Stochastic Mirror Descent for Low-Rank Tensor Decomposition Under Non-Euclidean Losses

Wenqiang Pu, *Member, IEEE*, Shahana Ibrahim, Xiao Fu, *Senior Member, IEEE*, and Mingyi Hong, *Senior Member, IEEE*

***Abstract*—This work considers low-rank canonical polyadic decomposition (CPD) under a class of non-Euclidean loss functions that frequently arise in statistical machine learning and signal processing. These loss functions are often used for certain types of tensor data, e.g., count and binary tensors, where the least squares loss is considered unnatural. Compared to the least squares loss, the non-Euclidean losses are generally more challenging to handle. Non-Euclidean CPD has attracted considerable interests and a number of prior works exist. However, pressing computational and theoretical challenges, such as scalability and convergence issues, still remain. This work offers a unified stochastic algorithmic framework for large-scale CPD decomposition under a variety of non-Euclidean loss functions. Our key contribution lies in a tensor fiber sampling strategy-based flexible stochastic mirror descent framework. Leveraging the sampling scheme and the multilinear algebraic structure of low-rank tensors, the proposed lightweight algorithm ensures global convergence to a stationary point under reasonable conditions. Numerical results show that our framework attains promising non-Euclidean CPD performance. The proposed framework also exhibits substantial computational savings compared to state-of-the-art methods.**

***Index Terms*—Tensor decomposition, stochastic optimization, mirror descent method, □-divergence, KL-divergence.**

I. Introduction

CANONICAL polyadic decomposition (CPD) has been used in many core tasks in signal processing and machine learning, such as neural signal analysis, video processing, array signal processing, text mining, social network analysis, link prediction, among others—see [1]–[4].

Manuscript received May 11, 2021; revised December 8, 2021 and March 10, 2022; accepted March 21, 2022. Date of publication March 31, 2022; date of current version April 26, 2022. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Evrim Acar. The work of Wenqiang Pu was supported by the NSFC, China under Grant 62101350. The work of Xiao Fu was supported in part by NSF ECCS under Grants 1808159 and IIS-1910118 and in part by ARO under Grant W911NF-19-1-0247. The work of Mingyi Hong was supported in part by NSF under Grant CIF-1910385 and in part by ARO under Grant W911NF-19-1-0247. *(Corresponding author: Wenqiang Pu.)*

Wenqiang Pu is with Shenzhen Research Institute of Big Data, Shenzhen 518172, China (e-mail: wenqiangpu@outlook.com).

Shahana Ibrahim and Xiao Fu are with the Electrical Engineering and Computer Science Department, Oregon State University, Corvallis, OR 97331 USA (e-mail: ibrahish@oregonstate.edu; xiao.fu@oregonstate.edu).

Mingyi Hong is with the Electrical and Computer Engineering Department, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: mhong@umn.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TSP.2022.3163896, provided by the authors.

Digital Object Identifier 10.1109/TSP.2022.3163896

The majority of classic CPD models and algorithms were developed for least squares (LS) problems using the Euclidean distance-based fitting criterion; see [1]–[3], [5] and references therein. However, the Euclidean distance is unnatural for measuring the "distance" between many types of real-world data, e.g., stochastic, integer, and binary data. In principle, using certain "data geometry-aware" divergences to serve as the fitting criteria may greatly improve performance and robustness in practice [6]–[9]. For example, the "distance" between two probability distributions is typically measured by a proper divergence, such as the generalized *Kullback-Leibler* (KL) divergence [10]–[13] and Itakura-Saito (IS) divergence [14]. These divergences take into consideration that the data is constrained in the probabilistic simplex, and thus are often more effective relative to the LS criterion in analyzing data that are not generated over the entire Euclidean space. From a statistical estimation viewpoint, many non-Euclidean divergences are closely related to the *maximum likelihood estimators* (MLEs) under plausible data distributions. For example, the generalized KL divergence [15] and logistic loss [6]–[8] can be derived from the MLEs of count integer data and binary data that follow certain Poisson distributions and Bernoulli distributions, respectively.

However, computing CPD under non-Euclidean divergences is much more challenging compared with the case under Euclidean loss (or, the LS loss), especially when the data size becomes huge. Algorithms developed under the LS loss are often not easily extendable to handle these more complicated loss functions, due to the lack of "nice" properties that are possessed by the LS loss, e.g., the gradient Lipschitz continuity under relatively mild conditions. Below, we provide a brief review on existing developments for CPD models with specific loss function.

A. Prior Works

Many existing non-Euclidean CPD approaches employ the *block coordinate descent* (BCD) paradigm [27] with divergence-specific strategies for block variable updating. For example, the work in [26] proposed a hierarchical alternating optimization algorithm for CPD with □ and □divergence. In [15], the generalized KL-divergence loss was considered, where a block majorization-minimization (MM) algorithm was developed. In [11], the exponential gradient algorithm was proposed for the KL-divergence. Similar strategies were developed for

TABLE I
BRIEF REVIEW OF ALGORITHMS FOR CPD MODEL

Algorithm	Stochastic	Loss function	Data Type
First-order type algorithm [16]–[18]	No	LS	continuous
Primal-dual algorithm [19]	No	LS	continuous
(Quasi-)Second-order algorithm [20], [21]	No	LS	continuous
Stochastic optimization algorithm [22]–[25]	Yes	LS	continuous
Hierarchical alternating optimization [26]	No	$\alpha, \beta-$ Div.	continuous
Majorization-minimization algorithm [15]	No	KL Div.	continuous and count
Multiplicative update algorithm [14]	No	LS, KL Div., IS Div.	binary
Exponential gradient algorithm [11]	No	KL Div.	continuous and count
Alternating optimization algorithm [8]	No	logistic loss and others	binary
Generalized Gaussian Newton algorithm [9]	No	β-div.	continuous and count
Stochastic gradient descent algorithm [7]	Yes	general loss*	continuous, count, and binary
Stochastic mirror descent algorithm (this work)	Yes	general loss*	continuous, count, and binary

*The general loss in this table refers to many ML criterion motivated losses [6], [7] as well as statistical divergences such as KL div., IS div. and etc.

the KL and IS divergences [14]. Recently in [8], several ML-based loss functions for binary data were considered and an alternating optimization algorithm with line search was proposed. Besides BCD, other optimization frameworks such as Gauss-Newton based methods [9], quasi-Newton methods [6], and stochastic gradient-based methods [7] were also developed for non-Euclidean CPD.

It is important to note that most of the algorithms mentioned above (such as [8], [9], [14], [15], [26], [28]), are *batch* algorithms, which utilize the entire data set to perform every update. They become increasingly slow when the size of the data increases. On the other hand, *stochastic* algorithms are effective in reducing per-iteration computational and memory burdens. Recently, a stochastic gradient descent (SGD) based algorithm [7] was proposed for CPD with a suite of statistical criterion based loss functions (including loss functions under missing values that are not considered in this work). The algorithm was developed based on randomly sampling the tensor entries. Hence, it is difficult to exploit some interesting multilinear algebraic properties of low-rank tensors to further improve computational efficiency. In addition, the SGD algorithm in [7] lacks convergence guarantees. Finally, in Table I, we summarize the properties of a number of recently developed algorithms for the CPD model.

B. Contributions

In this paper, we develop a *unified stochastic mirror descent* (SMD) algorithmic framework for large-scale CPD under various non-Euclidean losses. Our major contributions are summarized as follows:

- *Efficient fiber-sampled stochastic MD framework:* We propose a block-randomized SMD algorithmic framework that is tailored for tensor decomposition. Both MD and SMD are known for its effectiveness in handling non-Euclidean losses [29], but directly applying generic SMD does not fully exploit the underlying CPD structure. We use a recently emerged tensor data sampling strategy (namely, fiber sampling [25], [30]) to assist designing SMD-type updates. The fiber sampling strategy judiciously uses the multilinear structure of low-rank tensors, which gives rise to structured (non-)convex subproblems. These structures can often be exploited to come up with economical update rules for CPD.
- *A suite of solutions for various losses and constraints:* We carefully craft solutions for a series of non-Euclidean losses. The proposed algorithmic framework allows flexible choices of the local surrogate functions under the SMD framework to adapt to different loss functions. Such flexibility also helps offer lightweight updates when the latent factors are under a variety of constraints that are of interest in data analytics. In particular, we pay special attention to binary and integer data CPD problems, which find numerous applications across disciplines.
- *Guaranteed convergence:* We offer convergence characterizations for our block-randomized SMD-based non-Euclidean CPD framework. Establishing stationary-point convergence for generic SMD is already a challenging problem. The work in [31] on SMD requires its gradient estimation error converging to zero, which is unrealistic in many cases, especially under the context of CPD. In this work, we leverage the notion of *relative smoothness* and the tensor fiber sampling strategy to construct lightweight SMD updates for different losses. This design also helps circumvent stringent conditions (e.g., vanishing gradient estimation error) when establishing convergence. To our best knowledge, such convergence results for multi-block SMD under nonconvex settings have been elusive in the literature.

Part of the work appears in IEEE ICASSP 2021 [32]. The conference version considered algorithm design under the □ divergence loss. This journal version extends the ideas to handle more non-Euclidean losses, e.g., the logistic loss that is critical in binary data analysis. More importantly, this version provides unified convergence analysis for the proposed algorithmic structure. Some important practical considerations, e.g., stepsize scheduling, is also discussed and experimented. The journal version also contains substantially more simulations and real-data validation.

C. Notation

We follow the conventional notation in signal processing, x, $\boldsymbol{x}$, $\boldsymbol{X}$, and $\underline{\boldsymbol{X}}$ denote scalar, vector, matrix, and tensor, respectively. Given a matrix $\boldsymbol{X}$, $\boldsymbol{X}^{\cdot c}$ and $\exp(\boldsymbol{X})$ denote the entry-wise power and exponential operations respectively; $\boldsymbol{X} \pm c$ denote

adding/subtracting a constant c for each entry of $\boldsymbol{X}$; vec($\boldsymbol{X}$) denote the vectorization operator that concatenates the columns of $\boldsymbol{X}$. We use $\circledast$, $\times$ and $\oslash$ to denote the Hadamard product, the Khatri-Rao product, and entry-wise division respectively. ^{T} denotes the transpose operation. Script letter $\mathcal{C}$ is used to denote a discrete set and $|\mathcal{C}|$ is the cardinality. $\mathcal{C}$ $\|\cdot\|_F$ denotes the Euclidean norm of vector, $\|\cdot\|_F$ denotes the Frobenius norm of matrix, and $\langle\cdot,\cdot\rangle$ denotes the inner product. For a positive integer N, the set $\{1, 2, \ldots, N\}$ is denoted as $[N]$ for short. For a function $\Box$ its domain and interior domain is denoted as dom $\Box$ and int dom $\Box$ respectively. Other notation will be explained when it first appears.

II. CPD Under Non-Euclidean Losses

Consider a data tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, where N is the order of the tensor $\boldsymbol{\mathcal{X}}$ and $I_n > 0$ ($n \in [N]$) is the size of the nth mode of $\boldsymbol{\mathcal{X}}$. Such multi-way data tensors arise in many applications. The entries of the data tensor $\boldsymbol{\mathcal{X}}$ could be continuous real numbers, non-negative integers or binaries. A general problem of interest is to approximate $\boldsymbol{\mathcal{X}}$ using a low rank tensor $\boldsymbol{\mathcal{M}}$, defined as

$$\boldsymbol{\mathcal{M}} = \sum_{r=1}^{R} \boldsymbol{A}_1(:, r) \circ \boldsymbol{A}_2(:, r) \circ \ldots \circ \boldsymbol{A}_N(:, r), \quad (1)$$

where "$\circ$" denotes the outer product of vectors, $\boldsymbol{A}_n \in \mathbb{R}^{I_n \times R}$ is the mode-n latent factor matrix; R is the smallest positive integer such that (1) holds, and it is also known as the rank of $\boldsymbol{\mathcal{M}}$.

Denote an N-dimensional integer vector $\boldsymbol{i}$ as the entry coordinate, i.e.,

$$\boldsymbol{i} \in \mathcal{I} \triangleq \{(i_1, i_2, \ldots, i_N) \mid i_n = 1, 2, \ldots, I_n, \forall n\}.$$

Then the CPD problem can be formulated as the following minimization problem with a loss function of interest $f(\cdot, \cdot)$: $\mathbb{R} \times \mathbb{R} \to \mathbb{R}$,

$$\begin{aligned} \min_{\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_N} \quad & \frac{1}{|\mathcal{I}|} \sum_{\boldsymbol{i} \in \mathcal{I}} f(\boldsymbol{\mathcal{X}}_{\boldsymbol{i}}, \boldsymbol{\mathcal{M}}_{\boldsymbol{i}}) \\ \text{s.t.} \quad & \boldsymbol{\mathcal{M}}_{\boldsymbol{i}} = \sum_{r=1}^{R} \prod_{n=1}^{N} \boldsymbol{A}_n(i_n, r), \ \forall \boldsymbol{i} \in \mathcal{I}, \\ & \boldsymbol{A}_n \in \mathcal{A}_n, \ \forall n, \end{aligned} \quad (2)$$

where $\boldsymbol{\mathcal{X}}_{\boldsymbol{i}}$ and $\boldsymbol{\mathcal{M}}_{\boldsymbol{i}}$ denote the entries of $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{M}}$ indexed by $\boldsymbol{i}$, respectively, $\mathcal{A}_n$ is a convex and closed set which captures the prior information about the structure of latent factors $\boldsymbol{A}_n$, e.g., non-negativity, sparsity, and smoothness. By choosing proper loss functions f, Problem (2) is used for handling different types of data, e.g., continuous, count, and binary data. Several representative motivating examples are as follows,

- *KL-divergence for count data:* In many real-world scenarios, data is naturally recorded as nonnegative integers, e.g., crime numbers across locations and time[1] and email interactions recorded over months [33]. As an information-theoretic measure, the KL divergence was originally proposed for quantifying similarity between two probability distributions. The *generalized* KL-divergence that handles nonngeative quantities beyond distributions is also widely used in data analytics [10], [12], [15], [34]. The generalized KL divergence has an f defined as follows:

$$\text{(KL-Div.)} \quad f(x, m) = m - x \log(m), \quad (3)$$

where $x \in \mathbb{N}$ and $m > 0$. Problem (2) with KL-divergence can also be interpreted as the MLE for estimating the Poisson parameter tensor $\boldsymbol{\mathcal{M}}$ from independent samples $\boldsymbol{\mathcal{X}}$, where $\boldsymbol{\mathcal{M}}$ has a low-rank structure [6], [7], [15]. To be more precise, the corresponding statistical model is

$$\boldsymbol{\mathcal{X}}_{\boldsymbol{i}} \sim \text{Poisson}(\boldsymbol{\mathcal{M}}_{\boldsymbol{i}}), \ \forall \boldsymbol{i},$$

where Poisson(m) denotes the Poisson distribution with a mean of m.

- *Log loss for binary data:* Binary data is also frequently encountered in data analytics, e.g., in adjacency matrix-based social network community detection [4], [35] and knowledge base analysis [6]–[8]. Binary data fitting is often handled using the following loss:

$$\text{(Log Loss)} \quad f(x, m) = \log(1 + e^m) - xm, \quad (4)$$

where $x \in \{0, 1\}$ and $m > 0$. The log loss can be interpreted as MLE for finding the Bernoulli distribution parameter [8]. The associated binary data generation model of the independent samples is

$$\boldsymbol{\mathcal{X}}_{\boldsymbol{i}} \sim \text{Bernoulli}(\boldsymbol{\mathcal{P}}_{\boldsymbol{i}}), \ \boldsymbol{\mathcal{P}}_{\boldsymbol{i}} = e^{\boldsymbol{\mathcal{M}}_{\boldsymbol{i}}}/(1 + e^{\boldsymbol{\mathcal{M}}_{\boldsymbol{i}}}), \ \forall \boldsymbol{i},$$

where Bernoulli(p) denotes the Bernoulli distribution, p is the probability for x taking 1, and $\boldsymbol{\mathcal{P}}$ has the same size of $\boldsymbol{\mathcal{X}}$.

- *β-divergence:* Non-Euclidean losses also find applications in continuous data CPD, especially under non-Gaussian and/or non-additive noise, e.g., multiplicative Gamma noise [9]. For example, the β-divergence was found useful for neural signal analysis [26] and recently is studied as CPD fitting criterion [6], [7], [9], [32]. The β-divergence is parametrized by a constant $\beta \in \mathbb{R}$ defined as

$$\text{(β-Div.)} \quad f(x, m) = \begin{cases} \frac{x}{m} - \log\left(\frac{x}{m}\right) - 1, & \beta = 0, \\ x \log\frac{x}{m} + m - x, & \beta = 1, \\ \frac{x^\beta + (\beta - 1)m^\beta - \beta x m^{\beta - 1}}{\beta(\beta - 1)}, & \text{o.w.} \end{cases}$$

The β-divergence subsumes the IS divergence ($\beta = 0$), the generalized KL divergence ($\beta = 1$), and the Euclidean distance ($\beta = 2$) as special cases. When $\beta = 0$, it can also be interpreted as MLE corresponds to data with multiplicative Gamma noise. In music data analysis, $\beta < 2$ was found useful, since such loss functions capture low intensity spectra components—but the Euclidean loss tends to focus on significant variations in data [34].

Remark 1: As one has seen in the examples, one way to select f is to take a statistical analysis viewpoint. Each entry of the data tensor is treated as a conditional independent random variable, whose generation is statistically modeled as follows:

$$\boldsymbol{\mathcal{X}}_{\boldsymbol{i}} \sim p(\boldsymbol{\mathcal{X}}_{\boldsymbol{i}} \mid \Box(\boldsymbol{\mathcal{M}}_{\boldsymbol{i}})), \ \forall \boldsymbol{i} \in \mathcal{I} \quad (5)$$

[1]See official website of the city of Chicago, www.cityofchicago.org.

TABLE II
DISTRIBUTIONS, LINK FUNCTIONS, AND LOSS FUNCTIONS FOR DIFFERENT TYPES OF DATA

Data Type	Distribution	Link Function	Loss function	Parameter Type	Name
Continuous	Gaussian	$\theta(m) = m$	$\frac{1}{2}(x-m)^2$	$x, m \in \mathbb{R}$	Euclidean Dis.
	Gamma	$\theta(m) = m$	$\frac{x}{m+\epsilon} + \log(m+\epsilon)$	$x > 0, m \geq 0$	IS Div.
	–	–	$(m+\epsilon)^\beta/\beta - x(m+\epsilon)^{\beta-1}/(\beta-1)$	$x \geq 0, m \geq 0$	β-Div. ($\beta \in \mathbb{R} \setminus \{0,1,2\}$)
Count	Poisson	$\theta(m) = m$	$m - x\log(m+\epsilon)$	$x \in \mathbb{N}, m \geq 0$	generalized KL Div.,
		$\theta(m) = e^m$	$e^m - xm$	$x \in \mathbb{N}, m \in \mathbb{R}$	–
Binary	Bernoulli	$\theta(m) = \frac{m}{1+m}$	$\log(m+1) - x\log(m+\epsilon)$	$x \in \{0,1\}, m \geq 0$	–
		$\theta(m) = \frac{e^m}{1+e^m}$	$\log(1+e^m) - xm$	$x \in \{0,1\}, m \in \mathbb{R}$	–

where $p(x;\square)$ is a distribution with natural parameter $\square$ (e.g., the Poisson and Bernoulli distribution) and $\square(\cdot)$: $\mathbb{R} \to \mathbb{R}$ is an invertible function whose inverse is often referred as the *link function* in statistics (e.g., $\square(m) = m$ and $\square(m) = \frac{m}{1+m}$). A straightforward intuition behind model (5) is that, the observed data tensor $\boldsymbol{\mathcal{X}}$ is 'embedded' on a *latent* low rank tensor $\boldsymbol{\mathcal{M}}$, whose generation procedure is characterized by $p(x;\square)$ and $\square(\cdot)$. To find $\boldsymbol{\mathcal{M}}$, a statistically efficient way is choosing $f(\cdot)$ as the negative log-likelihood function associated with model (5), given as

$$f(\boldsymbol{\mathcal{X}}_i, \boldsymbol{\mathcal{M}}_i) \overset{\square}{=} -\log p(\boldsymbol{\mathcal{X}}_i \mid \square(\boldsymbol{\mathcal{M}}_i)) + \text{constant},$$

which naturally leads to a non-Euclidean CPD problem (if the distribution is not Gaussian).

In Table II, some frequently used $f(\cdot)$, link function $\square(\cdot)$, and distribution of our interests are illustrated. In the table, $c > 0$ is a sufficiently small number to avoid the function value or gradient being $\square$, i.e., $c = 10^{-10}$. This modification is often used in the literature [6].

Tackling Problem (2) at large scale is highly nontrivial. For example, a $5000 \times 5000 \times 5000$ tensor can be as large as 900 GB if the double precision arithmetic is used, which means that batch algorithms may not be a viable option. Instead, stochastic algorithms that sample 'partial data' per iteration become an attractive choice. In Euclidean loss CPD, it has been observed that stochastic algorithms can significantly reduce computational and memory cost per iteration; see [23], [25], [30]. Nonetheless, unlike Euclidean CPD, where various data sampling schemes and update rules may all offer competitive algorithms [5], non-Euclidean losses' complex structures may make stochastic algorithm design a more delicate process. In other words, the sampling schemes may affect the subsequent update rules' complexity and convergence properties of the algorithm.

Next, we offer a unified stochastic algorithmic structure that can efficiently tackle CPD under a variety of non-Euclidean losses. Our development is an integrated design of data sampling and *Lipschitz-like convexity* [36] based local surrogate construction, leveraging the underlying multilinear structure of low-rank tensors.

III. PROPOSED APPROACH

A. Preliminaries

A number of algorithmic frameworks have been considered for handling Problem (2) with non-Euclidean losses, e.g., stochastic gradient descent (SGD) [7], block coordinate descent (or, alternating optimization (AO)) [8], [15], and the Gauss-Newton (GN) method [9].

Let us denote $\boldsymbol{A} := (\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_N)$, $A := A^1 \times A^2 \times \cdots \times A^N$. We also use $F(\boldsymbol{A})$ to represent the objective function of Problem (2). The updates of AO and SGD type algorithms can be summarized as follows:

$$\text{(AO)}\ \boldsymbol{A}_n^{t+1} \overset{\square}{=} \arg\min_{\boldsymbol{A}_n \in A^n} F(\boldsymbol{A}_n; \boldsymbol{A}_{\square^n}^t), \tag{6a}$$

$$\text{(SGD)}\ \boldsymbol{A}^{t+1} = \text{Proj}_A(\boldsymbol{A}^t - \square_t \hat{\boldsymbol{G}}^t)$$
$$= \arg\min_{\boldsymbol{A} \in A} \langle \boldsymbol{A}, \hat{\boldsymbol{G}}^t \rangle + \frac{1}{2\square_t} \|\boldsymbol{A} - \boldsymbol{A}^t\|_F^2. \tag{6b}$$

In (6), $\boldsymbol{A}_{\square^n}$ corresponds to $\boldsymbol{A}$ with $\boldsymbol{A}_n$ being removed; $F(\boldsymbol{A}^n; \boldsymbol{A}_{\square^n}^t)$ is the objective function with fixed $\boldsymbol{A}_{\square^n}^t$; $\hat{\boldsymbol{G}}^t$ is *gradient estimation* from sampled data; $\square_t > 0$ represents the step size; and $\text{Proj}_A(\cdot)$ denotes the projection onto constraint set A. Many deterministic non-Euclidean tensor decomposition algorithms take the AO route; see, e.g., [8], [15], whereas the recent work in [9] used a GN method to improve the iteration complexity (i.e., the number of iterations needed for reaching a satisfactory solution). However, the AO and GN methods face heavy per-iteration computational and memory complexities when handling large-scale tensors; see the "MTTKRP" challenge discussed in [1], [2], [5], [25]. The SGD approach in [7] is more lightweight in terms of the per-iteration resource consumption. Constructing $\hat{\boldsymbol{G}}$ can be fairly economical (c.f. [6, Theorem 3]) since only partial data is used. This makes the per-iteration complexity of the algorithm affordable, even if the tensor of interest is large.

However, simply using SGD for the non-Euclidean CPD problem may not be the most effective approach. One can see that from (6b), every iteration of SGD is equivalent to solving a quadratic program, which is used as a local surrogate of the original cost function. However, it is known that such quadratic functions may not be a good approximation for many non-Euclidean losses. In particular, using quadratic local surrogates may result in slow progresses since it largely ignores the geometry of the cost function [29], [36].

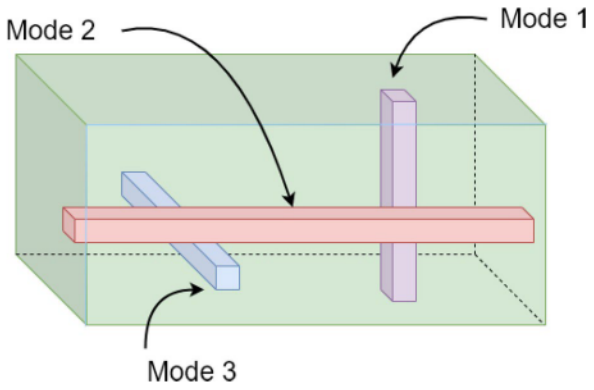


Fig. 1. Mode-n fibers of a third-order tensor, where $n = 1, 2, 3$.

In this section, we will propose a stochastic mirror descent (SMD) framework to handle the non-Euclidean CPD problem. Compared with SGD (6b), SMD replaces the quadratic term $\|\boldsymbol{A}_1 - \boldsymbol{A}_1^t\|_F^2$ by a general divergence metric named Bregman divergence $D_\psi(\boldsymbol{A}, \boldsymbol{A}^t)$ (see Definition 1),

$$(\text{SMD})\ \boldsymbol{A}^{t+1} = \arg\min_{\boldsymbol{A}} \langle \boldsymbol{A}, \hat{\boldsymbol{G}}^t \rangle + \frac{1}{\eta_t} D_\psi(\boldsymbol{A}, \boldsymbol{A}^t). \tag{7}$$

If ψ is properly designed, SMD can often take advantages of the problem geometry and attain substantial efficiency improvement compared to SGD, especially when non-Euclidean cost functions are used; see more on MD and SMD in the optimization literature [37]–[41]. The reason why using MD/SMD improves convergence over GD/SGD will become clearer later in the next subsections when specific examples of non-Euclidean losses and D_ψ's (e.g. the example in Fig. 2) are discussed.

Definition 1 (Bregman Divergence): Given a strongly convex function $\psi(\cdot)$: dom $\psi \subseteq \mathbb{R}^n \to \mathbb{R}$, the Bregman divergence between $\boldsymbol{a} \in$ dom ψ and $\boldsymbol{a}^t \in$ int dom ψ is

$$D_\psi(\boldsymbol{a}, \boldsymbol{a}^t) = \psi(\boldsymbol{a}) - \psi(\boldsymbol{a}^t) - \langle \nabla\psi(\boldsymbol{a}^t), \boldsymbol{a} - \boldsymbol{a}^t \rangle.$$

Remark 2: The Bregman divergence was originally defined in [42] using a Legendre function ψ. Here, we consider the case where ψ is a strongly convex function; see more details in [36]. It should be noted that $D_\psi(\boldsymbol{a}, \boldsymbol{a}^t) \geq 0$, $\boldsymbol{a}, \boldsymbol{a}^t$ and $D_\psi(\boldsymbol{a}, \boldsymbol{a}^t) = 0$ if and only if $\boldsymbol{a} = \boldsymbol{a}^t$ due to the strong convexity of ψ. In addition, if ψ is a one-dimensional function, $D_\psi(\boldsymbol{a}, \boldsymbol{a}^t)$ can be defined in a coordinate-wise form as $D_\psi(\boldsymbol{a}, \boldsymbol{a}^t) = \sum_{j=1}^n \psi(a_j) - \psi(a_j^t) - \langle \nabla\psi(a_j^t), a_j - a_j^t \rangle$.

The general-purpose SMD scheme in (7) is not new in the optimization literature and its convergence behavior under the stochastic setting has been studied in recent works, e.g., [31], [38], [40]. However, the general-purpose SMD scheme does not exploit the structure of the CPD problem in (2). Next, together with a tensor fiber sampling strategy advocated in [5], [25], [30], we will develop a block randomized SMD algorithm to exploit the multilinear structure of CPD, where only one block $\boldsymbol{A}_n$ is randomly selected and updated per SMD iteration.

B. Data Sampling

A key ingredient for stochastic algorithms lies in the data sampling strategy, which uses partial data to estimate the gradient in each iteration. Under the Euclidean loss, sub-tensor sampling [23], random entry sampling [22], and tensor fiber sampling [25], [30] were all considered—which all offered effective solutions. In principle, all the sampling strategies considered in the Euclidean case could still be used in the non-Euclidean cases. For example, the recent non-Euclidean CPD work in [7] used an entry sampling scheme. Nonetheless, since non-Euclidean losses are inherently more complex, different sampling strategies may lead to algorithms that admit drastically different updating rules and convergence properties.

In this work, we advocate the fiber sampling strategy that was used in [25], [30] for Euclidean loss CPD; see illustration in Fig. 1 for tensor fibers. We find this sampling strategy particularly handy in the non-Euclidean case, for a couple of reasons:

- *Incorporating Prior Information on $\boldsymbol{A}_n$:* Randomly sampling some indexes $\boldsymbol{i}$ [22] or selecting a subtensor [23] faces an issue that the sampled data may relate to only some rows of $\boldsymbol{A}_n$, while many important constraints under statistical non-Euclidean CPD are imposed on the columns of $\boldsymbol{A}_n$, e.g., the probability simplex constraints in [11], [43]–[45]. This may lead to difficulties in developing efficient updating rules while enforcing some types of prior information on $\boldsymbol{A}_n$. For example, the probability simplex constraints on $\boldsymbol{A}_n(:, r)$, norm constraints on $\boldsymbol{A}_n(:, r)$, and smoothness constraints on $\boldsymbol{A}_n(:, r)$ all have important applications [5], but the information of $\boldsymbol{A}_n(:, r)$ may not be included in the selected samples. With fiber sampling, every batch of sampled data contains information about one full $\boldsymbol{A}_n$, making dealing with such constraints easy.
- *Convex Approximation to Optimize $\boldsymbol{A}_n$:* The fiber sampling strategy also provides a way to further exploit the block-wise structure of the loss functions. Even under complex non-Euclidean losses, with the notion of *Lipschitz-like convexity* [36], the loss function $f(\cdot)$ with respect to each block $\boldsymbol{A}_n$ can often be locally approximated (or, to be precise, majorized) by a strongly convex function—which will prove useful for deriving lightweight updates.

The fiber sampling scheme can be understood using the *matrix unfolding* representations of low-rank tensors. The mode-n matrix unfolding of $\underline{\boldsymbol{X}}$ is a $J_n \times I_n$ matrix with $J_n = \prod_{m=1, m\neq n}^{N} I_m$, denoted as $\boldsymbol{X}_n$, and the entry-wise correspondence is $\boldsymbol{X}_{[n]}(j, i_n) = \underline{\boldsymbol{X}}(i_1, \ldots, i_N)$, $j = 1 + \sum_{k=1, k\neq n}^{N} (i_k - 1) S_k$, where $S_k = \prod_{m=1, m\neq n}^{k-1} I_m$. For the low-rank tensor $\underline{\boldsymbol{M}}$ in (1), its mode-n matrix unfolding can be expressed as $\boldsymbol{M}_n = \boldsymbol{H}_n \boldsymbol{A}_n^T$ where $\boldsymbol{H}_n = \boldsymbol{A}_N \odot \boldsymbol{A}_{N-1} \odot \cdots \odot \boldsymbol{A}_{n+1} \odot \boldsymbol{A}_{n-1} \odot \cdots \odot \boldsymbol{A}_1$ and $\odot$ denotes the Khatri-Rao product.

Based on the mode-n matrix unfolding for both $\underline{\boldsymbol{X}}$ and $\underline{\boldsymbol{M}}$, Problem (2) with $\boldsymbol{A}_{-n}$ being fixed can be recast as:

$$\min_{\boldsymbol{A}_n} \frac{1}{J_n} \sum_{j=1}^{J_n} f_j^n(\boldsymbol{A}_n; \boldsymbol{H}_n) \quad \text{s.t. } \boldsymbol{A}_n \in \mathbb{A}, \tag{8}$$

where $f_j^n(\boldsymbol{A}_n; \boldsymbol{H}_n) = \frac{1}{I_n} \sum_{i=1}^{I_n} f(\boldsymbol{X}_n(j, i), \boldsymbol{H}_n(j, :) \boldsymbol{A}_n(i, :)^T)$. According to the reformulation in (8), fiber sampling can be

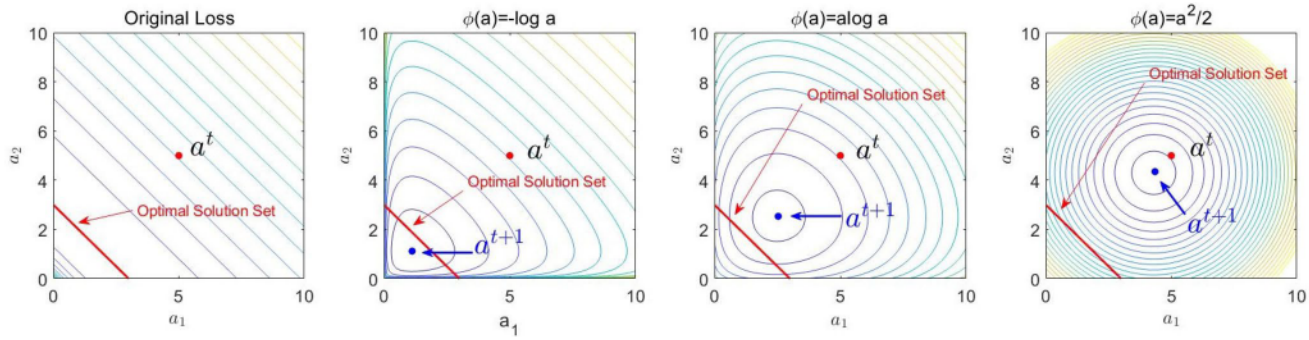


Fig. 2. Contour maps of upper bound functions over $[0, 10]^2$ indicated by (11) with different $\phi(a)$. The loss function $(x, m) = m - x\log(m + E)$, where $x = 3, E = 10^{-9}$, $m = \boldsymbol{h}^T\boldsymbol{a}$ with $\boldsymbol{h} = (1, 1)$ and $\boldsymbol{a} = (a_1, a_2)$, and $L = 1$ for all different $\phi(a)$. The solid red point is $\boldsymbol{a}^- = (5, 5)$ and the solid red line denotes the solution set, $\boldsymbol{h}^T\boldsymbol{a} = x - E$.

understood as sampling the index j from $\{1, \ldots, J_n\}$. Note that f_j^n only relates to the jth row of $\boldsymbol{X}_n$, where each row of $\boldsymbol{X}_n$ is a mode-n fiber. Thus, sampling f_j^n is equivalent to sampling rows of $\boldsymbol{X}_n$. Denote $\Omega^n \subseteq \{1, 2, \ldots, J_n\}$ as the index set of the sampled fibers. Then, the sampled version of Problem (8) becomes

$$\min_{\boldsymbol{A}_n \in \mathbb{R}^{I_n \times R}} \frac{1}{|\Omega^n| I_n} \sum_{j=1}^{|\Omega^n|} \sum_{i=1}^{I} f(\hat{\boldsymbol{X}}_n(j, i), \hat{\boldsymbol{H}}_n(j, :)\boldsymbol{A}_n(i, :)^T), \tag{9}$$

where $\hat{\boldsymbol{X}}_n = \boldsymbol{X}_n(\Omega^n, :)$ and $\hat{\boldsymbol{H}}_n = \boldsymbol{H}_n(\Omega^n, :)$. Usually, the size of Ω^n is chosen to satisfy $|\Omega^n| \ll J_n$. This may help substantially reduce the amount of data samples worked with (and thus the computational workload) in each iteration. In addition, the loss term in the finite summation of (9) composes the nonlinear function f and the linear mapping $\hat{\boldsymbol{H}}_n(j, :)\boldsymbol{A}_n(i, :)^T$. Fiber sampling preserves such special composition form always taking summation over all $i = 1, 2, \ldots, I_n$, i.e., one fiber relates to all entries of $\boldsymbol{A}_n$. This helps to derive a lightweight update of $\boldsymbol{A}_n$ in Section III-C.

We also note that if f is the Euclidean loss, the subproblem (9) is a (constrained) least squares problem, which is convex (if Π^n is a convex set) and can be relatively easily solved [25], [30]. However, when f is a non-Euclidean loss, the problem in (9) may still be nonconvex (e.g., β-divergence with $\beta < 1$) and challenging. One hopes to solve (9) using economical updates, which is often an art when non-Euclidean losses are considered.

C. Block-Wise Approximation Via Bregman Divergence

Note that all the loss functions $f(x, m)$'s given in Table II can be decomposed into a convex part $\bar{f}(x, m)$ plus a concave part $\hat{f}(x, m)$ as $f(x, m) = \bar{f}(x, m) + \hat{f}(x, m)$. To see how to make use of such a convex-concave property together with the linear composition form to derive a lightweight update of $\boldsymbol{A}_n$ from (9), we first give the definition of the Lipschitz-like convexity below.

Definition 2 (Lipschitz-like Convexity): The Lipschitz-like convexity condition for a function pair (ϕ, f) is that

$$\exists L > 0 \text{ such that } L\phi - f \text{ is convex on int dom } \phi,$$

where f and ϕ are differentiable convex functions.

A more general definition of the Lipschitz-like convexity condition can be found in [36], which is defined for the so-called Legendre function ϕ and lower semicontinuous convex function f. This more general definition contains Definition 2 as a special case.

Next, let us consider the (i, j)th component in (9). For notation simplicity, denote it as $f(x, \boldsymbol{h}^T\boldsymbol{a})$, where we have defined $x = \hat{\boldsymbol{X}}_n(j, i)$, $\boldsymbol{h}^T = \hat{\boldsymbol{H}}_n(j, :)$, and $\boldsymbol{a} = \boldsymbol{A}_n(i, :)^T$. The following lemma constructs a strongly convex surrogate function of $f(x, \boldsymbol{h}^T\boldsymbol{a})$ (a_r and h_r are the r-th components of $\boldsymbol{a}$ and $\boldsymbol{h}$ respectively):

Lemma 1: Let $\phi(\cdot) : \mathbb{R} \to \mathbb{R}$ be a strongly convex function. Suppose that for the function pair (ϕ, f) the following Lipschitz-like convexity condition holds:

$$\forall r, L(a_r) - f\left(x, \frac{h_r}{\bar{h}_r} a_r\right) \text{ is convex w.r.t. } a_r, \tag{10}$$

where $\bar{h}_r := \frac{h_r \bar{a}_r}{\boldsymbol{h}^T\bar{\boldsymbol{a}}}$, $\bar{a}_r$ and $L, h_r, \bar{a}_r > 0$, $\forall r$, are constants. Then, the following holds:

$$f(x, \boldsymbol{h}^T\boldsymbol{a}) \le f(x, \boldsymbol{h}^T\bar{\boldsymbol{a}}) + \langle \nabla f(x, \boldsymbol{h}^T\bar{\boldsymbol{a}}), \boldsymbol{a} - \bar{\boldsymbol{a}} \rangle + L D_\phi(\boldsymbol{a}, \bar{\boldsymbol{a}}), \tag{11}$$

where $D_\phi(\boldsymbol{a}, \bar{\boldsymbol{a}})$ is the Bregman divergence. The equality in (11) holds if $\boldsymbol{a} = \bar{\boldsymbol{a}}$.

Proof: See Appendix A in supplementary material. □

The upper bound constructed in Lemma 1 is reminiscent of the majorization-minimization (MM) scheme developed for β-divergence [28]. The difference here is the choice for $\phi(\cdot)$. The MM scheme in [28] suggested to choose $\phi(\cdot)$ to be the convex part of $f(\cdot)$ (up to a constant scaling) while Lemma 1 suggests to choose $\phi(\cdot)$ to 'fit' the geometry of the convex part of $f(x, \boldsymbol{h}^T\boldsymbol{a})$. Clearly, the condition in (10) is more general. The corresponding functions $(\phi(\cdot), f(x, \cdot))$ are referred to as the *Lipschitz-like convexity* function pair [36].

Lemma 1 suggests that if one can find the appropriate $\phi(\cdot)$, then problem (9) can be approximately solved via the following SMD update:

$$\boldsymbol{A}_n^{t+1} = \arg\min_{\boldsymbol{A} \in \mathbb{R}^{I_n \times R}} \langle \hat{\boldsymbol{G}}_n^t, \boldsymbol{A} - \boldsymbol{A}_n^t \rangle + \frac{1}{\eta_t} D_\phi(\boldsymbol{A}, \boldsymbol{A}_n^t), \tag{12}$$

where $\hat{\boldsymbol{G}}_n^t$ is the gradient estimation w.r.t. $\boldsymbol{A}_n$ using sampled data at iteration t and $D_\phi(\boldsymbol{A}, \boldsymbol{A}_n^t)$ is the Bregman divergence

TABLE III
EXPLICIT FORMS OF $\widetilde{G}_n^t$ FOR DIFFERENT $f(\cdot)$

Loss Function $f(\cdot)$	$[\mathcal{F}_n]I_n \cdot \hat{G}_n^t$
$\frac{1}{2}(x-m)^2$	$\left(\hat{X}_n^t - \hat{X}_n\right)^T H_n$
$\frac{x}{m+\epsilon} + \log(m+\epsilon)$	$\left[\left(\hat{X}_n^t + \epsilon\right) \circledast \left(\hat{X}_n^t + \epsilon\right)^{\circledast -2}\right]^T H_n$
$m - x\log(m+\epsilon)$	$\left[-\hat{X}_n \circledast \left(\hat{X}_n^t + \epsilon\right)^{\circledast -1} + 1\right]^T H_n$
$e^m - xm$	$\left[\exp(\hat{X}_n^t) - \hat{X}_n\right]^T H_n$
$\log(m+1) - x\log(m+\epsilon)$	$\left[\left(\hat{X}_n^t + 1\right)^{\circledast -1} - \hat{X}_n \circledast \left(\hat{X}_n^t + \epsilon\right)^{\circledast -1}\right]^T H_n$
$\log(1+e^m) - xm$	$\left[\exp(\hat{X}_n^t) \circledast \left(\exp(\hat{X}_n^t) + 1\right)^{\circledast -1} - \hat{X}_n\right]^T H_n$
β-divergence	$\left[(\hat{X}_n^t + \epsilon)^{\circledast \beta-2} \circledast (\hat{X}_n^t - \hat{X}_n + \epsilon)\right]^T H_n$

*In this table, $\hat{X}_n^t = \hat{H}_n A_n^t$

between A_n and A_n^t. For different $f(x, m)$ in Table II, the expressions of $\widehat{G}_n^t$ are summarized in Table III. The form of the gradient for general losses $f(x, m)$ was recently presented in [6, Theorem 3].

Compared with (7), the SMD step in (12) is only for the block variable A_n. The term $\langle \widehat{G}_n^t, A_n \rangle + D_{\psi}(A_n, A_n^t)$ in (12) can be regarded as a geometry-aware augmentation for properly approximating the loss in (9). If choosing L in (10) as $\frac{1}{\epsilon_t}$ such that (10) holds, then (12) minimizes an upper bound function for the loss in (9). In addition, considering $\psi(a) = \frac{1}{2}a^2$, then (12) reduces to the well-known projected SGD.

Remark 3: For non-Euclidean losses, the choice of $\psi(\cdot)$ can heavily affect the behavior of the algorithm. On the other hand, the flexibility of using different $\psi(\cdot)$'s also entails opportunities of developing fast non-Euclidean CPD algorithms. An illustrative example using the generalized KL loss is shown in Fig. 2. One can see that using $\psi(a) = a^2/2$, the progress from a^t to a^{t+1} is very small. However, by using $\psi(a)$'s that are more adapted to the cost function's geometry (reflected by the contour of the cost function), i.e., $\psi(a) = a\log a$ or $-\log a$, the progress in one iteration can be much larger[2].

D. Stochastic Mirror Descent for CPD

The SMD step in (12) specifies the update for one latent matrix A_n. Combining it with a random selection of block n, the proposed algorithm is summarized in Algorithm 1. The major advantage of using uniformly at random block selection is that such a scheme leads to an unbiased gradient estimation [25] (up to a constant scaling), which simplifies the convergence analysis. In essence, the proposed algorithm is a block-randomized (inexact) coordinate descent method, which admits a similar structure as the algorithm in [25] for CPD under the Euclidean loss. The key difference is that Algorithm 1 employs SMD to solve each subproblem inexactly, while the algorithm in [25] uses proximal gradient. For simplicity, we name it as Stochastic Mirror descent AlgoRiThm for CPD (`SmartCPD`).

Algorithm 1: Stochastic Mirror Descent (SMD) Algorithm.

Require: $X, A_1^0, A_2^0, \ldots, A_N^0$, $\{\eta_t\}_{t=0,1,\ldots}$

1: **for** $t = 0, 1, \ldots$, until meet some convergence criteria **do**
2: Uniformly sample n from $\{1, 2, \ldots, N\}$;
3: Uniformly sample fibers $\mathcal{F}_n \subseteq \{1, 2, \ldots, J_n\}$;
4: Compute the sampled gradient $\widehat{G}_n^t$;
5: $A_n^{t+1} = \arg\min_{A} \langle \widehat{G}_n^t, A \rangle + \frac{1}{\eta_t} D_{\psi}(A, A_n^t)$;
6: $A_i^{t+1} = A_i^t, \forall i \neq n$;
7: **end for**

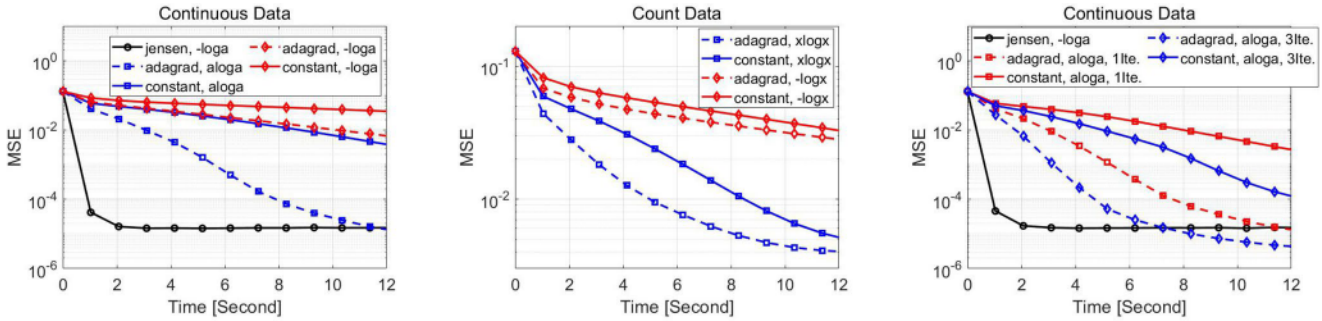
Remark 4: If no data sampling or block sampling is considered, the full batch version of `SmartCPD` subsumes many existing non-Euclidean and Euclidean matrix/tensor decomposition algorithms as its special cases—see the algorithms in [11], [14], [15], [17], [25], [28], [46]. In particular, consider the KL divergence. If one chooses $\psi(a) = a\log a$ and choose the step size η_t properly, `SmartCPD` becomes the MM algorithm [15]; when one uses $\psi(a) = a\log a$, then `SmartCPD` becomes the MD algorithm developed in [11]. This connection is not surprising, since MD includes many first-order approaches as its special cases. Nonetheless, using this connection, our convergence analysis (cf. Sec. IV) may also shed some light on the convergence behaviors of some existing algorithms whose convergence analyses were not considered at the time (e.g., [14]).

E. Practical Implementation

To implement the `SmartCPD` algorithm in practice, several key aspects need to be considered carefully. In particular, as in all stochastic algorithms, the step size-related parameter selection (i.e., η_t and L in `SmartCPD`) needs to be carefully carried out. In addition, the $\psi(\cdot)$ function should be chosen judiciously. In this subsection, we discuss these aspects in detail.

1) Choice of $\psi(\cdot)$: As indicated by Lemma 1, $\psi(\cdot)$ should be chosen to adapt to the geometry of $f(\cdot)$—e.g., by setting $\psi(\cdot)$ to be the convex part of $f(\cdot)$, if $f(\cdot)$ has convex-concave structure. In addition, the update (12) needs to solve a subproblem which minimizes the constructed surrogate loss over constraint set A_n. Practically, it is desirable that this subproblem can be solved easily, preferably in closed form. Hence, the choice of $\psi(\cdot)$ should be an integrated consideration of the function geometry of f and the constraint Ω_n. For example, consider $f(x, m) = x\log m$ with A_n being the probability simplex constraint, i.e., $A_n^T\mathbf{1} = \mathbf{1}, A_n(i,j) \geq 0, \forall i, j$ where $\mathbf{1}$ denotes a vector (of proper length) with all elements being 1. Then, it is preferred to choose $\psi(a) = a\log a$ rather than $\psi(a) = -\log a$ since the former admits a closed-form solution of the MD update, which is also known as the *exponential gradient descent* or *entropic descent* [29]. Several other examples of Ω_n under various ψ's which admit closed-form solutions are summarized in Table IV.

[2]In this illustrative example, we used $L = 1$. This is only for the ease of illustration. The L that satisfies the condition in (10) should be larger. Although the geometries are similar under $L > 1$ and $L = 1$, the former case is harder to visualize due to the condensed contours.



(a) Continuous Data (diff. step size and ϕ) (b) Count Data (diff. step size and ϕ) (c) Continuous Data (diff. no. of ite.)

Fig. 3. Numerical examples on 100 × 100 × 100 tensors with rank $R = 10$, where the generalized KL divergence is selected as the loss function and non-negative constraints are considered. For `SmartCPD`, the number of fibers is $2R = 20$, $b = 10^{-3}$ for adagrad step size (see (13)) and $\rho = 1$ for constant step size. The averaged mean squared error (MSE, see definition in (17)) of the latent matrix (averaged over 20 independent trials) is used as performance metric. The latent matrices are drawn from i.i.d. uniform distribution between 0 and 1. For continuous data case (Fig. 3(a) and (c)), Gaussian noise with SNR = 40 dB (see definition in Section V-A4) is added to the data; in count data case (Fig. 3(b)), each tensor entry is drawn from Poisson distribution with identity link function, i.e., $\mu(m) = m$.

TABLE IV
EXAMPLES FOR PAIR (ϕ, $\mathcal{A}_n$) AND CLOSED-FORM SOLUTION OF (12)

$\phi(\cdot)$	$\mathcal{A}_n$	Closed-form Solution
$-\log a$	non-negative	$\boldsymbol{A}_n^t \circledast \left[\hat{\boldsymbol{G}}_n^t \circledast (\boldsymbol{A}_n^t \oslash L) + 1\right]$
$a \log a$	non-negative	$\boldsymbol{A}_n^t \circledast \exp(-\hat{\boldsymbol{G}}_n^t \oslash L)$
a^c $(c > 1$ or $c < 0)$	non-negative	$\left[(\boldsymbol{A}_n^t)^{\circledast(c-1)} - \frac{1}{c}\hat{\boldsymbol{G}}_n^t \oslash L\right]^{\circledast\frac{1}{c-1}}$
$a \log a$	simplex	$\text{colnorm}(\boldsymbol{A}_n^t \circledast \exp(-\hat{\boldsymbol{G}}_n^t \oslash L))$
x^2	many forms	refer [25], [47]

*In this table, $\text{colnorm}(\boldsymbol{A}) = \boldsymbol{A}\boldsymbol{D}(\boldsymbol{1}^T\boldsymbol{A})^{-1}$, which denotes the column-wise normalization operation.

TABLE V
($\hat{\boldsymbol{G}}_n^t$) BASED ON JENSEN'S INEQUALITY

Loss Function	$\phi(a)$	$\mathcal{F}_n \mid I_n \cdot I_n^t$
Eucl. Dis.	$\frac{1}{2}a^2$	$\frac{1}{2}(\boldsymbol{X}_n^t)^T \boldsymbol{H}_n \oslash \boldsymbol{A}_n^t$
IS Div.	$\frac{1}{a}$	$\boldsymbol{A}_n^t \circledast \left[\hat{\boldsymbol{X}}_n \oslash (\hat{\boldsymbol{X}}_n^t)^{\cdot 2}\right]^T \hat{\boldsymbol{H}}_n$
KL Div.	$-\log a$	$(\boldsymbol{A}_n^t)^T \circledast \left[\hat{\boldsymbol{H}}_n^T (\hat{\boldsymbol{X}}_n \oslash (\hat{\boldsymbol{X}}_n^t))\right]$
β-Div. $(\beta > 1)$	a^β	$\frac{1}{\beta}(\hat{\boldsymbol{X}}_n^t)^{\cdot\beta-1})^T \boldsymbol{H}_n \oslash (\boldsymbol{A}_n^t)^{\cdot\beta-1}$
β-Div. $(\beta < 1)$	$a^{\beta-1}$	$\frac{1}{(1-\beta)}[\hat{\boldsymbol{X}}_n \circledast (\hat{\boldsymbol{X}}_n^t)^{\cdot\beta-2})^T \boldsymbol{H}_n \oslash (\boldsymbol{A}_n^t)^{\cdot\beta-2}$

*In this table, $\hat{\boldsymbol{X}}_n^t = \hat{\boldsymbol{H}}_n \boldsymbol{A}_n^t$.

2) Choice of ρ: Since the Bregman divergence $D_\phi(\boldsymbol{A}, \boldsymbol{A}^t)$ is defined in an entry-wise summation form, scaling $D_\phi(\boldsymbol{A}, \boldsymbol{A}^t)$ with a constant $\frac{1}{\rho}$ for all entries may be less effective for approximating the loss in (9). In this work, we propose to use coordinate-dependent step size, i.e., different positive scaling factors $\gamma_n^t(i_n, j)$'s for different coordinates (i_n, j)'s, where $i_n \in [I_n]$, $j \in [R]$, and $\gamma_n^t \in \mathbb{R}^{I_n \times R}$. Consequently, the term $\frac{1}{\rho} D_\phi(\boldsymbol{A}, \boldsymbol{A}_n^t)$ in (12) becomes

$$\sum_{i_n, j} \gamma_n^t(i_n, j) D_\phi(\boldsymbol{A}_n(i_n, j), \boldsymbol{A}_n^t(i_n, j)).$$

As illustrated in Fig. 3, such a coordinate-specific γ_n^t empirically helps accelerate convergence. In what follows, we provide two schemes on choosing $\gamma_n^t(i_n, j)$'s.

- *Jensen's inequality based choice:* One important criteria is to choose $\gamma_n^t(i_n, j)$ such that the inequality in (11) holds for each coordinate (i_n, j). Such a choice makes the local approximation in (12) be an upper bound of (9). In the case where $f(\cdot)$ enjoys the convex-concave property and the constraint sets $\mathcal{A}_n$ satisfy $\mathcal{A}_n \subseteq \mathbb{R}_+^{I_n \times R}$, $\forall n$, Lemma 1 implies that one can choose $\phi(\cdot)$ to be the convex part of $f(\cdot)$. Consequentially, $\gamma_n^t(i_n, j)$ can be derived based on the Jensen's inequality in each iteration. A number of examples of $\gamma_n^t(i_n, j)$ with respect to different $f(\cdot)$ are given in Table V, where the derivation are based on (18) and (19) in supplementary material. We note that the Jensen's inequality based $\gamma_n^t(i_n, j)$ choice is often used by popular methods such as MM [48] and block successive upper bound minimization (BSUM) [49] to simplify the solution of the subproblems. The difference here is that our construction uses sampled fibers instead of the whole data. Such Jensen's inequality-based step-size choice also works well with certain non-Euclidean losses under our stochastic settings, as will be seen in the experiments.

- *Adaptive step size based choice:* The deep learning community has developed a number of effective adaptive step size scheduling methods, e.g., the Adagrad [50] and Adam type schemes [51]. These schemes typically exploit the past iterations' gradient information to scale the current sampled gradient in a coordinate-wise manner. The upshot of these methods is that they often require very small amount of step size tuning, yet offer highly competitive empirical performance; also see theoretical understanding in [51]. Under the Bregman divergence, the adaptive step size schemes can be used to 'schedule' $\gamma_n^t(i_n, j)$ in each iteration. Specifically, we propose the following Adagrad step size rule for $\gamma_n^t(i_n, j)$,

$$\gamma_n^t(i_n, j) = \frac{\rho}{\sqrt{\sum_{t'=0}^{t} [\hat{\boldsymbol{G}}_n^{t'}(i_n, j)]^2 + b}}, \tag{13}$$

where $\hat{\boldsymbol{G}}_n^{t'}(i_n, j)$ denotes the (i_n, j)-th entry of $\hat{\boldsymbol{G}}_n^{t'}$ and $b > 0$ is a constant to ensure $\gamma_n^t(i_n, j) > 0$. Note that the adaptive step size was considered in fiber-sampling based stochastic Euclidean CPD in [25] and entry-sampling based stochastic non-Euclidean CPD [7], and encouraging results were observed in both cases.

Remark 5: It can be observed from numerical examples in Fig. 3 that, for continuous data (Fig. 3(a)), using different $\gamma_n(i_n, j) > 0$ for different coordinate (i_n, j) exhibits much faster convergence behavior than using a constant step size

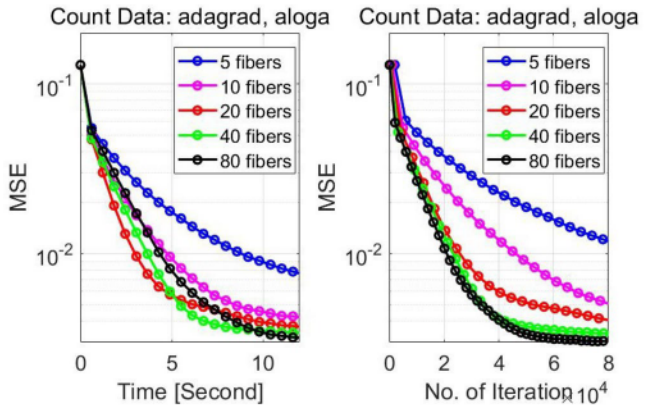


Fig. 4. Averaged MSE of count tensor with different number of fibers ($I_n = 100$ and $R = 10$).

for all coordinates. For count data (Fig. 3(b)), we have similar observations. Another observation from Fig. 3 is that, for count or binary data, the Jensen's inequality based step size scheme may be less competitive, especially when the data contains many zeros. The zero entries may make $\boxtimes_n(i_n, j)$ very small, thereby causing numerical issues. The 'adagrad' scheme in (13) has empirically much more stable convergence behavior under such circumstances. In practice, a user could make a choice between a more conservative algorithm with better convergence understandings (constant step size $\square_t$) or a more aggressive one with less theoretical guarantees (coordinate-wise step size $\boxtimes_n^t$). This may be analogous to constant step size SGD or Adam/Adagrad type SGD, where the former has better convergence understanding while the latter works faster for some problems in practice.

3) Inner Iterations: The proposed Algorithm 1 only contains one iteration per block. Nonetheless, one can also extend it to multiple SMD updates (i.e., inner iterations) per block. We have observed that implementing with a few more inner iterations could improve the practical convergence behavior—as shown in Fig. 3(c). There are two ways of having multiple inner iterations. The first way is to repeat lines 3-5 in Algorithm 1 to update $\boldsymbol{A}_n$ several times before moving to the next block, where the fibers are re-sampled for each inner iteration. The second way is that, for fixed sampled fibers, repeat lines 4-5 for multiple times. Both methods work reasonably well in practice. In Fig. 3(c), we use the latter because it uses less samples and has similar performance (in our experience) compared with re-sampling per inner iteration.

4) Number of Fibers: By regarding $f(x, m)$ as a certain 'distance' measure between x and m, the SMD step (12) in `SmartCPD` can be understood as approximately solving the linear system $\hat{\boldsymbol{X}}_n = \hat{\boldsymbol{H}}_n \boldsymbol{A}_n^T$ under such 'distance' measure. From this viewpoint, the number of fibers should be chosen to be larger than rank R, e.g., $|\mathcal{F}_n| = 2R$. In Fig. 4, with the same problem setting as in Fig. 3(b), `SmartCPD` (adagrad step size, $\square(a) = a \log a$, 1 inner iteration) with different numbers of fibers are compared. Clearly, increasing number of fibers improves per iteration efficiency but also needs more time to compute the sampled gradient. Using $2R$ fibers seems to strike a good balance between the gradient estimation's variance and the per-iteration complexity.

IV. Convergence Analysis

In general, convergence guarantees of stochastic tensor decomposition algorithms are difficult to establish, as nonconvex constrained optimization problems are intrinsically harder to analyze under stochastic settings. The non-Euclidean version is even more so, since the sampled subproblems may still be nonconvex. There has not been an analytical framework for SMD based non-Euclidean tensor/matrix factorization. The nonconvex block SMD in the optimization literature [31] is the most closely related to our algorithmic framework. However, the convergence analysis there does not cover the proposed algorithm. Specifically, the convergence of the algorithm in [31] hinges on some special *incremental block averaging steps*, which is not used in our algorithm. More importantly, the algorithm in [31] requires that the block-wise gradient estimation error vanishes to zero when the iterations progress. This may require implementing the algorithm with an active variance reduction technique, e.g., increasing the batch size in each iteration [25], which is not entirely realistic, and it is somewhat against the purpose of using stochastic algorithms.

Our goal is to offer tailored convergence analysis for `SmartCPD` that does not rely on conditions like incremental block averaging or vanishing gradient estimation error. We note that for constrained problems, convergence analysis for SMD with adaptive step size $\boxtimes_n$ is very challenging. Theoretical understanding of SGD with adaptive step size scheme was recently discussed in [51]—but the SMD case is still an open problem. In this work, we focus on the case where $\boxtimes_n^t(i_n, j) = \square_t > 0$ are all identical.

By using Bregman divergence as a proximity measure, the recent notion of *relative smoothness* [41] (or called L-smooth adaptable [37]) can be defined as below:

Definition 3 (Relative Smoothness): A continuous differentiable function f is relatively smooth to a strongly convex function $\square$ with $\square L$ $(0 < L < \square)$ if $L\square - f$ is convex on $\operatorname{int}\operatorname{dom}\square$

If f is convex, relative smoothness (Definition 3) becomes the Lipschitz-like convexity in Definition 2. Also, there are several different concepts which are closely related to relative smoothness, e.g., relative weak convexity [39], [40], and relative continuity [38], which were used in several recent works to analyze single-block SMD type algorithms' convergence; see [37]–[40]. Our analysis leverages the notion of relative smoothness, generalizing this prior work to cover multi-block `SmartCPD`. Particularly, the proof takes advantage of the multilinear low-rank tensor structure and the block-randomized fiber sampling strategy to brige the gap between the single-block and multi-block cases.

The objective function in (2) is denoted as $F(\boldsymbol{A})$ with $\boldsymbol{A} = (\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_N)$. Then, Problem (2) can be re-expressed as:

$$\min_{\boldsymbol{A}} \; F(\boldsymbol{A}) + h(\boldsymbol{A}) \tag{14}$$

where $h(\boldsymbol{A}) = \sum_{n=1}^{N} h_n(\boldsymbol{A}_n)$ and $h_n(\boldsymbol{A}_n)$ is the indicator function of set $\mathcal{A}_n$, i.e., $h_n(\boldsymbol{A}_n) = 0$ if $\boldsymbol{A}_n \in \mathcal{A}_n$ and otherwise $h_n(\boldsymbol{A}_n) = \infty$.

Our first observation is as follows:

Lemma 2: If $\{\boldsymbol{A}^t_n\}$ all reside in a compact set for all n, there exists $0 < L < \infty$ such that for any $\boldsymbol{A}^t$, $\boldsymbol{A}$ in this compact set function F is relatively smooth to ψ with L, i.e.,

$$|F(\boldsymbol{A}^t) - F(\boldsymbol{A}) - \langle \nabla F(\boldsymbol{A}), \boldsymbol{A}^t - \boldsymbol{A} \rangle| \le L D_\psi(\boldsymbol{A}^t, \boldsymbol{A}), \tag{15}$$

where $\psi(\cdot)$ can be any strongly convex function.

Proof: See Appendix B in the supplementary material.

It is easy to see Lemma 2 immediately implies inequality (11) in Lemma 1 summed over all the entries. Also, Lemma 2 has no restriction on the choice of $\psi(\cdot)$. This brings much convenience for the practical usage of Algorithm 1. More importantly, Lemma 2 holds for the all latent factors simultaneously (as opposed to one latent factor as in Lemma 1). This is critical for establishing the convergence guarantee. Also, we note that inequality (15) is a generalization of standard Lipschitz-continuous gradient property of $F(\boldsymbol{A})$ under Bregman divergence, which is also known as *relative smoothness* [52]. We remark that the assumption that $\{\boldsymbol{A}^t_n\}$ live in a compact set is not easy to check in advance, yet it is not hard to satisfy. When the constraints $\mathcal{A}$ are compact sets, then this assumption is naturally satisfied per the defined update in (12). Even if $\mathcal{A}^n$ is not bounded, unbounded iterates are rarely (if not ever) observed in our extensive numerical experiments.

Based on Lemma 2, convergence of the proposed Algorithm 1 can be guaranteed for any strongly convex function $\psi(\cdot)$ (see Theorem 1). Our convergence analysis starts by using the following "reference" function in each iteration:

$$\mathcal{L}(\boldsymbol{A}; \boldsymbol{A}^t) := F(\boldsymbol{A}) + h(\boldsymbol{A}) + \frac{1}{2\eta} D_\psi(\boldsymbol{A}, \boldsymbol{A}^t)$$

where $0 < \eta < \frac{1}{2L}$ is a constant such that $\mathcal{L}(\boldsymbol{A}; \boldsymbol{A}^t)$ is strongly convex in $\boldsymbol{A}$. The minimal value and minimizer of $\mathcal{L}(\boldsymbol{A}; \boldsymbol{A}^t)$ for a given $\boldsymbol{A}^t$, which are also known as *Bregman Moreau envelope* $M(\boldsymbol{A}^t)$ and *Bregman proximal mapping* $\mathcal{T}(\boldsymbol{A}^t)$ [53], are defined as

$$M(\boldsymbol{A}^t) = \min_{\boldsymbol{A}} \mathcal{L}(\boldsymbol{A}; \boldsymbol{A}^t), \quad \mathcal{T}(\boldsymbol{A}^t) = \arg\min_{\boldsymbol{A}} \mathcal{L}(\boldsymbol{A}; \boldsymbol{A}^t).$$

Denote $\hat{\boldsymbol{A}} = \mathcal{T}(\boldsymbol{A})$ for a given $\boldsymbol{A}$, we use the following lemma to show that $D_\psi(\hat{\boldsymbol{A}}, \boldsymbol{A})$ is a measure for attaining stationary points of Problem (14).

Lemma 3: $\boldsymbol{A}$ is a stationary point of Problem (14), i.e., $\mathbf{0} \in \nabla F(\boldsymbol{A}) + \partial h(\boldsymbol{A})$, where $\partial h(\boldsymbol{A})$ denotes the subgradient, if and only if $D_\psi(\hat{\boldsymbol{A}}, \boldsymbol{A}) = 0$.

Proof: see Appendix C in the supplementary material.

The key step for analyzing Algorithm 1 is to quantify its one iteration behavior:

Lemma 4: Suppose that $\{\boldsymbol{A}^t\}$ all reside in a compact set for all n, and that ψ is a ψ-strongly convex function. Let $\boldsymbol{A}^{t+1}$ be generated by Algorithm 1 at iteration t, then we have

$$\mathbb{E}\, M(\boldsymbol{A}^{t+1}) \le M(\boldsymbol{A}^t) - c_1 \rho_t D_\psi(\hat{\boldsymbol{A}}^t, \boldsymbol{A}^t) + c_2 \rho_t^2 \mathbb{E}\, \|\boldsymbol{G}^t\|^2, \tag{16}$$

where $c_1 = \frac{(1-\rho_2)}{4\eta}$ and $c_2 = \frac{1}{4\eta}$. The expectation in (16) is taken over the random variable responsible for fiber sampling in iteration t.

Proof: See Appendix D in the supplementary material.

The lemma implies that, in expectation, if the step size ρ_t is properly chosen, then $M(\boldsymbol{X}^t)$ decreases after every iteration. Using Lemma 4 as a stepping stone, we show our main result:

Theorem 1: Suppose that the assumptions in Lemma 4 hold.

1) for diminishing step size ρ_t satisfying $\sum_{t=1}^{\infty} \rho_t = \infty$ and $\sum_{t=1}^{\infty} \rho_t^2 < \infty$, Algorithm 1 converge to a stationary point in expectation,

$$\liminf_{t\to\infty} \mathbb{E}\, D_\psi(\hat{\boldsymbol{A}}^t, \boldsymbol{A}^t) = 0;$$

2) for a constant step size $\rho_t = \frac{1}{\sqrt{T}}$, $\frac{1}{\sqrt{T}}$-stationary solution of Problem (14) can be obtained by Algorithm 1 within T iterations,

$$\min_{1\le t\le T} \mathbb{E}\, D_\psi(\hat{\boldsymbol{A}}^t, \boldsymbol{A}^t) \le C/\sqrt{T},$$

where $C > 0$ is a constant. The expectations are taken over the fiber and block sampling in all iterations jointly.

Proof: See Appendix E in the supplementary material.

V. Simulations

We use synthetic and real data experiments to showcase the effectiveness of `SmartCPD`. The MATLAB code is available at https://github.com/WQPu/SmartCPD.git.

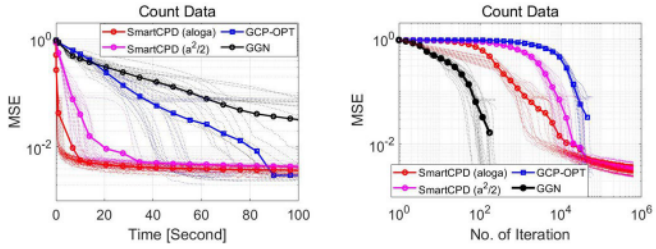
A. Synthetic Data

We evaluate the numerical performance on different types of synthetic data, i.e., continuous, count, and binary data, under various non-Euclidean losses in Table II (c in Table II is set to be 10^{-6}).

1) Baselines and Performance Metric: We use two recent competitive algorithms as our main baselines. The first one is an entry-sampling based stochastic non-Euclidean CPD optimization algorithm, namely, `GCP-OPT`, proposed in [7]. The second baseline is the generalized Gauss-Newton (`GGN`) method for non-Euclidean CPD [9]. The `GCP-OPT` method is implemented in Tensor Toolbox [54] and 'Adam' is selected as the optimization solver. The sampling rule of `GCP-OPT` is the default 'uniform' setting for dense tensors unless we specify it for particular examples. The `GGN` method [9] is implemented by `nlsb_gndl` and the 'preconditioner' is set as 'block-Jacobi'; see more details in the Tensorlab toolbox [55]. For the proposed `SmartCPD`, different choices of step size ρ_t and function $\psi(\cdot)$ will be specified according to data type.

We generate third-order tensors with different sizes and ranks, each dimension of the tensor keeps the same as $I = I_1 = I_2 = I$. For the two stochastic algorithms, in each iteration, `SmartCPD` samples $2R$ fibers[3] and `GCP-OPT` samples $2I_nR$ entries. This makes the two algorithms use the same number of samples per iteration, though it differs from the recommendation in [7, Section 5.1]. The other hyperparameters of the baselines follow their default settings. The MSE of the latent matrices is used as a performance metric [25], which is

[3]Note that that using $2R$ is not the only choice, but based on our experience.



(a) MSE over time. (b) MSE over no. of iteration.

Fig. 5. MSE of count tensor ($I = 100$ and $R = 20$).

defined as $\text{MSE} = \frac{1}{N}\sum_{n=1}^{N} \text{MSE}_n$ with

$$\text{MSE}_n = \min_{\pi(r)\ [R]} \sum_{r=1}^{R} \frac{\left\| \boldsymbol{A}_n(:,\pi(r)) - \hat{\boldsymbol{A}}_n(:,r) \right\|_2}{\left\| \boldsymbol{A}_n(:,\pi(r)) \right\|_1 \left\| \hat{\boldsymbol{A}}_n(:,r) \right\|_1}, \tag{17}$$

where $\hat{\boldsymbol{A}}_n$ denotes the estimated $\boldsymbol{A}_n$ and $\pi(1), \ldots, \pi(R)$ represents a permutation of the set $[R] = \{1, \ldots, R\}$, which is used to fix the intrinsic column permutation in CPD.

2) Count Data: We first evaluate the performance on count data tensors. For the proposed SmartCPD, η_n^t is scheduled following (13). We use $\psi(a) = a \log a$ and $\phi(a) = \frac{1}{2}a^2$ in the SMD update. The constant b in (13) is 10^{5} for all simulation trials.

We use the loss function $f(x, m) = m - x \log(m + c)$ as in [15] and draw the latent matrices $\boldsymbol{A}_1$, $\boldsymbol{A}_2$, and $\boldsymbol{A}_3$ from i.i.d. uniform distribution between 0 and $A_{\max}$, where $A_{\max} = 0.5$ is a positive constant. For each column of the latent matrices, 5% elements are randomly selected and replaced by i.i.d. samples from uniform distribution between 0 and $10A_{\max}$. This way, the elements have more diverse scales. The observed count data tensor $\boldsymbol{\mathcal{X}}$ is generated following the Poisson distribution, i.e., $\boldsymbol{\mathcal{X}}_i \sim \text{Poisson}(\boldsymbol{\mathcal{M}}_i)$. We set $\boldsymbol{A}_n$ for all n as the nonnegative orthant.

Fig. 5(a) shows the performance of the algorithms under $I = 100$ and $R = 20$, where the solid lines correspond to the average MSEs and dashed lines are for individual trials. One can see that SmartCPD improves the MSE quickly in all trials. On average, it brings the MSE below 10^{-2} using less than 10 seconds, while the best baseline, i.e., GCP-OPT takes at least around 40 seconds to reach the same MSE level. Fig. 5(b) shows the MSEs of the algorithms against the number of iteration. Clearly, GGN has the best iteration complexity since it exploits second-order information based on all samples. Both SmartCPDs enjoy better iteration complexity than GCP-OPT. In addition, SmartCPD ($a \log a$) performs better than SmartCPD ($a^2/2$) (which is equivalent to the BrasCPD algorithm in [25]). GCP-OPT achieves smaller MSE for some trials. Note that both SmartCPD ($a^2/2$) and GCP-OPT can be viewed as SGD type algorithms with fiber sampling and entry sampling, respectively.

In many trials, SmartCPD uses at least one order of magnitude fewer data entries to reach MSE= 10^{-2}. Fig. 6 shows a closer look at the runtime breakouts of the two stochastic algorithms. It can be found that both use similar time for sampling but GCP-OPT requires slightly more time for computing the gradient as well as updating all latent matrices. These

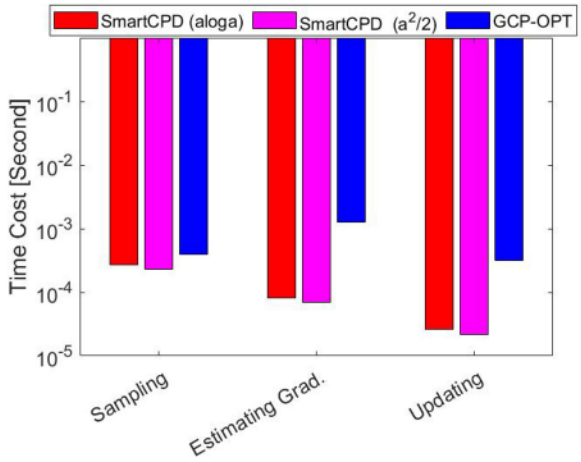


Fig. 6. Detailed time cost per iteration of SmartCPD and GCP-OPT algorithms ($I = 100$ and $R = 20$).

TABLE VI
MSE AFTER 100 SECONDS ($I = 100$ AND DIFF. R)

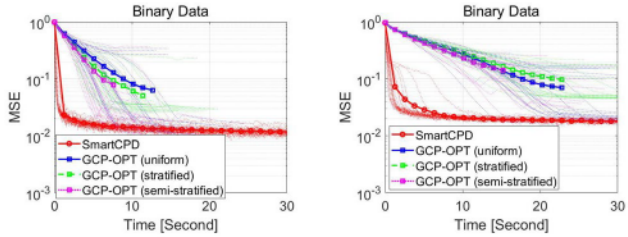
Rank R		10	20	50
SmartCPD ($a \log a$)	mean	2.7E-3	3.7E-3	6.6E-3
	median	2.7E-3	3.7E-3	6.5E-3
GCP-OPT	mean	9.6E-3	0.028	0.408
	median	2.3E-3	3.3E-3	0.406
GGN	mean	0.023	0.031	0.288
	median	1.9E-3	6.5E-3	0.281

results clearly show that using non-Euclidean proximal term (e.g., $a \log a$) and fiber sampling both play important roles in accelerating KL-divergence CPD.

In Table VI, the achieved MSEs after 100 seconds when $I = 100$ and $R = 10, 20, 50$ are compared. It can be observed that SmartCPD ($a \log a$) outputs MSEs that are smaller than 10^{-2} for all cases, while GGN and GCP-OPT could not reach MSE $= 0.1$ in 100 seconds when $R = 50$, in terms of both mean and median. Also, GCP-OPT and GGN can often attain a slightly smaller MSE value relative to SmartCPD when the rank is not large, e.g., $R = 10$.

3) Binary Data: Next, we evaluate the performance on binary tensors. We use the loss function that is related to the MLE of Bernoulli tensors, i.e., $f(x, m) = \log(m + 1) - x \log(m + c)$ (cf. Table II). We set $\psi(a) = a \log a$ and non-negativity constraints are considered. Note that GGN is developed for ψ-divergence, and thus cannot be used for this loss function. Hence, we only use GCP-OPT to benchmark our algorithm in the binary case. We set $I = 100$ as before and each entry of the binary tensor is generated from the Bernoulli distribution, i.e., $\boldsymbol{\mathcal{X}}_i = 1$ with probability $\boldsymbol{\mathcal{M}}_i/(1 + \boldsymbol{\mathcal{M}}_i)$. The latent matrices are generated as in Sec. V-A2. First, we set $R = 5, 10$ and the constant $A_{\max} = 0.3$. This results in the generated binary tensor having about 4% ($R = 5$) and 8% ($R = 10$) non-zero entries, respectively. For such sparse binary tensors, 'stratified' and 'semi-stratified' sampling rules in GCP-OPT are also included in the comparison.

Fig. 7 shows the convergence curves (averaged over 20 independent trials) and SmartCPD achieves the fastest convergence speed. Also, GCP-OPT with 'semi-stratified' shows better



(a) $R = 5$, $\approx 4\%$ non-zero entries. (b) $R = 10$, $\approx 8\%$ non-zero entries.

Fig. 7. MSE of binary tensor ($I_n = 100$ and $R = 5, 10$).

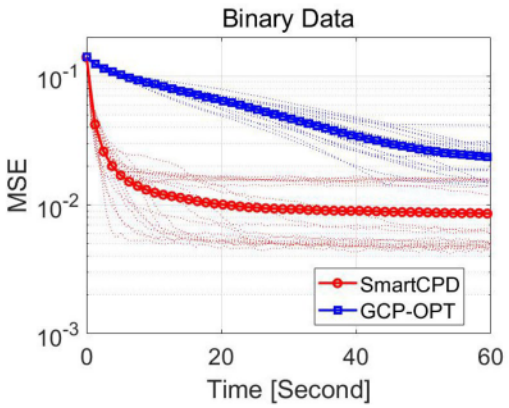


Fig. 8. MSE of binary tensor ($I_n = 100$, $R = 20$, and 15% non-zero entries).

convergence behavior than the other two sampling rules (i.e., 'uniform' and 'stratified').

In Fig. 8, we increase rank to $R = 20$ and set the constant $A_{\max}$ 0.2, 0.3, 0.4, 0.5 to generate four different binary data tensors, with about 5\%, 15\%, 28\%, 40\% nonzero entries, respectively. For $R = 20$, we observe GCP-OPT with 'uniform' sampling rule performs more consistent than the other two sampling rules and hence, in this figure we only present the result of GCP-OPT with 'uniform' sampling rule. Fig. 8 shows the MSEs against time of 20 independent trials when the tensor has 15% nonzero entries. Similar as before, SmartCPD requires much shorter time to achieve MSE 10^{-}. In addition, the histograms of MSEs after 60 seconds are presented in Fig. 9. One can see from there that that SmartCPD consistently outperforms the baseline, and the advantage is more articulated when the data becomes denser.

4) Continuous Data: We also evaluate the performance of SmartCPD on continuous tensors under divergence. We consider the multiplicative Gamma noise, i.e.,

$$\texttt{Gamma}: \boldsymbol{X}_i = \boldsymbol{M}_i \cdot \boldsymbol{N}_i,$$

in which $\boldsymbol{N}_i$ is i.i.d. Gamma noise. The SNR for Gamma noise is defined the same way as in [9], which is determined by the shape parameter of Gamma distribution. The latent matrices $\boldsymbol{A}_1$, $\boldsymbol{A}_2$, and $\boldsymbol{A}_3$ are drawn from the i.i.d. uniform distribution between 0 and 1 and non-negativity constraints are considered. Since the div. loss functions satisfy the convex-concave property, function and $_n^t$ are used based on the Jensen's inequality for SmartCPD; see details in Table V. We find such setup particularly efficient for dealing with dense and continuous tensors under divergence.

TABLE VII
MSE OF CONTINUOUS DATA TENSOR WITH GAUSSIAN NOISE AFTER 100 SECONDS ($I = 300$, $R = 20$, AND DIFFERENT)

β		-1	-0.5	0	0.5	1	2	3
SmartCPD	mean	2.4E-5	2.1E-5	1.9E-5	2.4E-5	3.7E-5	3.4E-5	1.5E-5
	med.	2.4E-5	2.1E-5	1.9E-5	2.4E-5	3.6E-5	3.4E-5	1.5E-5
GCP-OPT	mean	9.2E-4	1.5E-4	1.4E-4	5.8E-4	4.1E-5	1.7E-4	1.7E-4
	med.	4.8E-5	4.1E-5	2.2E-5	3.8E-5	1.8E-5	2.9E-5	3.3E-5
GGN	mean	0.304	0.287	0.299	0.139	0.093	<1E-6	0.320
	med.	0.299	0.306	0.297	0.134	0.090	<1E-6	0.326

Fig. 10 shows simulations where the tensor has a size of 300 300 300 and $R = 20$. The average MSEs over 20 independent trials for = 0 with SNR = 20 dB are compared. Clearly, the two stochastic algorithms, i.e., SmartCPD and GCP-OPT have faster convergence than GGN. Further, SmartCPD is even faster than GCP-OPT, especially in the beginning. We also observe that the original SmartCPD's MSE saturates at a certain level after few iterations.

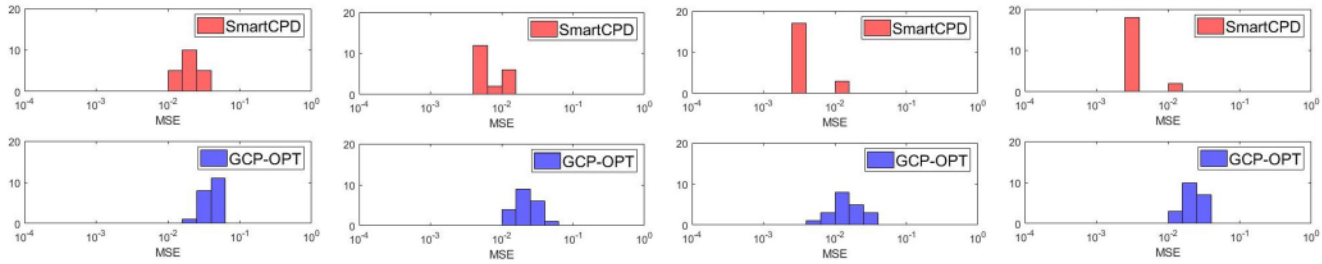
Next, we test the proposed algorithm on CPD under various s, i.e., 1, 0.5, 0, 0.5, 1, 2, 3 . The data is generated using an additive noise model, i.e., $\boldsymbol{X}_i = \boldsymbol{M}_i + \boldsymbol{N}_i$, where $\boldsymbol{N}_i$ denotes zero-mean i.i.d. Gaussian noise and $\boldsymbol{M}$ is generated as in the previous simulation.

In Table VII, the mean and median MSE over 20 independent trials after 100 seconds are compared with SNR = 40 dB. SmartCPD achieves smaller mean MSE than other two algorithms and both SmartCPD and GCP-OPT have similar median MSEs. This is because 100 seconds is not enough for GCP-OPT to ensure all trials converge; see the convergence curves for = 1 in Fig. 11. As a deterministic second-order algorithm, GGN attains better MSEs for = 2 This shows effectiveness of exploiting the second-order information for handling the Euclidean loss.

5) Column Constraints: In Fig. 12, we evaluate the performance for problems with simplex constraints, i.e., $\boldsymbol{A}_n^T \mathbf{1} = \mathbf{1}, \boldsymbol{A}_n(i,j) \geq 0, \; i,j$ and the loss function is $f(x,m) = m - x\log(m+c)$ (KL div.). Tensors that represent joint probability distributions with $I = 100$ and $R = 10$ are considered; see [11], [43]–[45], [56] for details. The latent matrices $\boldsymbol{A}_1$, $\boldsymbol{A}_2$, and $\boldsymbol{A}_3$ are drawn from i.i.d. uniform distribution between 0 and 1, then each column is normalized so that it represents a probability mass function. For SmartCPD, the step size scheme in (13) with $(a) = a \log a$ is used. Note that both the baselines are not able to handle such simplex constraints, while this problem frequently arises in probabilistic tensor decomposition; see [11], [43], [44], [57]. To benchmark our algorithm, we use the deterministic block MD algorithm in [11]. In Fig. 12, one can see that SmartCPD has much faster convergence behavior than the MD algorithm due to stochastic sampling.

B. Real Data

1) Chicago Crime Data: We apply the algorithms to the Chicago Crime dataset. The dataset records crime reports in the



(a) $\approx 5\%$ nonzero entries (b) $\approx 15\%$ nonzero entries (c) $\approx 28\%$ nonzero entries (d) $\approx 40\%$ nonzero entries

Fig. 9. Histogram of MSE after 60 seconds of $100 \times 100 \times 100$ binary tensor (rank 20) with different level of sparsity.

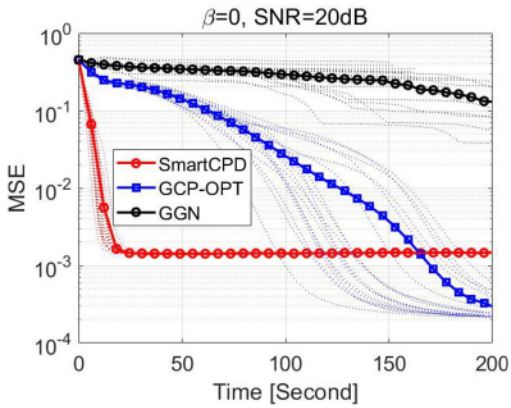


Fig. 10. MSE of continuous data tensor with multiplicative Gamma noise ($I_n = 300$, $R = 20$, and $= 0$).

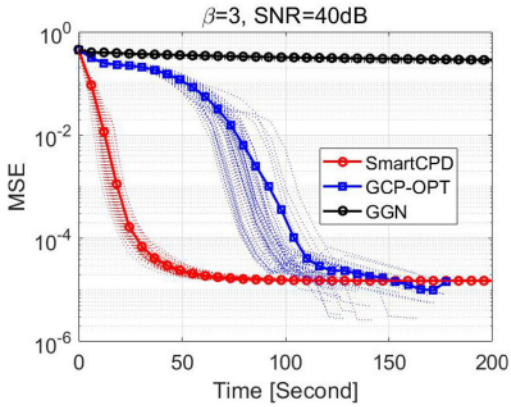


Fig. 11. MSE of continuous data tensor with additive Gaussian noise ($I_n = 300$, $R = 20$, and $= 3$).

city of Chicago, Illinois, United States, between January 1, 2001 to December 11, 2017. The original dataset is published in the official website of the city of Chicago (www.cityofchicago.org). Here, we use the version in [58]. The data is in the form of a fourth order tensor (day $\times$ hour $\times$ crime community) with integer entries representing the number of crimes reported. The size of the tensor is $6186 \times 24 \times 77 \times 32$. It has 5,330,678 (1.5%) nonzero entries.

We choose the loss function corresponding to the Poisson distribution, i.e., $f(x, m) = m - x \log(m + c)$ (see Table II), and non-negativity constraints are considered for the latent matrices. In every iteration, 40 fibers are used by `SmartCPD`. For

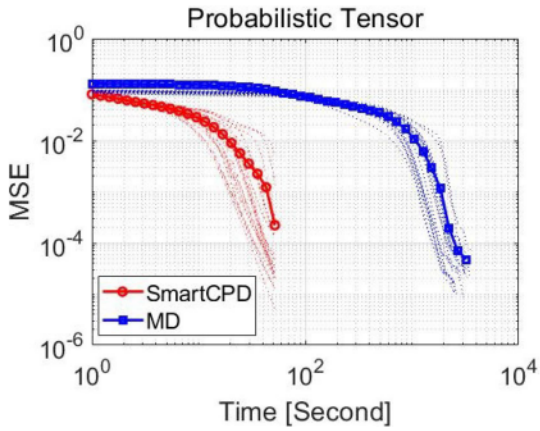


Fig. 12. MSE of continuous data tensor with simplex constraint ($I_n = 100$, $R = 10$, and $= 1$).

`GCP-OPT`, $40 \times (6186 + 24 + 77 + 32)/4$ tensor entries are sampled in every iteration, such that we sample an equal amount of data on average for both the algorithms, though it differs from the recommendation in [7, Section 5.1]. The uniform sampler of GCP-OPT is used, but note that this choice is not the GCP-OPT default for sparse tensors. All algorithms under test are stopped when the relative change in the -divergence (in this case, $= 1$) is less than 10^{-8}.

Fig. 14 shows the cost change against time of the algorithms under $R = 5$ and $R = 10$, respectively. Each algorithm is run for 20 trials and in each trial, the factor matrices are initialized by randomly sampling its entries from uniform distribution between 0 and 1. One can see that the proposed algorithm `SmartCPD` exhibits a fast runtime performance in this case. For each of the cases, the `SmartCPD` takes only about 5 seconds to reach a low cost value, whereas the baselines take much more time but still could not attain the same cost value. In some trials, especially when $R = 5$, there exist some cases where `GGN` did not converge. More details on the algorithm-output latent factors can be found in Fig. 13.

2) Plant-Pollinator Network Data: We also consider the plant-pollinator network dataset published by [59]. The dataset consists of plant-pollinator interactions collected over 12 meadows in the Oregon Cascade Mountains, USA; also see [12] for detailed data descriptions. We extract the number of interactions between 562 pollinator species and 124 plant species over 123 days during 2011 and 2015. Hence, we form a count-type

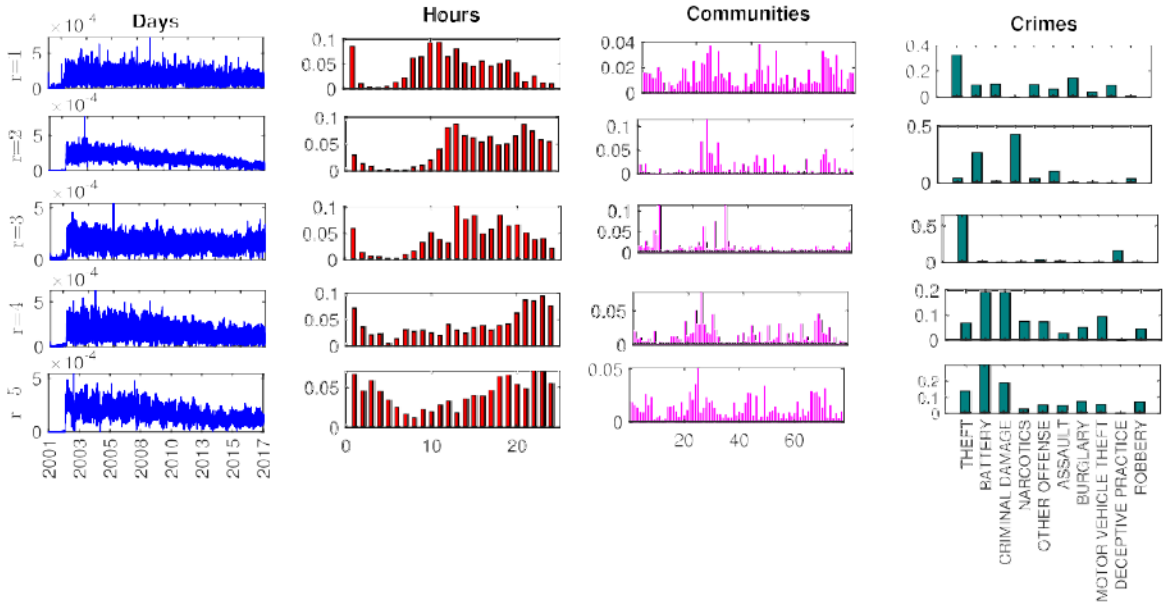
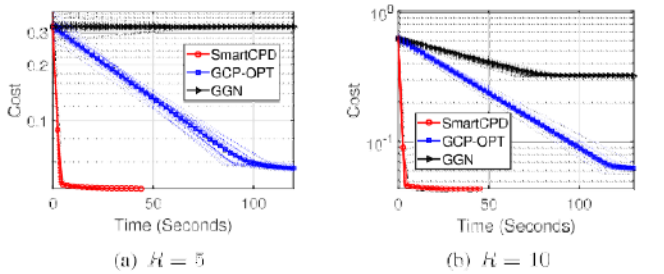
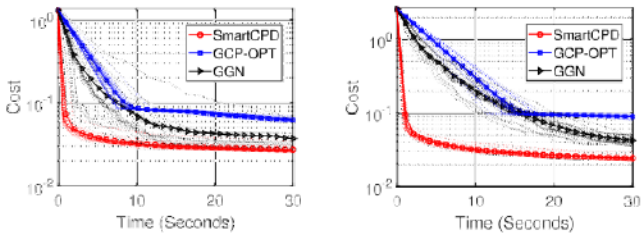


Fig. 13. Learned latent factors by SmartCPD when it reached the stopping criterion with cost value 0.043 (time= 35.81 sec.). Out of the 32 crimes reported, we chose the most frequent 10 crimes for visualization.



(a) $R = 5$ (b) $R = 10$

Fig. 14. Convergence of the algorithms (Count data, Chicago Crime, KL div., size = 6186 × 24 × 77 × 32).

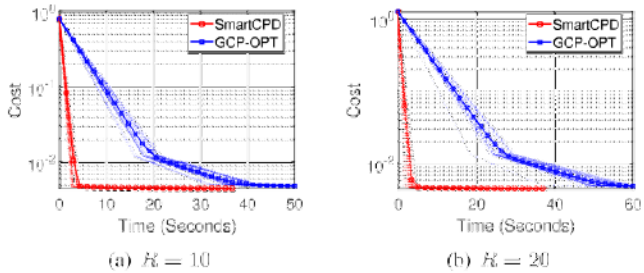


(a) $R = 10$ (b) $R = 20$

Fig. 15. Convergence of the algorithms (Count data, Plant-Pollinator network, gen. KL div., size = 562 × 124 × 123).

third-order tensor with a size of 562 × 124 × 123, where each entry represents the number of interactions between a particular plant and a pollinator on a specific day. The data is fairly sparse with only 8,370 (≈0.1%) nonzero entries. We choose the loss function corresponding to the Poisson distribution, i.e., $f(x, m) = m - x \log(m + c)$ along with non-negativity constraints. Other algorithm settings are also same as those used in the Chicago Crime data.

Fig. 15 plots the cost values of the algorithms against time, for different values of R. Both the baselines GCP-OPT and GGN



(a) $R = 10$ (b) $R = 20$

Fig. 16. Convergence of the algorithms (binary data, UCI chat network, log loss, size = 400 × 400 × 196).

work reasonably well in this dataset. However, the proposed SmartCPD has better performance—it converges to lower cost values compared to the other baselines. Also, one can observe that the SmartCPD is about 15 times faster in reaching low cost values compared to GCP-OPT and GGN.

3) UCI Chat Network Data: We test the algorithms on real-world binary data using a social network which contains the online interactions of the students from the University of California, Irvine, USA. The original dataset was published by [60], which includes 59,835 online messages sent between 1,899 students over 196 days from March 2004 to October 2004. We select 400 most prolific senders and form a third-order binary tensor of size 400 × 400 × 196 having 18862 (0.06%) nonzero entries. Each entry of the binary tensor indicates if sender i has sent a message to receiver j on the kth day. We choose the loss function corresponding to the Bernoulli distribution, i.e., $f(m, x) = \log(m + 1) - x \log(m + c)$ (see Table II). Other settings and parameters are as before. Since GGN method is not designed for the loss function considered in this case, it is not included.

Fig. 16 shows the cost value change against time in seconds for different values of R. Similar to the previous datasets, the proposed `SmartCPD` shows considerable runtime advantages over `GCP-OPT`. In all the trials, `SmartCPD` is at least 40 times faster for converging to a cost value that is later attained by `GCP-OPT`.

VI. Conclusion

In this work, we proposed a unified SMD algorithmic framework for low-rank CPD under non-Euclidean losses. By integrating a fiber-sampling strategy within the SMD optimization technique, the proposed framework is flexible in dealing with a variety of loss functions and constraints that are of interest in real-world data analytics. By its stochastic nature, the proposed algorithm enjoys low computational and memory costs. In addition, under different data types and loss functions, we discussed a number of "best practices," e.g., step size scheduling and local surrogate function construction, which were shown critical for effective implementation. We also provided rigorous convergence analysis that is tailored for the non-Euclidean CPD, since generic SMD proofs do not cover the proposed algorithm. We tested the algorithm over various types of simulated and real data. Substantial computational savings relative to state-of-the-art methods were observed. These results show encouraging and promising performance of using geometry-aware algorithm design for large-scale tensor decomposition.

References

[1] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3551–3582, Jul. 2017.

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

[3] A. Cichocki *et al.*, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE Signal Process. Mag.*, vol. 32, no. 2, pp. 145–163, Mar. 2015.

[4] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky, "Tensor decompositions for learning latent variable models," *J. Mach. Learn. Res.*, vol. 15, pp. 2773–2832, 2014.

[5] X. Fu, N. Vervliet, L. De Lathauwer, K. Huang, and N. Gillis, "Computing large-scale matrix and tensor decomposition with structured factors: A unified nonconvex optimization perspective," *IEEE Signal Process. Mag.*, vol. 37, no. 5, pp. 78–94, Sep. 2020.

[6] D. Hong, T. G. Kolda, and J. A. Duersch, "Generalized canonical polyadic tensor decomposition," *SIAM Rev.*, vol. 62, no. 1, pp. 133–163, 2020.

[7] T. G. Kolda and D. Hong, "Stochastic gradients for large-scale tensor decomposition," *SIAM J. Math. Data Sci.*, vol. 2, no. 4, pp. 1066–1095, Oct. 2020.

[8] M. Wang and L. Li, "Learning from binary multiway data: Probabilistic tensor decomposition and its statistical optimality," *J. Mach. Learn. Res.*, vol. 21, no. 154, pp. 1–38, 2020.

[9] M. Vandecappelle, N. Vervliet, and L. De Lathauwer, "A second-order method for fitting the canonical polyadic decomposition with non-least-squares cost," *IEEE Trans. Signal Process.*, vol. 68, pp. 4454–4465, Jul. 2020, doi: 10.1109/TSP.2020.3010719.

[10] K. Huang and N. D. Sidiropoulos, "Kullback-Leibler principal component for tensors is not NP-hard," in *Proc. IEEE 51st Asilomar Conf. Signals, Syst, Comput.*, 2017, pp. 693–697.

[11] N. Kargas and N. D. Sidiropoulos, "Learning mixtures of smooth product distributions: Identifiability and algorithm," in *Proc. 22nd Int. Conf. Artif. Intell. Statist.*, 2019, pp. 388–396.

[12] X. Fu, E. Seo, J. Clarke, and R. A. Hutchinson, "Link prediction under imperfect detection: Collaborative filtering for ecological networks," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 8, pp. 3117–3128, Aug. 2021.

[13] L. Cheng, X. Tong, S. Wang, Y.-C. Wu, and H. V. Poor, "Learning nonnegative factors from tensor data: Probabilistic modeling and inference algorithm," *IEEE Trans. Signal Process.*, vol. 68, pp. 1792–1806, Feb. 2020, doi: 10.1109/TSP.2020.2975353.

[14] B. Ermiş,, E. Acar, and A. T. Cemgil, "Link prediction in heterogeneous data via generalized coupled tensor factorization," *Data Mining Knowl. Discov.*, vol. 29, no. 1, pp. 203–236, 2015.

[15] E. C. Chi and T. G. Kolda, "On tensors, sparsity, and nonnegative factorizations," *SIAM J. Matrix Anal. Appl.*, vol. 33, no. 4, pp. 1272–1299, 2012.

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

[17] A.-H. Phan, P. Tichavský, and A. Cichocki, "Fast alternating LS algorithms for high order candecomp/parafac tensor factorizations," *IEEE Trans. Signal Process.*, vol. 61, no. 19, pp. 4834–4846, Oct. 2013.

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

[19] 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, Oct. 2016.

[20] L. Sorber, M. Van Barel, and L. De Lathauwer, "Optimization-based algorithms for tensor decompositions: Canonical polyadic decomposition, decomposition in rank-$(l_r, l_r, 1)$ terms, and a new generalization," *SIAM J. Optim.*, vol. 23, no. 2, pp. 695–720, 2013.

[21] A.-H. Phan, P. Tichavsky, and A. Cichocki, "Low complexity damped gauss-newton algorithms for candecomp/parafac," *SIAM J. Matrix Anal. Appl.*, vol. 34, no. 1, pp. 126–147, 2013.

[22] 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 Int. Conf. Data Mining*, 2014, pp. 109–117.

[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, Mar. 2016.

[24] M. Sorensen and L. De Lathauwer, "Fiber sampling approach to canonical polyadic decomposition and application to tensor completion," *SIAM J. Matrix Anal. Appl.*, vol. 40, no. 3, pp. 888–917, 2019.

[25] X. Fu, S. Ibrahim, H.-T. Wai, C. Gao, and K. Huang, "Block-randomized stochastic proximal gradient for low-rank tensor factorization," *IEEE Trans. Signal Process.*, vol. 68, pp. 2170–2185, Mar. 2020, doi: 10.1109/TSP.2020.2982321.

[26] A. Cichocki and A.-H. Phan, "Fast local algorithms for large scale nonnegative matrix and tensor factorizations," *IEICE Trans. Fundamentals Electron., Commun. Comput. Sci.*, vol. 92, no. 3, pp. 708–721, 2009.

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

[28] C. Févotte and J. Idier, "Algorithms for nonnegative matrix factorization with the □divergence," *Neural Computation*, vol. 23, no. 9, pp. 2421–2456, 2011.

[29] A. Beck and M. Teboulle, "Mirror descent and nonlinear projected subgradient methods for convex optimization," *Operations Res. Lett.*, vol. 31, no. 3, pp. 167–175, 2003.

[30] C. Battaglino, G. Ballard, and T. G. Kolda, "A practical randomized cp tensor decomposition," *SIAM J. Matrix Anal. Appl.*, vol. 39, no. 2, pp. 876–901, 2018.

[31] C. D. Dang and G. Lan, "Stochastic block mirror descent methods for nonsmooth and stochastic optimization," *SIAM J. Optim.*, vol. 25, no. 2, pp. 856–881, 2015.

[32] W. Pu, S. Ibrahim, X. Fu, and M. Hong, "Fiber-sampled stochastic mirror descent for tensor decomposition with □divergence," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2021, pp. 2925–2929, doi: 10.1109/ICASSP39728.2021.9413830.

[33] J. Shetty and J. Adibi, "The enron email dataset database schema and brief statistical report," Inf. Sci. Inst. Tech. Rep., Univ. Southern California, vol. 4, no. 1, pp. 120–128, 2004.

[34] C. Févotte, N. Bertin, and J.-L. Durrieu, "Nonnegative matrix factorization with the Itakura-Saito divergence: With application to music analysis," *Neural Computation*, vol. 21, no. 3, pp. 793–830, 2009.

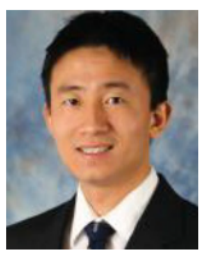
[35] K. Huang and X. Fu, "Detecting overlapping and correlated communities without pure nodes: Identifiability and algorithm," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2859–2868.
[36] H. H. Bauschke, J. Bolte, and M. Teboulle, "A descent lemma beyond Lipschitz gradient continuity: First-order methods revisited and applications," *Math. Operations Res.*, vol. 42, no. 2, pp. 330–348, 2017.
[37] J. Bolte, S. Sabach, M. Teboulle, and Y. Vaisbourd, "First order methods beyond convexity and Lipschitz gradient continuity with applications to quadratic inverse problems," *SIAM J. Optim.*, vol. 28, no. 3, pp. 2131–2151, 2018.
[38] H. Lu, "Relative continuity for non-Lipschitz nonsmooth convex optimization using stochastic (or deterministic) mirror descent," *INFORMS J. Optim.*, vol. 1, no. 4, pp. 288–303, 2019.
[39] D. Davis, D. Drusvyatskiy, and K. J. MacPhee, "Stochastic model-based minimization under high-order growth," 2018, *arXiv:1807.00255*.
[40] S. Zhang and N. He, "On the convergence rate of stochastic mirror descent for nonsmooth nonconvex optimization," 2018, *arXiv:1806.04781*.
[41] Q. Li, Z. Zhu, G. Tang, and M. B. Wakin, "Provable Bregman-Divergence based methods for nonconvex and non-Lipschitz problems," 2019, *arXiv:1904.09712*.
[42] L. M. Bregman, "The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming," *USSR Comput. Math. Math. Phys.*, vol. 7, no. 3, pp. 200–217, 1967.
[43] A. Yeredor and M. Haardt, "Estimation of a low-rank probability-tensor from sample sub-tensors via joint factorization minimizing the Kullback-Leibler divergence," in *Proc. IEEE 27th Eur. Signal Process. Conf.*, 2019, pp. 1–5.
[44] A. Yeredor and M. Haardt, "Maximum likelihood estimation of a low-rank probability mass tensor from partial observations," *IEEE Signal Process. Lett.*, vol. 26, no. 10, pp. 1551–1555, Oct. 2019.
[45] S. Ibrahim and X. Fu, "Recovering joint probability of discrete random variables from pairwise marginals," *IEEE Trans. Signal Process.*, vol. 69, pp. 4116–4131, Jun. 2021, doi: 10.1109/TSP.2021.3090960.
[46] L. T. K. Hien and N. Gillis, "Algorithms for nonnegative matrix factorization with the kullback-leibler divergence," 2020, *arXiv:2010.01935*.
[47] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations Trends Optim.*, vol. 1, no. 3, pp. 127–239, 2014.
[48] Y. Sun, P. Babu, and D. P. Palomar, "Majorization-minimization algorithms in signal processing, communications, and machine learning," *IEEE Trans. Signal Process.*, vol. 65, no. 3, pp. 794–816, Feb. 2017.
[49] M. Hong, M. Razaviyayn, Z.-Q. Luo, and J.-S. Pang, "A unified algorithmic framework for block-structured optimization involving Big Data: With applications in machine learning and signal processing," *IEEE Signal Process. Mag.*, vol. 33, no. 1, pp. 57–77, Jan. 2016.
[50] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, no. 61, pp. 2121–2159, 2011. [Online]. Available: http://jmlr.org/papers/v12/duchi11a.html
[51] X. Chen, S. Liu, R. Sun, and M. Hong, "On the convergence of a class of adam-type algorithms for non-convex optimization," in *Proc. Int. Conf. Learn. Representations*, 2019.
[52] H. Lu, R. M. Freund, and Y. Nesterov, "Relatively smooth convex optimization by first-order methods, and applications," *SIAM J. Optim.*, vol. 28, no. 1, pp. 333–354, 2018.
[53] M. Teboulle, "A simplified view of first order methods for optimization," *Math. Program.*, vol. 170, no. 1, pp. 67–96, 2018.
[54] T. Kola *et al.*, "Tensor toolbox for MATLAB v. 3.0," Sandia National Lab. (SNL-NM), Albuquerque, NM (United States), Oct. 2017. [Online]. Available: https://www.tensortoolbox.org
[55] N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer, "Tensorlab 3.0," Mar. 2016. [Online]. Available: www.tensorlab.net
[56] N. Kargas, N. D. Sidiropoulos, and X. Fu, "Tensors, learning, and 'Kolmogorov extension' for finite-alphabet random vectors," *IEEE Trans. Signal Process.*, vol. 66, no. 18, pp. 4854–4868, Sep. 2018.
[57] S. Ibrahim, X. Fu, N. Kargas, and K. Huang, "Crowdsourcing via pairwise co-occurrences: Identifiability and algorithms," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 7847–7857.
[58] S. Smith *et al.*, "FROSTT: The formidable repository of open sparse tensors and tools," 2017. [Online]. Available: http://frostt.io/
[59] J. A. Jones, R. A. Hutchinson, and V. W. Pfeiffer, "Plant pollinator data at hj andrews experimental forest, 2011 to 2015," *Environmental Data Initiative, ver. 5*.
[60] T. Opsahl and P. Panzarasa, "Clustering in weighted networks," *Social Netw.*, vol. 31, no. 2, pp. 155–163, 2009.

Wenqiang Pu (Member, IEEE) received the B.S. and Ph.D. degrees in electrical engineering from Xidian University, Xi'an, China, in 2013 and 2018, respectively. From January 2019 to October 2020, he was a Postdoc Researcher with the School of Science and Engineering, The Chinese University of Hong Kong (Shenzhen), Shenzhen, China. He is currently a Research Scientist with the Shenzhen Research Institute of Big Data, Shenzhen, China. His research interests include signal processing and optimization algorithms.

Shahana Ibrahim received the B.Tech. degree in electronics and communication engineering from the National Institute of Technology, Calicut, Kozhikode, India, in 2012, and the M.S. degree in electrical engineering, in 2019, from Oregon State University, Corvallis, OR, USA, where she is currently working toward the Ph.D. degree. From 2012 to 2017, she was a System Validation Engineer with Texas Instruments, Bengaluru, India. Her research interests include statistical machine learning and signal processing.

Xiao Fu (Senior Member, IEEE) received the B.Eng. and M.Sc. degrees from the University of Electronic Science and Technology of China, Chengdu, China, in 2005 and 2010, respectively, and the Ph.D. degree in electronic engineering from The Chinese University of Hong Kong, Hong Kong, in 2014. From 2014 to 2017, he was a Postdoctoral Associate with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN, USA. Since 2017, he has been an Assistant Professor with the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR, USA. His research interests include the broad area of signal processing and machine learning.

Dr. Fu was the recipient of the Best Student Paper Award at ICASSP 2014, and Outstanding Postdoctoral Scholar Award at the University of Minnesota in 2016. His coauthored papers was the recipient of the Best Student Paper Awards from IEEE CAMSAP 2015 and IEEE MLSP 2019, respectively. He is a Member of the Sensor Array and Multichannel Technical Committee of the IEEE Signal Processing Society. He is also a Member of the Signal Processing for Multisensor Systems Technical Area Committee of EURASIP. He is the Treasurer of the IEEE SPS Oregon Chapter. He is the Editor of *Signal Processing* and is also an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING. He was a Tutorial Speaker at ICASSP 2017 and SIAM Conference on Applied Linear Algebra 2021.



Mingyi Hong (Senior Member, IEEE) received the Ph.D. degree from the University of Virginia, Charlottesville, VA, USA, in 2011. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN, USA. His research interests include optimization theory and applications in signal processing and machine learning. He is on the IEEE Signal Processing for Communications and Networking Technical Committee. He is also an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING.