{H}_{(n)}) = F$. Note that $(\boldsymbol{H}_{(n)}^{\top} \boldsymbol{H}_{(n)})^{-1}$ is not difficult to compute by exploiting the Khatri-Rao structure of $\boldsymbol{H}_{(n)}$ [1, 2, 13]. However, when the problem dimension is large (which often happens in applications such as medical imaging, remote sensing, and computer vision), solving the seemingly simple problem in (5) can be computationally prohibitive. The reason is that both $\boldsymbol{X}_{(n)} \in \mathbb{R}^{(\prod_{j=1, j\neq n}^{N} I_j) \times I_n}$ and $\boldsymbol{H}_{(n)} \in \mathbb{R}^{(\prod_{j=1, j\neq n}^{N} I_j) \times F}$ can be very large matrices. In particular, the so-called *matricized tensor times Khatri-Rao product* (MTTKRP) operation, i.e.,

$$\mathsf{MTTKRP}: \qquad \boldsymbol{H}_{(n)}^{\top} \boldsymbol{X}_{(n)}$$

that happens in every iteration of ALS costs $\mathcal{O}(\prod_{n=1}^{N} I_n F)$ flops (or, $\mathcal{O}(I^N F)$ if $I_n = I$). This is quite costly even if I_n is moderately large. Many works have considered fast algorithms for computing MTTKRP, but these methods are mainly enabled by judiciously exploiting sparsity of the tensor data [18, 38].

In a lot of applications, some prior knowledge on the latent factors is known—e.g., in image processing, $\boldsymbol{A}_{(n)}$'s are normally assumed to be nonnegative [6]; in statistical machine learning, sometimes the columns of $\boldsymbol{A}_{(n)}$ are assumed to be constrained within the probability simplex [35,39]; i.e.,

$$\mathbf{1}^{\top} \boldsymbol{A}_{(n)} = \mathbf{1}^{\top}, \ \boldsymbol{A}_{(n)} \geq \mathbf{0}. \tag{6}$$

In those cases, the following criterion is often of interest:

$$\begin{aligned} &\underset{\{\boldsymbol{A}_{(n)}\}_{n=1}^{N}}{\text{minimize}}\ f(\boldsymbol{A}_{(1)},\ldots,\boldsymbol{A}_{(N)}) + \sum_{n=1}^{N} h_n(\boldsymbol{A}_{(n)}) \\ &\text{subject to } \boldsymbol{A}_{(n)} \in \mathcal{A}_n \end{aligned} \tag{7}$$

Compared to the unconstrained version, Problem (7) is even harder to handle. Some recent methods combine first-order constrained optimization and ALS [13, 14] to make the tensor factorization algorithms more flexible in handling constraints and regularizations—but the complexity orders of those algorithms often scale similarly as that of ALS, since these algorithms cannnot avoid computing $\boldsymbol{H}_{(n)}^{\top}\boldsymbol{X}_{(n)}$ that is the bottleneck for computing CPD.

2.3 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. Using Eq. (2), Problem (3) with the LS objective is equivalent to the following:

$$\underset{\{\boldsymbol{A}_{(n)}\}}{\text{minimize}}\ (1/T)\ \textstyle\sum_{i_1,\ldots,i_N}\ f_{i_1,\ldots,i_N}(\boldsymbol{\theta})\,, \tag{8}$$

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

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

The objective function in (8) can be understood as the empirical risk of SA [22]. Using this observation, the algorithms in [23, 24] randomly sample a subset set of entries indexed by $\{(i_1,\ldots,i_N)\}$ 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, [24] uses a stochastic gradient (SG) based approach and update the $\boldsymbol{A}_{(n)}(i_n,:)$'s that are associated with the sampled entries. The sampling method in [23] is similar, while the update is not gradient-based but Gauss-Newton or ALS applied to the sampled set of entries (or, sub-tensors, to be precise). The upshot of this line of work is that the per-iteration complexity can be quite low.

Despite of such favorable complexity savings, the approaches in [23, 24] have a couple of limitations. First, 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. Second—which is perhaps more critical—many useful prior information cannot be incorporated in the algorithm. The reason is that these algorithms update some of the *rows* of $\boldsymbol{A}_{(n)}$'s, while many useful priors are defined w.r.t. the *columns* of the latent factors, e.g., the probability simplex constraint in (6) and the *total variation constraint* that is heavily used in image processing. Third, convergence properties of these methods are often unclear.

An alternative [25] 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. 2) is a row

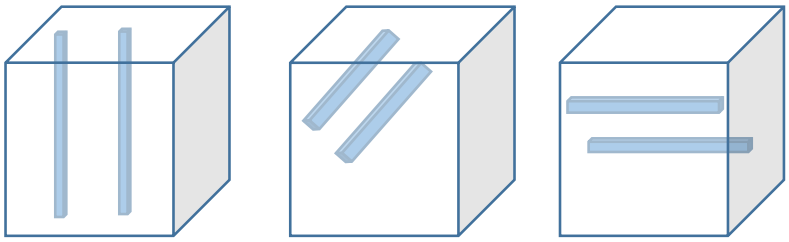

Figure 2: From left to right: mode-$1, 2, 3$ tensor fibers of a third-order tensor, respectively.

of the mode-n unfolding $\boldsymbol{X}_{(n)}$ [1]. 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):

$$\boldsymbol{A}_{(n)} \leftarrow \arg\min_{\boldsymbol{A}} \ \left\| \boldsymbol{X}_{(n)}(\mathcal{F}_n, :) - \boldsymbol{H}_{(n)}(\mathcal{F}_n, :)\boldsymbol{A}^\top \right\|_F^2, \tag{9}$$

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 [25] 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 larger than I_n in tensor decomposition. 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 this method is also unclear.

3 Proposed Algorithm

In this work, we propose a new stochastic optimization strategy for CPD. Our method combines the insights from ALS and fiber sampling, but allows $|\mathcal{F}_n| \ll F$. This is instrumental in practice, since it is the key for achieving low per-iteration complexity. The proposed algorithm can easily handle a variety of constraints and regularizations that are commonly used in signal processing and data analytics—which is reminiscent of *stochastic proximal gradient* (SPG) [29, 40]. In addition, we provide convergence analyses to back up the proposed approach.

3.1 Basic Idea: Unconstrained Case

We first consider Problem (3). Our idea is combining SA and exploiting the tensor fiber structure. Specifically, at each iteration, we sample a set of mode-n fibers for a certain n as the method in [25] does. However, instead of exactly solving the least squares subproblems (5) for all the modes following a Gauss-Seidel manner in each iteration, we update $\boldsymbol{A}_{(n)}$ using a doubly stochastic procedure. To be more precise, at iteration r, we first randomly sample a mode index $n \in \{1, ..., N\}$. Then, we randomly sample a set of mode-n fibers that is indexed by $\mathcal{F}_n \subset \{1, ..., J_n\}$. 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:

$$\begin{aligned} \boldsymbol{G}_{(n)}^{(r)} &= \frac{1}{|\mathcal{F}_n|}\Big(\boldsymbol{A}_{(n)}^{(r)}\boldsymbol{H}_{(n)}^{\top}(\mathcal{F}_n)\boldsymbol{H}_{(n)}(\mathcal{F}_n) - \boldsymbol{X}_{(n)}^{\top}(\mathcal{F}_n)\boldsymbol{H}_{(n)}(\mathcal{F}_n)\Big) \\ \boldsymbol{G}_{(n')}^{(r)} &= \boldsymbol{0}, \quad n' \neq n, \end{aligned} \tag{10}$$

where $\boldsymbol{G}_{(n)}^{(r)}$ denotes the nth block of $\boldsymbol{G}^{(r)}$, and we used the shorthand notations

$$\boldsymbol{X}_{(n)}(\mathcal{F}_n) = \boldsymbol{X}_{(n)}(\mathcal{F}_n, :), \quad \boldsymbol{H}_{(n)}(\mathcal{F}_n) = \boldsymbol{H}_{(n)}(\mathcal{F}_n, :).$$

The latent variables are updated by

$$\boldsymbol{A}_{(n)}^{(r+1)} \leftarrow \boldsymbol{A}_{(n)}^{(r)} - \alpha^{(r)}\boldsymbol{G}_{(n)}^{(r)}, \quad n = 1, ..., N. \tag{11}$$

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 [13,14] 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 $|\mathcal{F}_n|$ 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 will show, this modification greatly simplifies our convergence analysis.

3.2 Constrained and Regularized Case

As mentioned, there are many cases in practice where considering regularizations or constraints on $\boldsymbol{A}_{(n)}$'s can benefit the associated tasks. Since our framework updates an entire $\boldsymbol{A}_{(n)}$ in each iteration, it is friendly for incorporating a large variety of commonly used constraints/regularizations—which is more flexible relative to the entry sampling based approaches in [23, 24]. Specifically, the algorithm can be easily extended to handle the constrained/regularized case:

$$\begin{aligned} &\underset{\{\boldsymbol{A}_{(n)}\}_{n=1}^{N}}{\text{minimize}} \; f(\boldsymbol{\theta}) + \sum_{n=1}^{N} h_n(\boldsymbol{A}_{(n)}) \\ &\text{subject to} \;\; \boldsymbol{A}_{(n)} \in \mathcal{A}_n, \end{aligned} \tag{12}$$

where $f(\boldsymbol{\theta})$ is the objective function of (3), $h_n(\boldsymbol{A}_{(n)})$ denotes a structure-promoting regularizer on $\boldsymbol{A}_{(n)}$. Note that $\boldsymbol{A}_{(n)} \in \mathcal{A}_n$ can also be written as a regularization $h_n\left(\boldsymbol{A}_{(n)}\right)$ if $h_n(\cdot)$ is defined as the indicator function of set $\mathcal{A}_n$, i.e.,

$$h_n(\boldsymbol{A}) = \mathcal{I}(\mathcal{A}_n) = \begin{cases} 0, & \boldsymbol{A} \in \mathcal{A}_n \\ \infty, & \text{o.w.} \end{cases}$$

Algorithm 1: `BrasCPD`

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

$r \leftarrow 0$;
repeat
  uniformly sample n from $\{1, \ldots, N\}$, then sample $\mathcal{F}_n$ from $\{1, \ldots, J_n\}$ with $|\mathcal{F}_n| = B$;
  form the stochastic gradient $\boldsymbol{G}^{(r)} \leftarrow (10)$;
  update $\boldsymbol{A}_{(n)}^{(r+1)} \leftarrow (13a)$, $\boldsymbol{A}_{(n')}^{(r+1)} \leftarrow \boldsymbol{A}_{(n')}^{(r)}$ for $n' \neq n$;
  $r \leftarrow r + 1$;
until *some stopping criterion is reached*;

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

where $\mathcal{I}(\mathcal{X})$ denotes the indicator function of the set $\mathcal{X}$. Using the same fiber sampling strategy as in the previous subsection, we update $\boldsymbol{A}_{(n)}$ by

$$\boldsymbol{A}_{(n)}^{(r+1)} \leftarrow \arg\min_{\boldsymbol{A}_{(n)}} \left\| \boldsymbol{A}_{(n)} - \left(\boldsymbol{A}_{(n)}^{(r)} - \alpha^{(r)} \boldsymbol{G}_{(n)}^{(r)}\right) \right\|_F^2 + h_n\left(\boldsymbol{A}_{(n)}\right) \tag{13a}$$

$$\boldsymbol{A}_{(n')}^{(r+1)} \leftarrow \boldsymbol{A}_{(n')}^{(r)}, \quad n' \neq n \tag{13b}$$

Problem (13a) is also known as the proximal operator of $h_n(\cdot)$, which is often denoted as

$$\boldsymbol{A}_{(n)}^{(r+1)} \rightarrow \mathsf{Prox}_{h_n}\left(\boldsymbol{A}_{(n)}^{(r)} - \alpha^{(r)} \boldsymbol{G}_{(n)}^{(r)}\right). \tag{14}$$

Many $h_n(\cdot)$'s admit simple closed-form solutions for their respective proximal operators, e.g., when $h_n(\cdot)$ is the indicator function of the nonnegative orthant and $h_n(\cdot) = \|\cdot\|_1$; see Table 1 and more details in [13, 41]. The complexity of computing (14) is often similar to that of the plain update in (11), and thus is also computationally efficient. An overview of the proposed algorithm can be found in algorithm 1, which we name ***B**lock-**Ra**ndomized **S**GD for **CPD*** (`BrasCPD`).

Table 1: Proximal/projection operator of some frequently used regularizations and constraints.

$h(\cdot)$	prox./proj. solution	complexity
$\|\cdot\|_1$	soft-thresholding	$\mathcal{O}(d)$
$\|\cdot\|_2$	re-scale	$\mathcal{O}(d)$
$\|\cdot\|_{2,1}$	block soft-thresholding	$\mathcal{O}(d)$
$\|\cdot\|_0$	hard-thresholding	$\mathcal{O}(d)$
$\mathcal{I}(\Delta)$	randomized pivot search [42]	$\mathcal{O}(d)$ in expectation
$\mathcal{I}(\mathbb{R}_+)$	max	$\mathcal{O}(d)$
monotonic	monotone regression [43]	$\mathcal{O}(d)$
unimodal	unimodal regression [44]	$\mathcal{O}(d^2)$

†In the table, d is the number of optimization variables.

4 Convergence Properties

Note that `BrasCPD` does not fall into any known framework of block stochastic gradient optimization, and thus its convergence properties are not immediately clear. Two most relevant works from the optimization literature are [14] and [31]. However, the work in [14] considers Gauss-Seidel type block SGD (i.e., cyclically updating the blocks), instead of the block-randomized version as `BrasCPD` uses. The work in [31] considers block-randomized SGD, but only for the convex case. In addition, many assumptions made in [14,31] for their respective generic optimization problems are not easily satisfied by our CPD problem. In this section, we offer tailored convergence analyses for `BrasCPD`.

4.1 Unconstrained Case

To proceed, we will use the following assumptions:

Assumption 1 *The stepsize schedule follows the Robbins-Monro rule [45]:*

$$\sum_{r=0}^{\infty} \alpha^{(r)} = \infty, \quad \sum_{r=0}^{\infty} (\alpha^{(r)})^2 < \infty.$$

Assumption 2 *The updates $\boldsymbol{A}_{(n)}^{(r)}$ are bounded for all n, r.*

Assumption 1 is a principle for stepsize scheduling, which is commonly used in stochastic approximation. Assumption 2 is a working assumption that we make to simplify the analysis. It is considered a relatively strong assumption, since it is hard to check or guarantee. Nevertheless, unbounded iterates rarely happen in practice, if the stepsize is well controlled.

There are also an array of problem structures that are useful for studying convergence of the algorithm.

Fact 1 *The LS fitting part in the objective function* (12) *(i.e.$f(\boldsymbol{\theta})$) satisfies*

$$\begin{aligned} f(\boldsymbol{\theta}) \leq & f(\bar{\boldsymbol{\theta}}) + \langle \nabla_{\boldsymbol{A}_{(n)}} f(\bar{\boldsymbol{\theta}}), \boldsymbol{A} - \bar{\boldsymbol{A}}_{(n)} \rangle \\ & + \frac{\bar{L}_{(n)}}{2} \|\boldsymbol{A} - \bar{\boldsymbol{A}}_{(n)}\|_F^2, \end{aligned} \tag{15}$$

where $\bar{L}_{(n)} \geq \lambda_{\max}(\bar{\boldsymbol{H}}_{(n)}^\top \bar{\boldsymbol{H}}_{(n)})$, $\bar{\boldsymbol{\theta}}$ is a feasible point, and $\bar{\boldsymbol{A}}_{(n)}$ and $\bar{\boldsymbol{H}}_{(n)}$ are extracted/constructed from $\bar{\boldsymbol{\theta}}$ following the respective definitions.

Eq. (15) holds because the objective function $f(\boldsymbol{\theta})$ w.r.t. $\boldsymbol{A}_{(n)}$ is a plain least squares fitting criterion, which is known to have a Lipschitz continuous gradient—and the smallest Lipschitz constant is $\lambda_{\max}(\bar{\boldsymbol{H}}_{(n)}^\top \bar{\boldsymbol{H}}_{(n)})$.

The second fact is instrumental in proving convergence of the algorithm:

Fact 2 *Denote $\xi_{(r)}$ and $\zeta^{(r)}$ as the random variables that are responsible for selecting the mode and fibers in iteration r, respectively. Also denote $\mathcal{B}^{(r)} = \{\xi^{(1)}, \zeta^{(1)}, \ldots, \xi^{(r-1)}, \zeta^{(r-1)}\}$, i.e., the filtration up to r. The block-wise stochastic gradient constructed in* (10) *is an unbiased estimation for the full gradient w.r.t. $\boldsymbol{A}_{(\xi^{(r)})}$, i.e.,*

$$\mathbb{E}_{\zeta^{(r)}} \left[\boldsymbol{G}_{(\xi^{(r)})}^{(r)} \mid \mathcal{B}^{(r)}, \xi^{(r)} \right] = \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}), \tag{16}$$

if $\zeta^{(r)}$ admits the following probability mass function (PMF):

$$\mathsf{Pr}(\zeta^{(r)} = i) = \frac{1}{J_n},\ \forall i \in \{1, \ldots, J_n\}. \tag{17}$$

The proof of the above is straightforward and thus skipped. The Fact says that even if our block stochastic gradient is not exactly an unbiased estimation for $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})$, it is an unbiased estimation for the "block gradient" $\nabla_{\boldsymbol{A}_{(n)}} f(\boldsymbol{\theta}^{(r)})$. This fact will prove quite handy in establishing convergence. In fact, the two-level sampling strategy (i.e., block sampling and fiber sampling, respectively), makes the gradient estimation w.r.t. $\boldsymbol{\theta}$ unbiased up to a scaling factor (see Appendix A). This connection intuitively suggests that the proposed algorithm should behave similarly as an ordinary single-block stochastic gradient descent algorithm.

We first have the following convergence property:

Proposition 1 *Consider the case where $h_n(\cdot) = 0$ and Assumptions 1-2 hold. Then, the solution sequence produced by `BrasCPD` satisfies:*

$$\lim_{r\to\infty} \inf \mathbb{E}\left[\left\|\nabla_{\boldsymbol{A}_{(n)}} f\left(\boldsymbol{A}_{(1)}^{(r)}, \ldots, \boldsymbol{A}_{(N)}^{(r)}\right)\right\|^2\right] = 0,\ \forall n.$$

The proof is relegated to Appendix B. The above proposition implies that there exists a subsequence of the solution sequence 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 (9) as in [25] may not have such convergence properties.

4.2 Constrained/Regularized Case

To understand convergence of the proximal gradient version with $h_n(\cdot) \neq 0$, denote $\Phi(\boldsymbol{\theta}) = f(\boldsymbol{\theta}) + \sum_{n=1}^{N} h_n(\boldsymbol{\theta})$ as the objective function. Our optimality condition amounts to $\boldsymbol{P}_{(n)}^{(r)} = \boldsymbol{0}$, $\forall\ n$, where

$$\boldsymbol{P}_{(n)}^{(r)} = \frac{1}{\alpha^{(r)}}\left(\boldsymbol{A}_{(n)}^{(r+1)} - \mathsf{Prox}_{h_n}\left(\boldsymbol{A}_{(n)}^{(r)} - \alpha^{(r)} \nabla_{\boldsymbol{A}_{(n)}} f(\boldsymbol{\theta}^{(r)})\right)\right);$$

i.e., the optimality condition is satisfied in a blockwise fashion [30, 32]. Hence, our goal of this section is to show that $\mathbb{E}[\|\boldsymbol{P}_{(n)}^{(r)}\|^2]$ for all n vanishes when r grows. We will use the following assumption:

Assumption 3 *There exists a sequence $\sigma^{(r)}$ for $r = 0, 2, \ldots$, such that*

$$\mathbb{E}_{\zeta^{(r)}}\left[\left\|\boldsymbol{G}_{(\xi^{(r)})}^{(r)} - \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)})\right\|^2 \ \middle|\ \mathcal{B}^{(r)}, \xi^{(r)}\right] \leq (\sigma^{(r)})^2,$$

and

$$\sum_{r=0}^{\infty} (\sigma^{(r)})^2 < \infty. \tag{18}$$

We show that `BrasCPD` produces a convergent solution sequence in the following proposition:

Proposition 2 *Assume that Assumptions 1-3 hold. Also assume that $h_n(\cdot)$ is a convex function. Then, the solution sequence produced by* `BrasCPD` *satisfies*

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

Remark 1 *Note that the convergence result in Proposition 2 inherits one possible drawback from single-block stochastic proximal gradient algorithms for nonsmooth nonconvex optimization. To be specific, the relatively strong assumption in* (18) *needs to be assumed for ensuring convergence. Assumption 3 essentially means that the variance of the gradient estimation error $\boldsymbol{\delta}_{(\xi^{(r)})}^{(r)} = \boldsymbol{G}_{(\xi^{(r)})}^{(r)} - \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)})$ decreases and converges to zero. This is not entirely trivial. One way to fulfill this assumption is to increase the minibatch size along the iterations, e.g., by setting [29, 32]:*

$$|\mathcal{F}_n^{(r)}| = \mathcal{O}(\lceil r^{1+\epsilon}\rceil), \quad \forall \epsilon > 0.$$

Then, one can see that $(\sigma^{(r)})^2 = \mathcal{O}(\frac{1}{\lceil r^{1+\epsilon}\rceil})$, so that $\sum_{r=0}^{\infty}(\sigma^{(r)})^2 < \infty$. Another popular way for achieving (18) *is to use some advanced variance reduction techniques such as SVRG [40]—which may go beyond the scope of this paper and thus is left out of the discussion. Also notice that as the convergence analysis is pessimistic, in practice constant minibatch size works fairly well—as we will see soon.*

5 An Adaptive Stepsize Scheme

One may have noticed that the convergence theories in Propositions 1-2 do not specify the sequence $\alpha^{(r)}$ except two constraints as in Assumption 1. This oftentimes gives rise to agonizing tuning experience for practitioners when implementing stochastic algorithms.

Recently, a series of algorithms were proposed in the machine learning community for adaptive stepsize scheduling when training deep neural networks [46–48]. Most of these works are variants of the `Adagrad` algorithm [33]. The insight of `Adagrad` can be understood as follows: If one optimization variable has been heavily updated before, then it is given a smaller stepsize for the current iteration (and a larger stepsize otherwise). This way, all the optimization variables can be updated in a balanced manner. `Adagrad` was proposed for single-block algorithms, and this simple strategy also admits many provable benefits under the context of convex optimization [33]. For our multi-block nonconvex problem, we extend the idea and propose the following updating rule: In iteration r, if $\xi^{(r)} = n$, then, for all $i \in \{1, \ldots, I_n\}$ and all $f \in \{1, \ldots, F\}$, we have

$$[\boldsymbol{\eta}_{(n)}^{(r)}]_{i,f} \leftarrow \frac{\eta}{\left(b + \sum_{t=0}^{r-1}[\boldsymbol{G}_{(n)}^{(t)}]_{i,f}^2\right)^{1/2+\epsilon}}, \tag{19a}$$

$$\boldsymbol{A}_{(n)}^{(r+1)} \leftarrow \boldsymbol{A}_{(n)}^{(r)} - \boldsymbol{\eta}_{(n)}^{(r)} \circledast \boldsymbol{G}_{(n)}^{(r)}, \tag{19b}$$

$$\boldsymbol{A}_{(n')}^{(r+1)} \leftarrow \boldsymbol{A}_{(n')}^{(r)}, \tag{19c}$$

where $\eta, b, \epsilon > 0$. The `Adagrad` version of block-randomized CPD algorithm is very simple to implement. The algorithm is summarized in Algorithm 2, which is named `AdaCPD`.

As one will soon see, such a simple stepsize strategy is very robust to a large number of scenarios under test—i.e., in most of the cases, `AdaCPD` performs well without tuning the stepsize schedule. In addition, the `AdaCPD` algorithm works well for both the constrained and unconstrained case.

Algorithm 2: `AdaCPD`

input : N-way tensor $\underline{\boldsymbol{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$; rank F; sample size B , 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 $|\mathcal{F}_n| = B$;
  form the stochastic gradient $\boldsymbol{G}^{(r)} \leftarrow (10)$;
  determine the step size $\boldsymbol{\eta}_{(n)}^{(r)} \leftarrow (19a)$
  update $\boldsymbol{A}_{(n)}^{(r+1)} \leftarrow (19b)$, $\boldsymbol{A}_{(n')}^{(r+1)} \leftarrow \boldsymbol{A}_{(n')}^{(r)}$ for $n' \neq n$;
  $r \leftarrow r + 1$;
until *some stopping criterion is reached*;

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

Proving convergence for nonconvex `Adagrad`-like algorithms is quite challenging [49,50]. In this work, we show that the following holds:

Proposition 3 *Assume $h_n(\cdot) = 0$ for all n, and that $\mathsf{Pr}(\xi^{(r)} = n) > 0$ for all r and n. Under the Assumptions 1-2, the solution sequence produced by `AdaCPD` satisfies*

$$\mathsf{Pr}\left(\lim_{r\to\infty} \|\nabla_{\boldsymbol{A}_{(n)}} f(\boldsymbol{\theta}^{(r)})\|^2 = 0\right) = 1.$$

Proposition 3 asserts that the algorithm converges almost surely. The proof is relegated to Appendix D in the *supplementary materials* due to page limitations. Our proof extends the idea from a recent paper [49] that focuses on using `Adagrad` for solving single-block nonconvex problems. As mentioned, our two-level sampling strategy makes our algorithm very similar to single-block SGD with a scaled gradient estimation (cf. Appendix A), and thus with careful modifications the key proof techniques in [49] goes through. Nevertheless, we detail the proof for being self-containing.

6 Numerical Results

In this section, we use simulations and real-data experiments to showcase the effectiveness of the proposed algorithm.

6.1 Synthetic Data Simulations

6.1.1 Data Generation

Throughout this subsection, we use synthetic third-order tensors (i.e., $N = 3$) whose latent factors are drawn from i.i.d. uniform distribution between 0 and 1—unless otherwise specified. This way, large and *dense* tensors can be created. For simplicity, we set $I_n = I$ for all n and test the algorithms on tensors having different I_n's and F's. In some simulations, we also consider CPD for noisy tensors, i.e., factoring data tensors that have the following signal model:

$$\underline{\boldsymbol{Y}} = \underline{\boldsymbol{X}} + \underline{\boldsymbol{N}},$$

where $\underline{\boldsymbol{X}}$ is the noiseless low-rank tensor and $\underline{\boldsymbol{N}}$ denotes the additive noise. We use zero-mean i.i.d. Gaussian noise with variance σ_N^2 in our simulations, and the signal-to-noise ratio (SNR) (in dB) is defined as $\mathsf{SNR} = 10\log_{10}\left(\frac{\frac{1}{\prod_{n=1}^N I_n}\|\underline{\boldsymbol{X}}\|^2}{\sigma_N^2}\right)$.

6.1.2 Baselines

A number of baseline algorithms are employed as benchmarks. Specifically, we mainly use the `AO-ADMM` algorithm [51] and the `APG` algorithm [14] as our baselines since they are the most flexible algorithms with the ability of handling many different regularizations and constraints. We also present the results output by the `CPRAND` algorithm [25]. Note that we are preliminarily interested in constrained/regularized CPD. Because `CPRAND` operates without constraints, the comparison is not entirely fair (e.g., `CPRAND` can potentially attain smaller cost values since it has a much larger feasible set). Nevertheless, we employ it as a benchmark since it uses the same fiber sampling strategy as ours. All the algorithms are initialized with the same random initialization; i.e., $\boldsymbol{A}_{(0)}$'s entries follow the uniform distribution between 0 and 1.

6.1.3 Parameter Setting

For `BrasCPD`, we set the stepsize to be

$$\alpha^{(r)} = \frac{\alpha}{r^{\beta}}, \tag{20}$$

where r is the number of iterations, $\beta = 10^{-6}$ and α typically takes a value in between 0.001 and 0.1, and we try multiple choices of α in our simulations. The batch size $|\mathcal{F}_n|$ is set to be below 25, which will be specified later. For `AdaCPD`, we fix $\beta = 10^{-6}$ and $\eta = 1$ for all the simulations. For `CPRAND`, we follow the instruction in the original paper [25] and sample $10F\log_2 F$ fibers for each update.

6.1.4 Performance Metrics

To measure the performance, we employ two metrics. The first one is the value of the cost function, i.e., $\mathsf{cost} = (1/\prod_{n=1}^N I_n) \times f(\boldsymbol{\theta}^{(r)})$. 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 [52, 53]:

$$\mathsf{MSE} = \min_{\pi(f)\in\{1,\ldots,F\}} \frac{1}{F}\sum_{f=1}^{F}\left\| \frac{\boldsymbol{A}_{(n)}(:,\pi(f))}{\|\boldsymbol{A}_{(n)}(:,\pi(f))\|_2} - \frac{\widehat{\boldsymbol{A}}_{(n)}(:,f)}{\|\widehat{\boldsymbol{A}}_{(n)}(:,f)\|_2}\right\|_2^2 \tag{21}$$

where $\widehat{\boldsymbol{A}}_{(n)}$ denotes the estimate of $\boldsymbol{A}_{(n)}$ and $\pi(f)$'s are under the constraint $\{\pi(1), \ldots, \pi(F)\} = \{1, \ldots, F\}$—which is used to fix the intrinsic column permutation in CPD.

Since the algorithms under test have very different operations and subproblem-solving strategies, it may be challenging to find an exactly unified complexity measure. In this section, we show the peformance of the algorithms against the number of MTTKRP operations $\boldsymbol{H}_{(n)}^{\top}\boldsymbol{X}_{(n)}$ used, since $\boldsymbol{H}_{(n)}^{\top}\boldsymbol{X}_{(n)}$ is the most costly step that dominates the complexity of all the algorithms under

comparison. All the simulations are conducted in Matlab. The results are averaged from ten random trials with different tensors.

6.2 Results

Fig. 1 in Sec. 1 has shown the MSE performance of the algorithms in a relatively small-size example, where $I_n = I = 100$, $F = 10$ and the nonnegativity constraints are used in the algorithms. In that simulation, we use $|\mathcal{F}_n| = 20$ so that every 500 iterations of the proposed algorithm compute a full MTTKRP. One can see that for this relatively easy case, all the algorithms can reach a good estimation accuracy for the latent factors. Nevertheless, the proposed methods exhibit remarkably higher efficiency.

Fig. 3 shows the average MSEs of the estimated latent factors by the algorithms under a much larger scale simulation, where $I_1 = I_2 = I_3 = 300$ and $F = 100$. We set $|F_n| = 18$ so that the proposed algorithms use 5,000 iterations to compute a full MTTKRP. All the algorithms use nonnegativity constraints except `CPRAND`. There are several observations in order: First, the stochastic algorithms (i.e., `BrasCPD`, `AdaCPD`, and `CPRAND`) are much more efficient relative to the deterministic algorithms (`AO-ADMM` and `APG`). After 30 MTTKRPs computed, the stochastic algorithms often have reached a reasonable level of MSE. This is indeed remarkable, since 30 MTTKRPs are roughly equivalent to 10 iterations of `AO-ADMM` and `APG`. Second, two of the proposed stochastic algorithms largely outperforms `CPRAND`. In particular, `BrasCPD` with $\alpha = 0.1$ gives the most promising performance. However, the performance of `BrasCPD` is affected a bit significantly by the parameters α. One can see that using $\alpha = 0.05$ and $\alpha = 0.01$ the algorithm does not give so promising results under this setting. Third, `AdaCPD` yields the second lowest MSEs, but its MSE curve starts saturating and decreases slower after it reaches MSE=10^{-3}. This is understandable, since the 'size' of $\boldsymbol{\eta}$ is shrinking after each iteration and thus the stepsize could vanish after a large number of iterations. Nevertheless, MSE=10^{-3} is already very satisfactory, and `AdaCPD` shows surprising robustness to changing scenarios, without changing any setup in its stepsize scheduling strategy. Fig. 4 shows the cost values against the number of full MTTKRPs computed, which is consistent to what we observed in Fig. 3.

Table 2 shows the MSEs and cost values of output by the algorithms when the tensor rank varies under $I = 300$. All the algorithms are stopped after 30 full MTTKRPs are used. One can see that `BrasCPD` in general exhibits the lowest MSEs if a proper α is chosen, under the employed stepsize schedule in (20). However, one can see that when F changes, there is a risk that `BrasCPD` runs into numerical issues and yields unbounded solutions. This suggests that `BrasCPD` may need extra care for tuning its stepsize. On the other hand, `AdaCPD` always outputs reasonably good results. The MSEs output by `AdaCPD` is slightly higher relative to `BrasCPD`, but is much lower compared to those of the baslines. More importantly, `AdaCPD` runs without tuning the stepsize parameters—which shows the power of the adaptive stepsize scheduling strategy.

Tables 4-5 show the performance of the algorithms under different SNRs. Except for adding noise, other settings are the same as those in Fig. 3. In a noisy environment, the ability of handling constraints/regularizations is essential for a CPD algorithm, since prior information on the latent factors can help improve estimation accuracy. Table 4 and Table 5 test the cases where $\boldsymbol{A}_{(n)}$ is elementwise nonnegative and the columns of $\boldsymbol{A}_{(n)}$ reside in a scaled version of the probability simplex, respectively. One can see from the two tables that both `BrasCPD` (with a proper α) and `AdaCPD` work very well. In Table 5, one can see that `BrasCPD` again shows its sensitivity to the choice of α, with $\alpha = 0.1$ and 0.05 actually not working. We also note that when the SNR is low, `CPRAND`

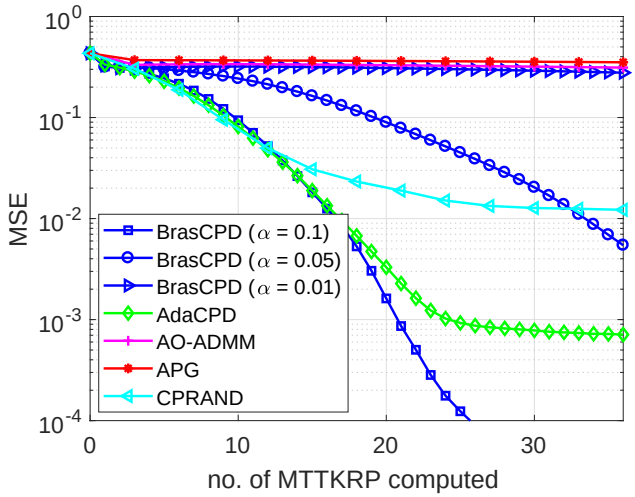

Figure 3: MSE of the algorithms. $I_1 = I_2 = I_3 = 300$ and $F = 100$. $\boldsymbol{A}_{(n)} \geq \mathbf{0}$.

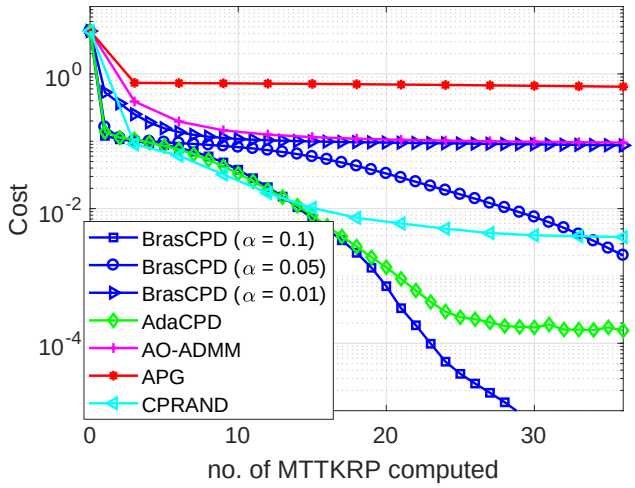

Figure 4: cost of the algorithms. $I_1 = I_2 = I_3 = 300$ and $F = 100$. $\boldsymbol{A}_{(n)} \geq \mathbf{0}$.

Table 2: MSEs of the estimated latent factors by the algorithms under different F; $I = 300$; all the algorithms are stopped after computing 30 MTTKRPs. "NaN" means the algorithm outputs unbounded solutions. $\boldsymbol{A}_{(n)} \geq \mathbf{0}$.

Algorithm	Metric	F			
		100	200	300	400
BrasCPD (α=0.1)	MSE	$\mathbf{8.4375 \times 10^{-5}}$	NaN	NaN	NaN
BrasCPD (α =0.05)	MSE	0.0126	**0.0494**	**0.0894**	NaN
BrasCPD (α =0.01)	MSE	0.2882	0.3142	0.3235	0.3239
AdaCPD	MSE	0.0016	0.1247	0.1467	**0.2382**
AO-ADMM	MSE	0.3190	0.3124	0.3093	0.3033
APG	MSE	0.3574	0.3527	0.3538	0.3545
CPRAND	MSE	0.0056	0.0967	0.2115	0.2404
BrasCPD (α =0.1)	Cost	$\mathbf{2.3759 \times 10^{-5}}$	NaN	NaN	NaN
BrasCPD (α =0.05)	Cost	0.0046	**0.0397**	**0.1162**	NaN
BrasCPD (α =0.01)	Cost	0.0903	0.1832	0.2687	0.3461
AdaCPD	Cost	4.8050×10^{-4}	0.1555	0.1684	**0.2144**
AO-ADMM	Cost	0.0990	0.1952	0.2800	0.3520
APG	Cost	0.6649	1.3629	2.0664	2.7707
CPRAND	Cost	0.0018	0.0691	0.1842	0.2481

Table 3: Performance of the algorithms under various I's, $F = 100$. $\boldsymbol{A}_{(n)} \geq \mathbf{0}$; all the algorithms are stopped after computing 30 MTTKRPs. $\boldsymbol{A}_{(n)} \geq \mathbf{0}$.

Algorithm	Metric	I			
		100	200	300	400
BrasCPD (α=0.1)	MSE	0.2432	0.0474	$\mathbf{8.4375 \times 10^{-5}}$	$\mathbf{1.2052 \times 10^{-9}}$
BrasCPD (α =0.05)	MSE	0.2724	0.2000	0.0126	1.0631×10^{-4}
BrasCPD (α =0.01)	MSE	0.2906	0.3086	0.2882	0.2127
AdaCPD	MSE	0.2214	**0.0121**	0.0016	1.0068×10^{-4}
AO-ADMM	MSE	0.2561	0.3171	0.3190	0.3235
APG	MSE	0.3107	0.3459	0.3574	0.3635
CPRAND	MSE	**0.1857**	0.0459	0.0056	0.0025
BrasCPD (α =0.1)	Cost	0.0795	0.0179	$\mathbf{2.3759 \times 10^{-5}}$	$\mathbf{3.5023 \times 10^{-10}}$
BrasCPD (α =0.05)	Cost	0.0862	0.0668	0.0046	2.8758×10^{-5}
BrasCPD (α =0.01)	Cost	0.1453	0.0981	0.0903	0.2127
AdaCPD	Cost	0.0814	**0.0058**	4.8050×10^{-4}	3.5958×10^{-5}
AO-ADMM	Cost	0.0843	0.0957	0.0990	0.1008
APG	Cost	0.5936	0.6450	0.6649	0.6776
CPRAND	Cost	**0.0566**	0.0136	0.0018	0.0011

Table 4: Performance of the algorithms under various SNRs; all the algorithms are stopped after computing 30 MTTKRPs. $I_1 = I_2 = I_3 = 300$, $F = 100$. $\boldsymbol{A}_{(n)} \geq \mathbf{0}$.

Algorithm	Metric	SNR			
		10	20	30	40
BrasCPD (α=0.1)	MSE	0.3685	0.0225	**0.0024**	**0.0003**
BrasCPD (α =0.05)	MSE	0.2962	0.0198	0.0066	0.0044
BrasCPD (α =0.01)	MSE	**0.3125**	0.2823	0.2774	0.2758
AdaCPD	MSE	0.3285	**0.0192**	0.0025	0.0004
AO-ADMM	MSE	0.3330	0.3135	0.3118	0.3101
APG	MSE	0.3524	0.3521	0.3521	0.3520
CPRAND	MSE	1.6047	0.0367	0.0104	0.0100
BrasCPD (α =0.1)	Cost	1.9627	0.2081	0.0212	**0.0021**
BrasCPD (α =0.05)	Cost	0.9086	0.0918	**0.0110**	0.0025
BrasCPD (α =0.01)	Cost	0.2812	0.1058	0.0885	0.0865
AdaCPD	Cost	0.3137	**0.0671**	0.0155	0.0024
AO-ADMM	Cost	**0.1533**	0.0999	0.0954	0.0948
APG	Cost	0.6445	0.6441	0.6430	0.6435
CPRAND	Cost	0.8038	0.0811	0.0100	0.0039

Table 5: Performance of the algorithms under various SNRs after computing 30 MTTKRPs. $I_1 = I_2 = I_3 = 300$, $F = 100$. $\mathbf{1}^\top \boldsymbol{A}_{(n)} = \rho \mathbf{1}^\top$, $\boldsymbol{A}_{(n)} \geq \mathbf{0}$. $\rho = 300$.

Algorithm	Metric	SNR			
		10	20	30	40
BrasCPD (α =0.1)	MSE	0.4697	0.4423	0.3956	0.4320
BrasCPD (α =0.05)	MSE	0.4443	0.4267	0.4135	0.4146
BrasCPD (α =0.01)	MSE	0.3940	**0.0335**	0.0033	0.0003
AdaCPD	MSE	**0.2983**	0.0611	**0.0011**	**0.0002**
AO-ADMM	MSE	0.3206	0.2996	0.2973	0.2972
APG	MSE	0.2761	0.2760	0.2760	0.2760
CPRAND	MSE	1.6020	0.0466	0.0045	**0.0112**
BrasCPD (α =0.1)	Cost	14274.5141	12059.7192	7386.9652	10944.0721
BrasCPD (α =0.05)	Cost	11424.7030	10159.3117	9059.4911	9152.1151
BrasCPD (α =0.01)	Cost	229.6424	24.8565	2.5698	0.2571
AdaCPD	Cost	**14.8627**	**3.4755**	0.5916	**0.1318**
AO-ADMM	Cost	9.6097	6.1642	5.8643	5.8359
APG	Cost	36.5461	36.5095	36.5059	36.5055
CPRAND	Cost	51.5269	5.2663	**0.5413**	0.2570

is not as competitive, perhaps because it cannot use constraints to incorporate prior information of the $\boldsymbol{A}_{(n)}$'s—this also shows the importance of being able to handle various constraints.

6.3 Real-Data Experiment

In this subsection, we test our algorithm on a constrained tensor decomposition problem; i.e., we apply the proposed `BrasCPD` and `AdaCPD` to factor hyperspectral images. Hyperspectral images (HSIs) are special images with pixels measured at a large number of wavelengths. Hence, an HSI is usually stored as a third-order tensor with two spatial coordinates and one spectral coordinate. HSIs are dense tensors and thus are suitable for testing the proposed algorithms. We use sub-images of the Indian Pines dataset that has a size of $145 \times 145 \times 220$ and the Pavia University dataset[1] that has a size of $610 \times 340 \times 103$.

Tables 6-7 show the cost values of the nonnegativity constrained optimization algorithms under different ranks, after computing 10 MTTKRPs for all three modes, which corresponds to 10 iterations for `AO-ADMM` and `APG` (we use this "all-mode MTTKRP" in this section since the tensors are unsymmetrical and thus single-mode MTTKRPs cannot be directly translated to iterations in batch algorithms). One can see that the proposed algorithms show the same merits as we have seen in the simulations: `BrasCPD` can exhibit very competitive performance when α is properly chosen (e.g., when $F = 10$ and $\alpha = 5$ for the Indian Pines dataset); in addition, `AdaCPD` gives consistently good performance without tuning the stepsize manually. Particularly, on the Pavia University dataset, `AdaCPD` gives much lower cost values compared to other algorithms. Fig. 5 shows how the cost values change along with the iterations on the Pavia University data using $F = 200$.

Table 6: Performance of the algorithms on the Indian Pines dataset under different F's.

Algorithm	Metric	F			
		10	20	30	40
`BrasCPD` (α =4)	Cost	$\mathbf{6.8343 \times 10^{-4}}$	4.7777×10^{-4}	$\mathbf{3.4738 \times 10^{-4}}$	$\mathbf{2.9053 \times 10^{-4}}$
`BrasCPD` (α =3)	Cost	6.8507×10^{-4}	4.8550×10^{-4}	4.1556×10^{-4}	3.1278×10^{-4}
`BrasCPD` (α =2)	Cost	6.9877×10^{-4}	5.7753×10^{-4}	5.4205×10^{-4}	4.2504×10^{-4}
`AdaCPD`	Cost	7.0677×10^{-4}	$\mathbf{4.6180 \times 10^{-4}}$	3.5328×10^{-4}	2.9848×10^{-4}
`AO-ADMM`	Cost	7.2503×10^{-4}	5.5708×10^{-4}	5.1489×10^{-4}	5.1505×10^{-4}
`APG`	Cost	1.9392×10^{-3}	1.8952×10^{-3}	1.8818×10^{-3}	1.8675×10^{-3}

Table 7: Performance of the algorithms on the Indian Pines dataset under different F's.

Algorithm	Metric	F	
		100	200
`BrasCPD` (α =0.5)	Cost	2.7193×10^{-3}	2.5275×10^{-3}
`BrasCPD` (α =0.3)	Cost	3.6496×10^{-3}	5.3453×10^{-3}
`BrasCPD` (α =0.1)	Cost	6.4221×10^{-3}	5.7509×10^{-3}
`AdaCPD`	Cost	$\mathbf{1.7269 \times 10^{-3}}$	$\mathbf{9.0080 \times 10^{-4}}$
`AO-ADMM`	Cost	6.2494×10^{-3}	4.5879×10^{-3}
`APG`	Cost	7.2966×10^{-3}	7.2647×10^{-3}

[1]Both datasets are available online: `http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes`

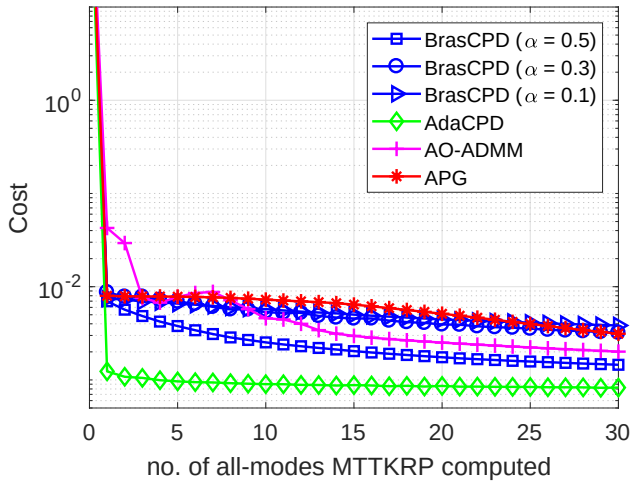

Figure 5: No. of all-mode MTTKRPs v.s. cost values output by the algorithms when applied to the Pavia University dataset. $F = 200$. Nonnegativity constraint is added.

7 Conclusion

To conclude, we proposed a block-randomized stochastic proximal gradient based CPD algorithmic framework for large-scale dense tensors. The framework works under a doubly stochastic manner, which randomly selects a mode and then samples a set of fibers for updating the associated latent factor. The framework has a series of nice features including being able to quickly improve estimation accuracy of the latent factors, being flexible with incorporating constraints and regularizations, and having rigorous convergence guarantees. We also proposed a practical and effective adaptive stepsize scheduling method that is reminiscent of recent advances in neural network training algorithms. Simulations and real-data experiments show that the proposed algorithms outperform a number of state-of-art constrained CPD algorithms when dealing with large dense tensors.

A Connection between $\nabla f(\boldsymbol{\theta}^{(r)})$ and $\boldsymbol{G}^{(r)}$

Let us consider the following conditional expectation:

$$\begin{aligned}
\overline{\boldsymbol{G}}_{(n)}^{(r)} &= \mathbb{E}\big[\boldsymbol{G}_{(n)}^{(r)} \,|\, \mathcal{B}^{(r)}\big] = \mathbb{E}\big[\boldsymbol{G}_{(n)}^{(r)} \,|\, \{\boldsymbol{A}_{(n)}^{(r)}\}_{n=1}^{N}\big] \\
&= \mathbb{E}_{n'}\left[\frac{1}{\big(\frac{J_{n'}}{F}\big)}\big(\boldsymbol{A}_{(n')}^{(r)}\boldsymbol{H}_{(n')}^{\top}\boldsymbol{H}_{(n')} - \boldsymbol{X}_{(n')}^{\top}\boldsymbol{H}_{(n')}\big)\right] \\
&\overset{(a)}{=} \sum_{n'=1}^{N}\frac{\delta(n'-n)}{N\big(\frac{J_{n'}}{F}\big)}\big(\boldsymbol{A}_{(n')}^{(r)}\boldsymbol{H}_{(n')}^{\top}\boldsymbol{H}_{(n')} - \boldsymbol{X}_{(n')}^{\top}\boldsymbol{H}_{(n')}\big) \\
&= \frac{1}{N\big(\frac{J_n}{F}\big)}\big(\boldsymbol{A}_{(n)}^{(r)}\boldsymbol{H}_{(n)}^{\top}\boldsymbol{H}_{(n)} - \boldsymbol{X}_{(n)}^{\top}\boldsymbol{H}_{(n)}\big) \qquad (22)
\end{aligned}$$

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. $\boldsymbol{A}^{(r)}_{(n)}$. 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.

B Proof of Proposition 1

To show Proposition 1, we will need the following [54, Proposition 1.2.4]:

Lemma 1 *Let $\{a_t\}_t$ and $\{b_t\}_t$ be two nonnegative sequences such that b_t is bounded, $\sum_{t=0}^{\infty} a_t b_t$ converges and $\sum_{t=0}^{\infty} a_t$ diverges, then we have*

$$\lim_{t\to\infty} \inf b_t = 0.$$

To make our notations precise, let us denote $\xi^{(r)}$ as the random index of mode chosen at iteration r and subsequently $\mathcal{F}^{(r)}_{(\xi^{(r)})}$ is the random set of fibers chosen. Under Assumption 2, we have $\|\boldsymbol{H}^{(r)}_{(\xi^{(r)})}\|_2^2 \leq L^{(r)}_{(\xi^{(r)})}$ where $\boldsymbol{H}^{(r)}_{(\xi^{(r)})} = \odot_{n'=1,n'\neq\xi^{(r)}}^{N} \boldsymbol{A}^{(r)}_{(n')}$ and $L^{(r)}_{(\xi^{(r)})} < \infty$. Combining with Fact 1, we observe the following bound:

$$\begin{aligned}
&f(\boldsymbol{\theta}^{(r+1)}) - f(\boldsymbol{\theta}^{(r)}) \\
&\leq \left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}), \boldsymbol{A}^{(r+1)}_{(\xi^{(r)})} - \boldsymbol{A}^{(r)}_{(\xi^{(r)})} \right\rangle + \frac{L}{2} \left\| \boldsymbol{A}^{(r+1)}_{(\xi^{(r)})} - \boldsymbol{A}^{(r)}_{(\xi^{(r)})} \right\|_F^2 \\
&= -\alpha^{(r)} \left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}), \boldsymbol{G}^{(r)}_{(\xi^{(r)})} \right\rangle + \frac{(\alpha^{(r)})^2 L}{2} \left\| \boldsymbol{G}^{(r)}_{(\xi^{(r)})} \right\|_F^2,
\end{aligned}$$

where we denote

$$L = \max_{r=0,\ldots,\infty} L^{(r)}_{(\xi^{(r)})} < \infty,$$

since $\boldsymbol{A}^{(r)}_{(n)}$ is bounded for all iterations.

Taking expectation conditioned on the filtration $\mathcal{B}^{(r)}$ and the chosen mode index $\xi^{(r)}$, we have

$$\begin{aligned}
&\mathbb{E}\left[f(\boldsymbol{\theta}^{(r+1)}) \,|\, \mathcal{B}^{(r)}, \xi^{(r)} \right] - f(\boldsymbol{\theta}^{(r)}) \\
&\leq -\alpha^{(r)} \left\| \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}) \right\|^2 \\
&\qquad + \frac{(\alpha^{(r)})^2 L}{2} \mathbb{E}\left[\left\| \boldsymbol{G}^{(r)}_{(\xi^{(r)})} \right\|_F^2 \,|\, \mathcal{B}^{(r)}, \xi^{(r)} \right] \\
&\leq -\alpha^{(r)} \left\| \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}) \right\|^2 + \frac{(\alpha^{(r)})^2 L M}{2}.
\end{aligned} \tag{23}$$

where the first inequality used the assumption that $L^{(r)}_{(\xi^{(r)})} \leq L$ and Fact 2, and the second inequality is again a consequence of Assumption 2, as we observe:

$$\begin{aligned}
\left\| \boldsymbol{G}^{(r)}_{(\xi^{(r)})} \right\| = \frac{1}{|\mathcal{F}_n|} \Big\| &\boldsymbol{A}^{(r)}_{(\xi^{(r)})} (\boldsymbol{H}^{(r)}_{(\xi^{(r)})}(\mathcal{F}_n))^\top \boldsymbol{H}^{(r)}_{(\xi^{(r)})}(\mathcal{F}_n) \\
&- \boldsymbol{X}^\top_{(\xi^{(r)})}(\mathcal{F}_n) \boldsymbol{H}^{(r)}_{(\xi^{(r)})}(\mathcal{F}_n) \Big\|
\end{aligned} \tag{24}$$

where in the right hand side we have dropped the superscript for $n^{(r)}$ for simplicity. As $\boldsymbol{X}_{(n)}$ is bounded for all n, and all the $\boldsymbol{A}_{(n)}^{(r)}$ are bounded under Assumption 2, we have $\|\boldsymbol{G}_{(\xi^{(r)})}^{(r)}\|^2 \leq M$ for all n, r and some $M < \infty$. Now, taking the expectation w.r.t. $\xi^{(r)}$ yields

$$\begin{aligned}
&\mathbb{E}_{\xi^{(r)}}\left[f(\boldsymbol{\theta}^{(r+1)}) \,|\, \mathcal{B}^{(r)}\right] - f(\boldsymbol{\theta}^{(r)}) \\
&\leq -\alpha^{(r)}\mathbb{E}_{\xi^{(r)}}\left[\left\|\nabla_{\boldsymbol{A}_{(n^{(r)})}} f(\boldsymbol{\theta}^{(r)})\right\|^2\right] + \frac{(\alpha^{(r)})^2 ML}{2}.
\end{aligned} \tag{25}$$

Finally, taking the total expectation, we have

$$\begin{aligned}
&\mathbb{E}\left[f(\boldsymbol{\theta}^{(r+1)})\right] - \mathbb{E}\left[f(\boldsymbol{\theta}^{(r)})\right] \\
&\leq -\alpha^{(r)}\mathbb{E}\left[\mathbb{E}_{\xi^{(r)}}\left[\left\|\nabla_{\boldsymbol{A}_{(n^{(r)})}} f(\boldsymbol{\theta}^{(r)})\right\|^2\right]\right] + \frac{(\alpha^{(r)})^2 ML}{2}.
\end{aligned} \tag{26}$$

Summing up the above from $t = 0$ to $t = r$, we have

$$\begin{aligned}
&\mathbb{E}\left[f(\boldsymbol{\theta}^{(t+1)})\right] - f(\boldsymbol{\theta}^{(0)}) \\
&\leq \sum_{t=0}^{r} -\alpha^{(t)}\mathbb{E}\left[\mathbb{E}_{\xi^{(t)}}\left[\left\|\nabla_{\boldsymbol{A}_{(n^{(t)})}} f(\boldsymbol{\theta}^{(t)})\right\|^2\right]\right] \\
&\quad + \sum_{t=0}^{r} \frac{(\alpha^{(t)})^2 ML}{2}.
\end{aligned} \tag{27}$$

Taking $r \to \infty$, the above implies that

$$\begin{aligned}
&\sum_{r=0}^{\infty} \alpha^{(r)}\mathbb{E}\left[\mathbb{E}_{\xi^{(r)}}\left[\left\|\nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)})\right\|^2\right]\right] \\
&\qquad \leq f(\boldsymbol{\theta}^{(0)}) - f(\boldsymbol{\theta}^{(\star)}) + \sum_{r=0}^{\infty} \frac{(\alpha^{(r)})^2 ML}{2},
\end{aligned} \tag{28}$$

where $f(\boldsymbol{\theta}^{(\star)})$ denotes the global optimal value. Note that the right hand side above is bounded from above because $\sum_{r=0}^{\infty}(\alpha^{(r)})^2 < \infty$. Hence, using Lemma 1, we can conclude that

$$\liminf_{r\to 0} \mathbb{E}\left[\mathbb{E}_{\xi^{(r)}}\left[\left\|\nabla_{\boldsymbol{A}_{(n^{(r)})}} f(\boldsymbol{\theta}^{(r)})\right\|^2\right]\right] = 0.$$

Finally, we conclude:

$$\liminf_{r\to 0} \mathbb{E}\left[\left\|\nabla f(\boldsymbol{\theta}^{(r)})\right\|^2\right] = 0,$$

since

$$\mathbb{E}_{\xi^{(r)}}\left[\left\|\nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)})\right\|^2\right] = \frac{1}{N}\sum_{n=1}^{N}\left\|\nabla_{\boldsymbol{A}_{(n)}} f(\boldsymbol{\theta}^{(r)})\right\|^2$$

by the fundamental theorem of expectation.

C Proof of Proposition 2

C.1 Preliminaries

For the constrained case, let us denote $\Phi(\boldsymbol{\theta}) = f(\boldsymbol{\theta}) + \sum_{n=1}^{N} h_n(\boldsymbol{\theta})$ as the objective function. Unlike the unconstrained case where we measure convergence via observing if the gradient vanishes, the optimality condition of the constrained case is a bit more complicated. Here, the idea is to observe the "generalized gradient". To be specific, consider the following optimization problem

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \;\; f(\boldsymbol{\theta}) + h(\boldsymbol{\theta}),$$

where $f(\boldsymbol{\theta})$ is continuously differentiable while h is convex but possibly nonsmooth. The deterministic proximal gradient algorithm for handling this problem is as follows:

$$\boldsymbol{\theta}^{(r+1)} \leftarrow \mathsf{Prox}_h\left(\boldsymbol{\theta}^{(r)} - \alpha^{(r)} \nabla f(\boldsymbol{\theta}^{(r)})\right).$$

Define $\boldsymbol{P}^{(r)} = \frac{1}{\alpha^{(r)}}\left(\boldsymbol{\theta}^{(r+1)} - \boldsymbol{\theta}^{(r)}\right)$, the update can also be represented as $\boldsymbol{\theta}^{(r+1)} \leftarrow \boldsymbol{\theta}^{(r)} - \alpha^{(r)}\boldsymbol{P}^{(r)}$, which is analogous to the gradient descent algorithm. It can be shown that $\boldsymbol{P}^{(r)} = \mathbf{0}$ implies that the necessary optimality condition is satisfied, and thus $\boldsymbol{P}^{(r)}$ can be considered as a "generalized gradient".

In our case, let us denote $\Phi(\boldsymbol{\theta}) = f(\boldsymbol{\theta}) + \sum_{n=1}^{N} h_n(\boldsymbol{\theta})$ as the objective function. Our optimality condition amounts to $\boldsymbol{P}_{(n)}^{(r)} = \mathbf{0}, \quad \forall\, n$, where

$$\boldsymbol{P}_{(n)}^{(r)} = \frac{1}{\alpha^{(r)}}\left(\boldsymbol{A}_{(n)}^{(r+1)} - \mathsf{Prox}_{h_n}\left(\boldsymbol{A}_{(n)}^{(r)} - \alpha^{(r)}\nabla_{\boldsymbol{A}_{(n)}} f(\boldsymbol{\theta}^{(r)})\right)\right);$$

i.e., the optimality condition is satisfied in a blockwise fashion [30, 32]. Hence, our goal of this section is to show that $\mathbb{E}\left[\boldsymbol{P}_{(n)}^{(r)}\right]$ for all n vanishes when r goes to infinity.

C.2 Proof

Our update is equivalent to the following:

$$\boldsymbol{A}_{(n)}^{(r+1)} \leftarrow \arg\min_{\boldsymbol{A}_{(n)}} \; \left\langle \boldsymbol{G}_{(n)}^{(r)}, \boldsymbol{A}_{(n)} - \boldsymbol{A}_{(n)}^{(r)} \right\rangle + \frac{1}{2\alpha^{(r)}}\left\|\boldsymbol{A}_{(n)} - \boldsymbol{A}_{(n)}^{(r)}\right\|^2 + h_n(\boldsymbol{A}_{(n)}) \tag{29}$$

for a randomly selected n, which is a proximity operator. For a given $\xi^{(r)}$, we have

$$\begin{aligned} & h_{\xi^{(r)}}\left(\boldsymbol{A}_{(\xi^{(r)})}^{(r+1)}\right) - h_{\xi^{(r)}}\left(\boldsymbol{A}_{(\xi^{(r)})}^{(r)}\right) \\ \leq & -\left\langle \boldsymbol{G}_{(\xi^{(r)})}^{(r)}, \boldsymbol{A}_{(\xi^{(r)})}^{(r+1)} - \boldsymbol{A}_{(\xi^{(r)})}^{(r)} \right\rangle - \frac{1}{2\alpha^{(r)}}\left\|\boldsymbol{A}_{(\xi^{(r)})}^{(r+1)} - \boldsymbol{A}_{(\xi^{(r)})}^{(r)}\right\|^2 \end{aligned}$$

by the optimality of $\boldsymbol{A}_{(\xi^{(r)})}^{(r+1)}$ for solving Problem (29).

By the block Lipschitz continuity of the smooth part (cf. Fact 1), we have

$$f(\boldsymbol{\theta}^{(r+1)}) - f(\boldsymbol{\theta}^{(r)}) \leq \left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}), \boldsymbol{A}_{(\xi^{(r)})}^{(r+1)} - \boldsymbol{A}_{(\xi^{(r)})}^{(r)} \right\rangle + \frac{L_{(\xi^{(r)})}^{(r)}}{2} \left\| \boldsymbol{A}_{(\xi^{(r)})}^{(r+1)} - \boldsymbol{A}_{(\xi^{(r)})}^{(r)} \right\|^2,$$

where f denotes the smooth part in the objective function and

$$L_{(\xi^{(r)})}^{(r)} = \lambda_{\max} \left(\left(\boldsymbol{H}_{(\xi^{(r)})}^{(r)} \right)^\top \boldsymbol{H}_{(\xi^{(r)})}^{(r)} \right) \leq L.$$

Combining the two inequalities, we have

$$\Phi(\boldsymbol{\theta}^{(r+1)}) \leq \Phi(\boldsymbol{\theta}^{(r)}) - \alpha^{(r)} \left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}) - \boldsymbol{G}_{(\xi^{(r)})}^{(r)}, \boldsymbol{p}_{(\xi^{(r)})}^{(r)} \right\rangle + \left(\frac{L(\alpha^{(r)})^2}{2} - \frac{\alpha^{(r)}}{2} \right) \|\boldsymbol{p}_{(\xi^{(r)})}^{(r)}\|^2 \tag{30}$$

where we define

$$\boldsymbol{p}_{(\xi^{(r)})}^{(r)} = \frac{1}{\alpha^{(r)}} \left(\boldsymbol{A}_{(\xi^{(r)})}^{(r+1)} - \boldsymbol{A}_{(\xi^{(r)})}^{(r)} \right).$$

The inequality in (30) can be further written as

$$\begin{aligned} &\Phi(\boldsymbol{\theta}^{(r+1)}) - \Phi(\boldsymbol{\theta}^{(r)}) \\ &\leq -\alpha^{(r)} \left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}) - \boldsymbol{G}_{(\xi^{(r)})}^{(r)}, \boldsymbol{p}_{(\xi^{(r)})}^{(r)} - \boldsymbol{P}_{(\xi^{(r)})}^{(r)} \right\rangle \\ &\quad - \alpha^{(r)} \left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}) - \boldsymbol{G}_{(\xi^{(r)})}^{(r)}, \boldsymbol{P}_{(\xi^{(r)})}^{(r)} \right\rangle \\ &\quad + \left(\frac{L(\alpha^{(r)})^2}{2} - \frac{\alpha^{(r)}}{2} \right) \|\boldsymbol{p}_{(\xi^{(r)})}^{(r)}\|^2, \end{aligned} \tag{31}$$

Again, taking expectation conditioning on the filtration $\mathcal{B}^{(r)}$ and $\xi^{(r)}$, we have

$$\begin{aligned} &\mathbb{E}_{\zeta^{(r)}} \left[\Phi(\boldsymbol{\theta}^{(r+1)}) | \mathcal{B}^{(r)}, \xi^{(r)} \right] - \Phi(\boldsymbol{\theta}^{(r)}) \\ &\leq \alpha^{(r)} \mathbb{E}_{\zeta^{(r)}} \left[\left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}) - \boldsymbol{G}_{(\xi^{(r)})}^{(r)}, \boldsymbol{P}_{(\xi^{(r)})}^{(r)} - \boldsymbol{p}_{(\xi^{(r)})}^{(r)} \right\rangle | \mathcal{B}^{(r)}, \xi^{(r)} \right] \\ &\quad + \left(\frac{L(\alpha^{(r)})^2}{2} - \frac{\alpha^{(r)}}{2} \right) \mathbb{E}_{\zeta^{(r)}} \left[\left\| \boldsymbol{p}_{(\xi^{(r)})}^{(r)} \right\|^2 \middle| \mathcal{B}^{(r)}, \xi^{(r)} \right], \end{aligned} \tag{32}$$

i.e., the second term on the right hand side of (31) becomes zero because of Fact 2. The first term on the right hand side of (32) can be bounded via the following chain of inequalities:

$$\begin{aligned} &\mathbb{E}_{\zeta^{(r)}} \left[\left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}) - \boldsymbol{G}_{(\xi^{(r)})}^{(r)}, \boldsymbol{P}_{(\xi^{(r)})}^{(r)} - \boldsymbol{p}_{(\xi^{(r)})}^{(r)} \right\rangle \middle| \mathcal{B}^{(r)}, \xi^{(r)} \right] \\ &\leq \mathbb{E}_{\zeta^{(r)}} \left[\left\| \boldsymbol{\delta}^{(r)} \right\| \left\| \boldsymbol{P}_{(\xi^{(r)})}^{(r)} - \boldsymbol{p}_{(\xi^{(r)})}^{(r)} \right\| \;\middle|\; \mathcal{B}^{(r)}, \xi^{(r)} \right] \\ &\leq \mathbb{E}_{\zeta^{(r)}} \left[\|\boldsymbol{\delta}^{(r)}\|^2 \mid \mathcal{B}^{(r)}, \xi^{(r)} \right] \\ &\leq (\sigma^{(r)})^2 \end{aligned} \tag{33}$$

where for the first inequality we have applied the Cauchy-Schwartz inequality, and for the second inequality we have used the non-expansiveness of the proximal operator of convex $h_n(\cdot)$. Taking expectation w.r.t. $\xi_{(r)}$ and then total expectation on both sides of (32), we have

$$\begin{aligned}&\mathbb{E}\left[\Phi(\boldsymbol{\theta}^{(r+1)})\right]-\mathbb{E}\left[\Phi(\boldsymbol{\theta}^{(r)})\right] \\ &\leq \alpha^{(r)}(\sigma^{(r)})^2+\left(\frac{L(\alpha^{(r)})^2}{2}-\frac{\alpha^{(r)}}{2}\right)\mathbb{E}\left[\left\|\boldsymbol{p}^{(r)}_{(\xi^{(r)})}\right\|^2\right].\end{aligned} \tag{34}$$

Summing up through $t=0$ to $t=r-1$, we have

$$\begin{aligned}&\mathbb{E}\left[\Phi(\boldsymbol{\theta}^{(r)})\right]-\Phi(\boldsymbol{\theta}^{(0)}) \\ &\leq \sum_{t=0}^{r}\alpha^{(t)}(\sigma^{(t)})^2+\sum_{t=0}^{r}\left(\frac{L(\alpha^{(t)})^2}{2}-\frac{\alpha^{(t)}}{2}\right)\mathbb{E}\left[\left\|\boldsymbol{p}^{(t)}_{(\xi^{(t)})}\right\|^2\right].\end{aligned} \tag{35}$$

Since we have assumed $\alpha^{(r)}<1/L$, we have $\frac{L(\alpha^{(r)})^2}{2}-\frac{\alpha^{(r)}}{2}<0$. Therefore, we have

$$\begin{aligned}&\sum_{t=0}^{r}\left(\frac{\alpha^{(t)}}{2}-\frac{L(\alpha^{(t)})^2}{2}\right)\mathbb{E}\left[\left\|\boldsymbol{p}^{(t)}_{(\xi^{(t)})}\right\|^2\right]. \\ &\leq \Phi(\boldsymbol{\theta}^{(0)})-\Phi(\boldsymbol{\theta}^{(\star)})+\sum_{t=0}^{r}\alpha^{(t)}(\sigma^{(t)})^2\end{aligned} \tag{36}$$

Taking $r\to\infty$, and by the assumption that $\sum_{r=0}^{\infty}\alpha^{(r)}(\sigma^{(r)})^2<\infty$, we can conclude that

$$\lim_{r\to\infty}\inf\mathbb{E}\left[\left\|\boldsymbol{p}^{(r)}_{(\xi^{(r)})}\right\|^2\right]=0,$$

using Lemma 1.

One can see that

$$\begin{aligned}&\mathbb{E}\left[\left\|\boldsymbol{P}^{(r)}_{(\xi^{(r)})}\right\|^2\right]\leq 2\mathbb{E}\left[\left\|\boldsymbol{p}^{(r)}_{(\xi^{(r)})}\right\|^2\right]+2\mathbb{E}\left[\left\|\boldsymbol{p}^{(r)}_{(\xi^{(r)})}-\boldsymbol{P}^{(r)}_{(\xi^{(r)})}\right\|^2\right] \\ &\leq 2\mathbb{E}\left[\left\|\boldsymbol{p}^{(r)}_{(\xi^{(r)})}\right\|^2\right] \\ &\quad+2\mathbb{E}\left[\mathbb{E}_{\zeta^{(r)}}\left[\left\|\boldsymbol{G}^{(r)}_{(\xi^{(r)})}-\nabla_{\boldsymbol{A}_{(\xi^{(r)})}}f(\boldsymbol{\theta}^{(r)})\right\|^2\Big|\mathcal{B}^{(r)},\xi^{(r)}\right]\right] \\ &\leq 2\mathbb{E}\left[\left\|\boldsymbol{p}^{(r)}_{(\xi^{(r)})}\right\|^2\right]+2(\sigma^{(r)})^2.\end{aligned} \tag{37}$$

where the last inequality is obtained via applying the nonexpansive property again. Note that both terms on the right hand side converge to zero. Hence, this relationship implies that

$$\lim_{r\to\infty}\inf\mathbb{E}\left[\left\|\boldsymbol{P}^{(r)}_{(\xi^{(r)})}\right\|^2\right]=0.$$

Note that by our sampling strategy, we have

$$\mathbb{E}\left[\left\|\boldsymbol{P}^{(r)}_{(\xi^{(r)})}\right\|^2\right] = \mathbb{E}\left[\mathbb{E}_{\xi^{(r)}}\mathbb{E}_{\zeta^{(r)}}\left[\left\|\boldsymbol{P}^{(r)}_{(\xi^{(r)})}\right\|^2 \middle| \mathcal{B}^{(r)}, \xi^{(r)}\right]\right].$$

However, since $\boldsymbol{P}^{(r)}_{(\xi^{(r)})}$ is not affected by the random seed $\zeta^{(r)}$, we have

$$\begin{aligned}\mathbb{E}\left[\left\|\boldsymbol{P}^{(r)}_{(\xi^{(r)})}\right\|^2\right] &= \mathbb{E}\left[\mathbb{E}_{\xi^{(r)}}\left[\left\|\boldsymbol{P}^{(r)}_{(\xi^{(r)})}\right\|^2 \middle| \mathcal{B}^{(r)}\right]\right] \\ &= \mathbb{E}\left[\sum_{n=1}^{N}\frac{1}{N}\left\|\boldsymbol{P}^{(r)}_{(n)}\right\|^2\right].\end{aligned}$$

This proves the proposition.

D Proof of Proposition 3

The insight of the proof largely follows the technique for single-block `Adagrad` [49], with some careful modifications to multiple block updates. One will see that the block sampling strategy and the block-wise unbiased gradient estimation are key to apply the proof techniques developed in [49] to our case. To show convergence, let us first consider the following lemma:

Lemma 2 *[49] Let $a_0 > 0$, $a_i \geq 0$, $i = 1, \ldots, T$ and $\beta > 1$. Then, we have*

$$\sum_{t=1}^{T}\frac{a_t}{(a_0 + \sum_{i=1}^{t} a_i)^{\beta}} \leq \frac{1}{(\beta-1)a_0^{\beta-1}}.$$

The proof is simple and elegant; see [49, Lemma 4].

Lemma 3 *[49] Consider a random variable X. If $\mathbb{E}[X] < \infty$, then $\mathsf{Pr}(X < \infty) = 1$.*

Let us consider the block-wise again:

$$\begin{aligned}f(\boldsymbol{\theta}^{(r+1)}) \leq f(\boldsymbol{\theta}^{(r)}) &+ \left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}), \boldsymbol{A}^{(r+1)}_{(\xi^{(r)})} - \boldsymbol{A}^{(r)}_{(\xi^{(r)})} \right\rangle \\ &+ \frac{L^{(r)}_{\xi^{(r)}}}{2}\left\|\boldsymbol{A}^{(r+1)}_{(\xi^{(r)})} - \boldsymbol{A}^{(r)}_{(\xi^{(r)})}\right\|^2. \end{aligned} \tag{38}$$

Plugging in our update rule under `AdaCPD`, one can see that

$$\begin{aligned}f(\boldsymbol{\theta}^{(r+1)}) &\leq f(\boldsymbol{\theta}^{(r)}) + \left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}), -\boldsymbol{\eta}^{(r)}_{(\xi^{(r)})} \circledast \boldsymbol{G}^{(r)}_{(\xi^{(r)})} \right\rangle \\ &\quad + \frac{L^{(r)}_{\xi^{(r)}}}{2}\left\|\boldsymbol{\eta}^{(r)}_{(\xi^{(r)})} \circledast \boldsymbol{G}^{(r)}_{(\xi^{(r)})}\right\|^2 \\ &= f(\boldsymbol{\theta}^{(r)}) - \left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}), \boldsymbol{\eta}^{(r)}_{(\xi^{(r)})} \circledast \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}) \right\rangle \\ &\quad + \left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}), \boldsymbol{\eta}^{(r)}_{(\xi^{(r)})} \circledast \left(\nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}) - \boldsymbol{G}^{(r)}_{(\xi^{(r)})}\right) \right\rangle \\ &\quad + \frac{L^{(r)}_{\xi^{(r)}}}{2}\left\|\boldsymbol{\eta}^{(r)}_{(\xi^{(r)})} \circledast \boldsymbol{G}^{(r)}_{(\xi^{(r)})}\right\|^2 \end{aligned} \tag{39}$$

Taking expectation w.r.t. $\zeta^{(r)}$ (the random seed that is responsible for selecting fibers) conditioning on the filtration $\mathcal{B}^{(r)}$ and the selected block $\xi^{(r)}$, the middle term is zero—since the block stochastic gradient is unbiased [cf. Fact 2]. Hence, we have reached the following

$$\begin{aligned}\mathbb{E}_{\zeta^{(r)}}\left[f(\boldsymbol{\theta}^{(r+1)})|\mathcal{B}^{(r)},\xi^{(r)}\right] &\leq \mathbb{E}_{\zeta^{(r)}}\left[f(\boldsymbol{\theta}^{(r)})|\mathcal{B}^{(r)},\xi^{(r)}\right] \\ &- \mathbb{E}_{\zeta^{(r)}}\left[\left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}), \boldsymbol{\eta}^{(r)}_{(\xi^{(r)})} \circledast \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)})\right\rangle |\mathcal{B}^{(r)},\xi^{(r)}\right] \\ &+ \frac{L^{(r)}_{\xi^{(r)}}}{2}\mathbb{E}_{\zeta^{(r)}}\left[\left\|\boldsymbol{\eta}^{(r)}_{(\xi^{(r)})} \circledast \boldsymbol{G}^{(r)}_{(\xi^{(r)})}\right\|^2 \Big|\mathcal{B}^{(r)},\xi^{(r)}\right] \end{aligned} \tag{40}$$

Taking total expectation on both sides, we have

$$\begin{aligned}\mathbb{E}\left[f(\boldsymbol{\theta}^{(r+1)})\right] &\leq \mathbb{E}\left[f(\boldsymbol{\theta}^{(r)})\right] \\ &- \mathbb{E}\left[\left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}), \boldsymbol{\eta}^{(r)}_{(\xi^{(r)})} \circledast \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)})\right\rangle\right] \\ &+ \mathbb{E}\left[\frac{L^{(r)}_{\xi^{(r)}}}{2}\left\|\boldsymbol{\eta}^{(r)}_{(\xi^{(r)})} \circledast \boldsymbol{G}^{(r)}_{(\xi^{(r)})}\right\|^2\right]. \end{aligned} \tag{41}$$

From the above inequality and the assumption that $L^{(r)}_{(n)}$ is bounded from above by L, we can conclude that

$$\begin{aligned}&\sum_{r=1}^{R}\mathbb{E}\left[\left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}), \boldsymbol{\eta}^{(r)}_{(\xi^{(r)})} \circledast \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)})\right\rangle\right] \\ &\leq f(\boldsymbol{\theta}^{(0)}) - f(\boldsymbol{\theta}^{(\star)}) + \sum_{r=1}^{R}\frac{L}{2}\mathbb{E}\left[\left\|\boldsymbol{\eta}^{(r)}_{(\xi^{(r)})} \circledast \boldsymbol{G}^{(r)}_{(\xi^{(r)})}\right\|^2\right]\end{aligned}$$

which is by summing up all the inequalities in (40) from $r = 1$ to R.

Let us observe the last term on the right hand side and take $R \to \infty$:

$$\begin{aligned}&\mathbb{E}\left[\sum_{r=1}^{\infty}\left\|\boldsymbol{\eta}^{(r)}_{(\xi^{(r)})} \circledast \boldsymbol{G}^{(r)}_{(\xi^{(r)})}\right\|^2\right] \\ &= \mathbb{E}\left[\sum_{r=1}^{\infty}\sum_{i=1}^{J_{\xi^{(r)}}}\sum_{f=1}^{F}\left[\boldsymbol{\eta}^{(r)}_{(\xi^{(r)})}\right]^2_{i,f}\left[\boldsymbol{G}^{(r)}_{(\xi^{(r)})}\right]^2_{i,f}\right] \\ &= \mathbb{E}\left[\sum_{r=1}^{\infty}\sum_{i=1}^{J_{\xi^{(r)}}}\sum_{f=1}^{F}\left[\boldsymbol{\eta}^{(r+1)}_{(\xi^{(r)})}\right]^2_{i,f}\left[\boldsymbol{G}^{(r)}_{(\xi^{(r)})}\right]^2_{i,f}\right. \\ &\left.+\sum_{r=1}^{\infty}\sum_{i=1}^{J_{\xi^{(r)}}}\sum_{f=1}^{F}\left(\left[\boldsymbol{\eta}^{(r)}_{(\xi^{(r)})}\right]^2_{i,f} - \left[\boldsymbol{\eta}^{(r+1)}_{(\xi^{(r)})}\right]^2_{i,f}\right)\left[\boldsymbol{G}^{(r)}_{(\xi^{(r)})}\right]^2_{i,f}\right].\end{aligned} \tag{42}$$

Note that we have exchanged the order of the limits and expectations, since the expectation is taking on nonnegative terms. Using Lemma 2, one can easily show the first term above satisfies is

bounded from above by

$$\frac{C_1}{2\epsilon\beta^{2\epsilon}}$$

where $0 < C_1 < \infty$ is a constant. To see the second term is also bounded, observe that

$$\begin{aligned}
&\sum_{r=1}^{\infty}\left(\left[\boldsymbol{\eta}^{(r)}_{(\xi^{(r)})}\right]^2_{i,f} - \left[\boldsymbol{\eta}^{(r+1)}_{(\xi^{(r)})}\right]^2_{i,f}\right)\left[\boldsymbol{G}^{(r)}_{(\xi^{(r)})}\right]^2_{i,f} \\
&\leq \max_{r\geq 0}\left[\boldsymbol{G}^{(r)}_{(\xi^{(r)})}\right]^2_{i,f}\sum_{r=1}^{\infty}\left(\left[\boldsymbol{\eta}^{(r)}_{(\xi^{(r)})}\right]^2_{i,f} - \left[\boldsymbol{\eta}^{(r+1)}_{(\xi^{(r)})}\right]^2_{i,f}\right) \\
&\leq \max_{r\geq 0}\left[\boldsymbol{G}^{(r)}_{(\xi^{(r)})}\right]^2_{i,f}\left[\boldsymbol{\eta}^{(r)}_{(\xi^{(r)})}\right]^2_{i,f} \\
&\leq \left[\boldsymbol{\eta}^{(r)}_{(\xi^{(r)})}\right]^2_{i,f}\left(\left[\nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)})\right]^2_{i,f}\right. \\
&\quad \left. + \left(\left[\nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)})\right]_{i,f} - \left[\boldsymbol{G}^{(r)}_{(\xi^{(r)})}\right]_{i,f}\right)^2\right)
\end{aligned} \tag{43}$$

Since we have assumed that $\boldsymbol{A}^{(r)}_{(n)}$'s are bounded, the right hand side is bounded from above. Therefore, we have reached the conclusion

$$\mathbb{E}\left[\sum_{r=1}^{\infty}\left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}), \boldsymbol{\eta}^{(r)}_{(\xi^{(r)})} \circledast \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)})\right\rangle\right] < \infty.$$

Applying Lemma 3, one can see that

$$\mathsf{Pr}\left(\sum_{r=1}^{\infty}\left[\boldsymbol{\eta}^{(r)}_{(\xi^{(r)})}\right]_{i,f}\left[\nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)})\right]^2_{i,f} < \infty\right) = 1.$$

Since $\mathsf{Pr}(\xi^{(r)} = n) > 0$, one immediate result is that any n appears infinitely many times in the sequence $r = 1, \ldots, \infty$, according to the second Borel-Cantelli lemma. This leads to

$$\mathsf{Pr}\left(\sum_{j=1}^{\infty}\left[\boldsymbol{\eta}^{(r_j(n))}_{(n)}\right]_{i,f}\left[\nabla_{\boldsymbol{A}_{(n)}} f(\boldsymbol{\theta}^{(r_j(n))})\right]^2_{i,f} < \infty\right) = 1,$$

holds for $n = 1, \ldots, N$, where $r_1(n), \ldots, r_j(n), \ldots$ is the subsequence of $\{r\}$ such that block n is sampled for updating.

Hence, with probability one there exists a subsequence $r_1(n), \ldots, r_\infty(n)$ such that at the corresponding iterations block n is sampled for updating. It is not hard to show that

$$\sum_{j=1}^{\infty}\left[\boldsymbol{\eta}^{(r_j(n))}_{(n)}\right]_{i,f} = \infty,$$

by the assumption that $\boldsymbol{A}^{(r)}_{(n)}$ are all bounded. This directly implies that

$$\sum_{r=1}^{\infty}\left[\boldsymbol{\eta}^{(r)}_{(n)}\right]_{i,f} = \infty, \quad \forall n.$$

Hence, by Lemma 2, we have

$$\lim_{r\to\infty} \; [\nabla_{\boldsymbol{A}_{(n)}} f(\boldsymbol{\theta}^{(r)})]_{i,f}^2 = 0$$

with probability one.

References

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

[2] 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.

[3] 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.

[4] B. Rao and K. Kreutz-Delgado, "An affine scaling methodology for best basis selection," *IEEE Trans. Signal Process.*, vol. 47, no. 1, pp. 187 –200, jan 1999.

[5] J. Sun, D. Tao, and C. Faloutsos, "Beyond streams and graphs: dynamic tensor analysis," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 374–383.

[6] 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.

[7] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky, "Tensor decompositions for learning latent variable models," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 2773–2832, 2014.

[8] A. Anandkumar, D. Hsu, and S. M. Kakade, "A method of moments for mixture models and hidden markov models," in *Conference on Learning Theory*, 2012, pp. 33–1.

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

[10] N. D. Sidiropoulos and X.-Q. Liu, "Identifiability results for blind beamforming in incoherent multipath with small delay spread," *IEEE Trans. Signal Process.*, vol. 49, no. 1, pp. 228–236, Jan. 2001.

[11] X. Fu, N. D. Sidiropoulos, J. H. Tranter, and W.-K. Ma, "A factor analysis framework for power spectra separation and emitter localization," *IEEE Trans. Signal Process.*, vol. 63, no. 24, pp. 6581–6594, 2015.

[12] C. J. Hillar and L.-H. Lim, "Most tensor problems are np-hard," *Journal of the ACM (JACM)*, vol. 60, no. 6, p. 45, 2013.

[13] 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.

[14] 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.

[15] 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.

[16] C. Navasca, L. De Lathauwer, and S. Kindermann, "Swamp reducing technique for tensor decomposition." in *EUSIPCO*, 2008, pp. 1–5.

[17] P. Comon, X. Luciani, and A. L. De Almeida, "Tensor decompositions, alternating least squares and other tales," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 23, no. 7-8, pp. 393–405, 2009.

[18] 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.

[19] 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.

[20] A. L. Alexander, J. E. Lee, M. Lazar, and A. S. Field, "Diffusion tensor imaging of the brain," *Neurotherapeutics*, vol. 4, no. 3, pp. 316–329, 2007.

[21] A. Shashua and T. Hazan, "Non-negative tensor factorization with applications to statistics and computer vision," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 792–799.

[22] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.

[23] 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.

[24] 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.

[25] 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.

[26] A. Beck and L. Tetruashvili, "On the convergence of block coordinate descent type methods," *SIAM journal on Optimization*, vol. 23, no. 4, pp. 2037–2060, 2013.

[27] Y. Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 341–362, 2012.

[28] S. Ghadimi and G. Lan, "Accelerated gradient methods for nonconvex nonlinear and stochastic programming," *Mathematical Programming*, vol. 156, no. 1-2, pp. 59–99, 2016.

[29] ——, "Stochastic first-and zeroth-order methods for nonconvex stochastic programming," *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2341–2368, 2013.

[30] M. Razaviyayn, M. Hong, and Z.-Q. Luo, "A unified convergence analysis of block successive minimization methods for nonsmooth optimization," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 1126–1153, 2013.

[31] H. Wang and A. Banerjee, "Randomized block coordinate descent for online and stochastic optimization," *arXiv preprint arXiv:1407.0107*, 2014.

[32] Y. Xu and W. Yin, "Block stochastic gradient iteration for convex and nonconvex optimization," *SIAM Journal on Optimization*, vol. 25, no. 3, pp. 1686–1716, 2015.

[33] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[34] X. Fu, C. Gao, H.-T. Wai, and K. Huang, "Block-randomized stochastic proximal gradient for constrained low-rank tensor factorization," in submitted to *IEEE ICASSP 2019*, 2019.

[35] E. C. Chi and T. G. Kolda, "On tensors, sparsity, and nonnegative factorizations," *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 4, pp. 1272–1299, 2012.

[36] S. A. Vorobyov, Y. Rong, N. D. Sidiropoulos, and A. B. Gershman, "Robust iterative fitting of multilinear models," *IEEE Trans. Signal Process.*, vol. 53, no. 8, pp. 2678–2689, Aug 2005.

[37] X. Fu, K. Huang, W.-K. Ma, N. Sidiropoulos, and R. Bro, "Joint tensor factorization and outlying slab suppression with applications," *IEEE Trans. Signal Process.*, vol. 63, no. 23, pp. 6315–6328, 2015.

[38] 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.

[39] N. Kargas, N. D. Sidiropoulos, and X. Fu, "Tensors, learning, and'kolmogorov extension'for finite-alphabet random vectors," *arXiv preprint arXiv:1712.00205*, 2017.

[40] L. Xiao and T. Zhang, "A proximal stochastic gradient method with progressive variance reduction," *SIAM Journal on Optimization*, vol. 24, no. 4, pp. 2057–2075, 2014.

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

[42] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, "Efficient projections onto the ℓ_1-ball for learning in high dimensions," in *Proc. the 25th international conference on Machine learning*. ACM, 2008, pp. 272–279.

[43] J. B. Kruskal, "Nonmetric multidimensional scaling: a numerical method," *Psychometrika*, vol. 29, no. 2, pp. 115–129, 1964.

[44] R. Bro and N. D. Sidiropoulos, "Least squares algorithms under unimodality and non-negativity constraints," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 12, no. 4, pp. 223–247, 1998.

[45] H. Robbins and S. Monro, "A stochastic approximation method," in *Herbert Robbins Selected Papers*. Springer, 1985, pp. 102–109.

[46] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[47] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[48] T. Dozat, "Incorporating nesterov momentum into adam," 2016.

[49] X. Li and F. Orabona, "On the convergence of stochastic gradient descent with adaptive stepsizes," *arXiv preprint arXiv:1805.08114*, 2018.

[50] X. Chen, S. Liu, R. Sun, and M. Hong, "On the convergence of a class of adam-type algorithms for non-convex optimization," *arXiv preprint arXiv:1808.02941*, 2018.

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

[52] 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.

[53] 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.

[54] D. P. Bertsekas, *Nonlinear programming*. Athena Scientific, 1999.