4-2019

Faster first-order methods for stochastic non-convex optimization on Riemannian manifolds

Pan ZHOU
Singapore Management University, panzhou@smu.edu.sg

Xiao-Tong YUAN

Jiashi FENG

Faster First-Order Methods for Stochastic Non-Convex Optimization on Riemannian Manifolds

Pan Zhou[*]
[*]National University of Singapore
pzhou@u.nus.edu

Xiao–Tong Yuan[†]
[†]Nanjing University of Information Science & Technology
xtyuan@nuist.edu.cn

Jiashi Feng[*]
elefjia@nus.edu.sg

Abstract

SPIDER (Stochastic Path Integrated Differential EstimatoR) is an efficient gradient estimation technique developed for non-convex stochastic optimization. Although having been shown to attain nearly optimal computational complexity bounds, the SPIDER-type methods are limited to linear metric spaces. In this paper, we introduce the Riemannian SPIDER (R-SPIDER) method as a novel nonlinear-metric extension of SPIDER for efficient non-convex optimization on Riemannian manifolds. We prove that for finite-sum problems with n components, R-SPIDER converges to an ϵ-accuracy stationary point within $\mathcal{O}\big(\min\big(n+\frac{\sqrt{n}}{\epsilon^2},\frac{1}{\epsilon^3}\big)\big)$ stochastic gradient evaluations, which is sharper in magnitude than the prior Riemannian first-order methods. For online optimization, R-SPIDER is shown to converge with $\mathcal{O}\big(\frac{1}{\epsilon^3}\big)$ complexity which is, to the best of our knowledge, the first non-asymptotic result for online Riemannian optimization. Especially, for gradient dominated functions, we further develop a variant of R-SPIDER and prove its linear convergence rate. Numerical results demonstrate the computational efficiency of the proposed methods.

1 Introduction

We consider the following *finite-sum* and *online* non-convex problems on a Riemannian manifold $\mathcal{M}$:

$$\min_{\boldsymbol{x}\in\mathcal{M}} f(\boldsymbol{x}) := \begin{cases} \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x}) & \text{(finite-sum)} \\ \mathbb{E}[f(\boldsymbol{x};\pi)] & \text{(online)} \end{cases}, \tag{1}$$

where $f:\mathcal{M}\mapsto\mathbb{R}$ is a smooth non-convex loss function. For the finite-sum problem, each individual loss $f_i(\boldsymbol{x})$ is associated with the i-th sample, while in online setting, the stochastic component $f(\boldsymbol{x};\pi)$ is indexed by a random variable π. Such a formulation encapsulates several important finite-sum problems and their corresponding online counterparts, including principle component analysis (PCA) [1], low-rank matrix/tensor completion/recovery [2, 3, 4, 5], dictionary learning [6, 7], Gaussian mixture models [8] and low-rank multivariate regression [9], to name a few.

One classic approach for solving problem (1) (or its convex counterpart) is to take it as a constrained optimization problem in ambient Euclidean space and find the minimizers via projected (stochastic) gradient descent [12, 13, 14]. This kind of methods, however, tend to suffer from high computational cost as projection onto certain manifolds (e.g., positive-definite matrices) could be expensive in large-scale learning problems [11].

As an appealing alternative, the Riemannian optimization methods have recently gained wide attention in machine learning [10, 11, 15, 16, 17, 18, 19]. In contrast to the Euclidean-projection based methods, the Riemannian methods directly move the iteration along the geodesic path towards the optimal solution, and thus can better respect the geometric structure of the problem in hand. Specifically, the Riemannian gradient methods have the following recursive form:

$$\boldsymbol{x}_{k+1} = \mathrm{Exp}_{\boldsymbol{x}_k}\left(-\eta_k \boldsymbol{g}_k\right), \tag{2}$$

where $\boldsymbol{g}_k$ is the gradient estimate of the full Riemannian gradient $\nabla f(\boldsymbol{x}_k)$, η_k denotes the learning rate, and the exponential mapping $\mathrm{Exp}_{\boldsymbol{x}}(\boldsymbol{y})$, as defined in Section 2, maps $\boldsymbol{y}$ in the tangent space at $\boldsymbol{x}$ to $\mathrm{Exp}_{\boldsymbol{x}}(\boldsymbol{y})$ on the manifold $\mathcal{M}$ along a proper geodesic curve. For instance, Riemannian gradient descent (R-GD) uses the full Riemannian gradient $\boldsymbol{g}_k=\nabla f(\boldsymbol{x}_k)$ in Eqn. (2) and has been shown to have sublinear rate of convergence in geodesically convex problems [16]. To boost efficiency, Liu *et al.* [20] and Zhang *et al.* [15] further introduced the Nesterov acceleration techniques [21] into R-GD with convergence rate significantly improved for geodesically convex functions.

Table 1: Comparison of IFO complexity for different Riemannian first-order stochastic optimization algorithms on the noncovnex problem (1) under finite-sum and online settings. The ϵ-accuracy solution is measured by the expected gradient norm $\mathbb{E}\left[\|\nabla f(\boldsymbol{x})\|\right] \leq \epsilon$. Here L, σ and ζ respectively denote the gradient Lipschitz constant, the gradient variance and the curvature parameter of the Riemannian manifold (see Section 2).

		Non-convex Problem	
		general non-convex	τ-gradient dominated
Finite-sum	R-SRG [10]	$\mathcal{O}\left(n+\frac{L^2}{\epsilon^4}\right)$	$\mathcal{O}\left((n+\tau^2L^2)\log\left(\frac{1}{\epsilon}\right)\right)$
	R-SVRG [11]	$\mathcal{O}\left(n+\frac{\zeta n^{\frac{2}{3}}}{\epsilon^2}\right)$	$\mathcal{O}\left((n+\tau L\zeta^{\frac{1}{2}}n^{\frac{2}{3}})\log\left(\frac{1}{\epsilon}\right)\right)$
	this work	$\mathcal{O}\left(\min\left(n+\frac{L\sqrt{n}}{\epsilon^2},\frac{L\sigma}{\epsilon^3}\right)\right)$	$\mathcal{O}\left(\min\left((n+\tau L\sqrt{n})\log\left(\frac{1}{\epsilon}\right),\frac{\tau L\sigma}{\epsilon}\right)\right)$
Online	this work	$\mathcal{O}\left(\frac{L\sigma}{\epsilon^3}\right)$	$\mathcal{O}\left(\frac{\tau L\sigma}{\epsilon}\right)$

To avoid the time-consuming full gradient computation required in R-GD, Riemannian stochastic optimization algorithms [10, 11, 17, 18, 19] leverage the decomposable (finite-sum) structure of problem (1). For instance, Bonnabel *et al.* [17] proposed R-SGD that only evaluates gradient of one (or a mini-batch) randomly selected sample for variable update per iteration. Though with good iteration efficiency, R-SGD converges slowly as it uses decaying learning rate for convergence guarantee due to its gradient variance. To tackle this issue, Riemannian stochastic variance-reduced gradient (R-SVRG) algorithms [11, 19] adapt SVRG [22] to problem (1). Benefiting from the variance-reduced technique, R-SVRG converges more stably and efficiently than R-SGD. More recently, inspired by the variance-reduced stochastic recursive gradient approach [23, 24], the Riemannian stochastic recursive gradient (R-SRG) algorithm [10] establishes a recursive equation to estimate the full Riemannian gradient so that the computational efficiency can be further improved (see Table 1).

SPIDER (Stochastic Path Integrated Differential EstimatoR) [25] is a recursive estimation method developed for tracking the history full gradients with significantly reduced computational cost. By combining SPIDER with normalized gradient methods, nearly optimal iteration complexity bounds can be attained for non-convex optimization in Euclidean space [25]. Though appealing in vector space problems, it has not been explored for non-convex optimization in nonlinear metric spaces such as Riemannian manifold.

In this paper, we introduce the *Riemannian Stochastic Path Integrated Differential EstimatoR* (R-SPIDER) as a simple yet efficient extension of the SPIDER from Euclidean space to Riemannian manifolds. Specifically, for a proper positive integer p, at each time instance k with mod $(k,p)\equiv 0$, R-SPIDER first samples a large data batch $\mathcal{S}_1$ and estimates the initial full Riemannian gradient $\nabla f(\boldsymbol{x}_k)$ as $\boldsymbol{v}_k = \nabla f_{\mathcal{S}_1}(\boldsymbol{x}_k) = \frac{1}{|\mathcal{S}_1|}\sum_{i\in\mathcal{S}_1} f_i(\boldsymbol{x}_k)$. Then at each of the next $p-1$ iterations, it samples a smaller mini-batch $\mathcal{S}_2$ and estimates/tracks $\nabla f(\boldsymbol{x}_k)$:

$$\boldsymbol{v}_k = \nabla f_{\mathcal{S}_2}(\boldsymbol{x}_k) - \mathrm{P}_{\boldsymbol{x}_{k-1}}^{\boldsymbol{x}_k}\left(\nabla f_{\mathcal{S}_2}(\boldsymbol{x}_{k-1}) - \boldsymbol{v}_{k-1}\right), \quad (3)$$

where the parallel transport $\mathrm{P}_{\boldsymbol{x}}^{\boldsymbol{z}}(\boldsymbol{y})$ (as defined in Section 2) transports $\boldsymbol{y}$ from the tangent space at $\boldsymbol{x}$ to that at the point $\boldsymbol{z}$. Here the parallel transport operation is necessary since $\nabla f_{\mathcal{S}_2}(\boldsymbol{x}_{k-1})$ and $\nabla f_{\mathcal{S}_2}(\boldsymbol{x}_k)$ are located in different tangent spaces. Given the gradient estimate $\boldsymbol{v}_k$, the variable is updated via normalized gradient descent $\boldsymbol{x}_{k+1} = \mathrm{Exp}_{\boldsymbol{x}_k}\big(-\eta_k\frac{\boldsymbol{v}_k}{\|\boldsymbol{v}_k\|}\big)$. Note that R-SRG [10] applies a similar recursion form as in (3) for full Riemannian gradient estimation, and the core difference between their method and ours lies in that R-SPIDER is equipped with gradient normalization which is missing in R-SRG. Then by carefully setting the learning rate η and mini-batch sizes of $\mathcal{S}_1$ and $\mathcal{S}_2$, R-SPIDER only needs to sample a necessary number of data points for accurately estimating Riemannian gradient and sufficiently decreasing the objective at each iteration. In this way, R-SPIDER achieves sharper bounds of incremental first order oracle (IFO, see Definition 2) complexity than R-SRG and other state-of-the-art Riemannian non-convex optimization methods.

Table 1 summarizes our main results on the computational complexity of R-SPIDER for non-convex problems, along with those for the above mentioned Riemannian gradient algorithms. The following are some highlighted advantages of our results over the state-of-the-arts.

For the finite-sum setting of problem (1) with general non-convex functions, the IFO complexity of R-SPIDER to achieve $\mathbb{E}\left[\|\nabla f(\boldsymbol{x})\|\right] \leq \epsilon$ is $\mathcal{O}\big(\min\big(n+\frac{L\sqrt{n}}{\epsilon^2},\frac{L\sigma}{\epsilon^3}\big)\big)$ which matches the lower IFO complexity bound in Euclidean space [25]. By comparison, the IFO complexity bounds of R-SRG and R-SVRG are $\mathcal{O}\big(n+\frac{L^2}{\epsilon^4}\big)$ and $\mathcal{O}\big(n+\frac{\zeta n^{\frac{2}{3}}}{\epsilon^2}\big)$, respectively. It can be verified that R-SPIDER improves over R-SRG by a factor of $\mathcal{O}\big(\frac{1}{\epsilon}\big)$ and R-SVRG by a factor $\mathcal{O}\big(n^{1/6}\big)$ regardless

of the relation between n and ϵ.

When $f(\boldsymbol{x})$ is a τ-gradient dominated function with finite-sum structure, R-SPIDER enjoys the IFO complexity of $\mathcal{O}\big(\min\big(\big(n+\tau L\sqrt{n}\big)\log\big(\frac{1}{\epsilon}\big), \frac{\tau L\sigma}{\epsilon}\big)\big)$ which is again lower than the $\mathcal{O}\big(\big(n+\tau L\zeta^{\frac{1}{2}}n^{\frac{2}{3}}\big)\log\big(\frac{1}{\epsilon}\big)\big)$ bound for R-SVRG by a factor of $\mathcal{O}\left(n^{1/6}\right)$. Note that our IFO complexity is not dependent on the curvature parameter $\zeta(\geq 1)$ of the manifold $\mathcal{M}$, because our analysis does not involve the geodesic trigonometry inequality on a manifold. To compare with R-SRG with complexity bound $\mathcal{O}\big(\big(n+\tau^2L^2\big)\log\big(\frac{1}{\epsilon}\big)\big)$, R-SPIDER is more efficient than R-SRG in large-sample-moderate-accuracy settings, e.g., in cases when n dominates $1/\epsilon$.

For the online version of problem (1), we establish the IFO complexity bounds $\mathcal{O}\big(\frac{L\sigma}{\epsilon^3}\big)$ and $\mathcal{O}\left(\frac{\tau L\sigma}{\epsilon}\right)$ for generic non-convex and gradient dominated problems, respectively. To our best knowledge, these non-asymptotic convergence results are novel to non-convex online Riemannian optimization. Comparatively, Bonnabel *et al.* [17] only provided asymptotic convergence analysis of R-SGD: the iterating sequence generated by R-SGD converges to a critical point when the iteration number approaches infinity.

Finally, our analysis reveals as a byproduct that R-SPIDER provably benefits from mini-batching. Specifically, our theoretic results imply linear speedups in parallel computing setting for large mini-batch sizes. We are not aware of any similar linear speedup results in the prior Riemannian stochastic algorithms.

2 Preliminaries

Throughout this paper, we assume that the Riemannian manifold $(\mathcal{M}, \mathfrak{g})$ is a real smooth manifold $\mathcal{M}$ equipped with a Riemannian metric $\mathfrak{g}$. We denote the induced inner product $\langle \boldsymbol{y}, \boldsymbol{z}\rangle$ of any two vectors $\boldsymbol{y}$ and $\boldsymbol{z}$ in the tangent space $\mathrm{T}_{\boldsymbol{x}}\mathcal{M}$ at the point $\boldsymbol{x}$ as $\langle \boldsymbol{y}, \boldsymbol{z}\rangle = \mathfrak{g}(\boldsymbol{y}, \boldsymbol{z})$, and denote the norm $\|\boldsymbol{y}\|$ as $\|\boldsymbol{y}\| = \sqrt{\mathfrak{g}(\boldsymbol{y}, \boldsymbol{y})}$. Let $\nabla f_i(\boldsymbol{x})$ be the stochastic Riemannian gradient of $f_i(\boldsymbol{x})$ and also be a unbiased estimate to the full Riemannian gradient $\nabla f(\boldsymbol{x})$, i.e. $\mathbb{E}_i[\nabla f_i(\boldsymbol{x})] = \nabla f(\boldsymbol{x})$.

The exponential mapping $\mathrm{Exp}_{\boldsymbol{x}}(\boldsymbol{y})$ maps $\boldsymbol{y} \in \mathrm{T}_{\boldsymbol{x}}\mathcal{M}$ to $\boldsymbol{z} \in \mathcal{M}$ such that there is a geodesic $\gamma(t)$ with $\gamma(0) = \boldsymbol{x}$, $\gamma(1) = \boldsymbol{z}$ and $\dot{\gamma}(0) = \frac{d}{dt}\gamma(t) = \boldsymbol{y}$. Here the geodesic $\gamma(t)$ is a constant speed curve $\gamma : [0, 1] \to \mathcal{M}$ which is locally distance minimized. If there exists a unique geodesic between any two points on $\mathcal{M}$, then the exponential map has an inverse mapping $\mathrm{Exp}_{\boldsymbol{x}}^{-1} : \mathcal{M} \to \mathrm{T}_{\boldsymbol{x}}\mathcal{M}$ and the geodesic is the unique shortest path with the geodesic distance $\mathrm{d}(\boldsymbol{x}, \boldsymbol{z}) = \|\mathrm{Exp}_{\boldsymbol{x}}^{-1}(\boldsymbol{z})\| = \|\mathrm{Exp}_{\boldsymbol{z}}^{-1}(\boldsymbol{x})\|$ between $\boldsymbol{x}, \boldsymbol{z} \in \mathcal{M}$.

To utilize the historical and current Riemannian gradients, we need to transport the historical gradients into the tangent space of the current point such that these gradients can be linearly combined in one tangent space. For this purpose, we need to define the parallel transport operator $\mathrm{P}_{\boldsymbol{x}}^{\boldsymbol{z}} : \mathrm{T}_{\boldsymbol{x}}\mathcal{M} \to \mathrm{T}_{\boldsymbol{z}}\mathcal{M}$ which maps $\boldsymbol{y} \in \mathrm{T}_{\boldsymbol{x}}\mathcal{M}$ to $\mathrm{P}_{\boldsymbol{x}}^{\boldsymbol{z}}(\boldsymbol{y}) \in \mathrm{T}_{\boldsymbol{z}}\mathcal{M}$ while preserving the inner product and norm, i.e., $\langle \boldsymbol{y}_1, \boldsymbol{y}_2\rangle = \langle \mathrm{P}_{\boldsymbol{x}}^{\boldsymbol{z}}(\boldsymbol{y}_1), \mathrm{P}_{\boldsymbol{x}}^{\boldsymbol{z}}(\boldsymbol{y}_2)\rangle$ and $\|\boldsymbol{y}\| = \|\mathrm{P}_{\boldsymbol{x}}^{\boldsymbol{z}}(\boldsymbol{y})\|$ for $\forall \boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y} \in \mathrm{T}_{\boldsymbol{x}}\mathcal{M}$.

We impose on the loss components $f_i(\boldsymbol{x})$ the assumption of geodesic gradient-Lipschitz-smoothness. Such a smoothness condition is conventionally assumed in analyzing Riemannian gradient algorithms [10, 11, 26, 27].

Assumption 1 (Geodesically L-gradient-Lipschitz)**.** *Each loss $f_i(\boldsymbol{x})$ is geodesically L-gradient Lipschitz such that $\mathbb{E}_i\|\nabla f_i(\boldsymbol{x}) - P_{\boldsymbol{y}}^{\boldsymbol{x}}(\nabla f_i(\boldsymbol{y}))\|^2 \leq L^2\|Exp_{\boldsymbol{x}}^{-1}(\boldsymbol{y})\|^2$.*

It can be shown that if each $f_i(\boldsymbol{x})$ is geodesically L-gradient-Lipschitz, then for any $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{M}$,

$$f(\boldsymbol{y}) \leq f(\boldsymbol{x}) + \left\langle \nabla f(\boldsymbol{x}), \mathrm{Exp}_{\boldsymbol{x}}^{-1}(\boldsymbol{y})\right\rangle + \frac{L}{2}\|\mathrm{Exp}_{\boldsymbol{x}}^{-1}(\boldsymbol{y})\|^2.$$

We also need to impose the following boundness assumption on the variance of stochastic gradient.

Assumption 2 (Bounded Stochastic Gradient Variance)**.** *For any $\boldsymbol{x} \in \mathcal{M}$, the gradient variance of each loss $f_i(\boldsymbol{x})$ is bounded as $\mathbb{E}_i\|\nabla f_i(\boldsymbol{x}) - \nabla f(\boldsymbol{x})\|^2 \leq \sigma^2$.*

We further introduce the following concept of τ-gradient dominated function [28, 29] which will also be investigated in this paper.

Definition 1 (τ-Gradient Dominated Functions)**.** *$f(\boldsymbol{x})$ is said to be a τ-gradient dominated function if it satisfies $f(\boldsymbol{x}) - f(\boldsymbol{x}_*) \leq \tau\|\nabla f(\boldsymbol{x})\|^2$ for any $\boldsymbol{x} \in \mathcal{M}$, where τ is a universal constant and $\boldsymbol{x}_* = \mathrm{argmin}_{\boldsymbol{x}\in\mathcal{M}} f(\boldsymbol{x})$ is the global minimizer of $f(\boldsymbol{x})$ on the manifold $\mathcal{M}$.*

The following defined incremental first order oracle (IFO) complexity is usually adopted as the computational complexity measurement for evaluating stochastic optimization algorithms [10, 11, 18, 19].

Definition 2 (IFO Complexity)**.** *For $f(\boldsymbol{x})$ in problem (1), an IFO takes in an index $i \in [n]$ and a point $\boldsymbol{x}$, and returns the pair $(f_i(\boldsymbol{x}), \nabla f_i(\boldsymbol{x}))$.*

3 Riemannian SPIDER Algorithm

We first elaborate on the Riemannian SPIDER algorithm, and then analyze its convergence performance for general non-convex problems. For gradient dominated problems, we further develop a variant of R-SPIDER with a linear rate of convergence.

3.1 Algorithm

The R-SPIDER method is outlined in Algorithm 1. At its core, R-SPIDER customizes SPIDER to recursively estimate/track the full Riemannian gradient in a computationally economic way. For each cycle of p iterations, R-SPIDER first samples a large data batch $\mathcal{S}_1$ by with-replacement sampling and views the gradient estimate $\boldsymbol{v}_k = \nabla f_{\mathcal{S}_1}(\boldsymbol{x}_k) = \frac{1}{|\mathcal{S}_1|}\sum_{i\in\mathcal{S}_1} f_i(\boldsymbol{x}_k)$ as the snapshot gradient. For the next forthcoming $p-1$ iterations, R-SPIDER only samples a smaller mini-batch $\mathcal{S}_2$ and estimates the full Riemannian gradient $\nabla f(\boldsymbol{x}_k)$ as $\boldsymbol{v}_k = \nabla f_{\mathcal{S}_2}(\boldsymbol{x}_k) - \mathrm{P}_{\boldsymbol{x}_{k-1}}^{\boldsymbol{x}_k}\left(\nabla f_{\mathcal{S}_2}(\boldsymbol{x}_{k-1}) - \boldsymbol{v}_{k-1}\right)$. Here the parallel transport operator $\mathrm{P}_{\boldsymbol{x}_{k-1}}^{\boldsymbol{x}_k}(\cdot)$ is applied to ensure that the Riemannian gradients can be linearly combined in a common tangent space. If $\|\boldsymbol{v}_k\| > 0.5\epsilon$, then R-SPIDER performs normalized gradient descent to update $\boldsymbol{x}_{k+1} = \mathrm{Exp}_{\boldsymbol{x}_k}\big(-\eta_k \frac{\boldsymbol{v}_k}{\|\boldsymbol{v}_k\|}\big)$. Otherwise, the algorithm terminates and returns $\boldsymbol{x}_k$.

The idea of recursive Riemannian gradient estimation has also been exploited by R-SRG [10]. Although sharing a similar spirit in full gradient approximation, R-SPIDER departs notably from R-SRG: at each iteration, R-SPIDER normalizes the gradient $\boldsymbol{v}_k$ and thus is able to well control the distance $\mathrm{d}(\boldsymbol{x}_k, \boldsymbol{x}_{k+1})$ between $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k+1}$ by properly controlling the stepsize η, while R-SRG directly updates the variable without gradient normalization. It turns out that this normalization step is key to achieving faster convergence speed for non-convex problem in R-SPIDER, since it helps reduce the variance of stochastic gradient estimation by properly controlling the distance $\mathrm{d}(\boldsymbol{x}_k, \boldsymbol{x}_{k+1})$ (see Lemma 1). As a consequence, at each iteration, R-SPIDER only needs to sample a necessary number of data points to estimate Riemannian gradient and decrease the objective sufficiently (see Theorems 1 and 2). In this way, R-SPIDER achieves lower overall computational complexity for solving problem (1).

3.2 Computational complexity analysis

The vanilla SPIDER is known to achieve nearly optimal iteration complexity bounds for stochastic non-convex optimization in Euclidean space [25]. We here show that R-SPIDER generalizes such an appealing property of SPIDER to Riemannian manifolds. We first present the following key lemma which guarantees sufficiently accurate Riemannian gradient estimation for R-SPIDER. We denote $\mathbb{I}_{\{\mathcal{E}\}}$ as the indicator function: if the event $\mathcal{E}$ is true, then $\mathbb{I}_{\{\mathcal{E}\}} = 1$; otherwise, $\mathbb{I}_{\{\mathcal{E}\}} = 0$.

Lemma 1 (Bounded Gradient Estimation Error)**.** *Suppose Assumptions 1 and 2 hold. Let $k_0 = \lfloor k/p \rfloor$ and $\widetilde{k}_0 = k_0 p$. The estimation error between the full Riemannian gradient $\nabla f(\boldsymbol{x}_k)$ and its estimate $\boldsymbol{v}_k$ in Algorithm 1 is bounded as*

$$\mathbb{E}\left[\|\boldsymbol{v}_k - \nabla f(\boldsymbol{x}_k)\|^2 \mid \boldsymbol{x}_{\widetilde{k}_0}, \cdots, \boldsymbol{x}_{\widetilde{k}_0+p-1}\right] \le \mathbb{I}_{\{|\mathcal{S}_1|<n\}}\frac{\sigma^2}{|\mathcal{S}_1|} + \frac{L^2}{|\mathcal{S}_2|}\sum_{i=\widetilde{k}_0}^{\widetilde{k}_0+p-1} d^2(\boldsymbol{x}_i, \boldsymbol{x}_{i+1}),$$

where $d(\boldsymbol{x}_i, \boldsymbol{x}_{i+1})$ is the distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i+1}$.

Proof. The key is to carefully handle the exponential mapping and parallel transport operators introduced for vector computation. See details in Appendix A. □

Algorithm 1 R-SPIDER ($\boldsymbol{x}_0$, ϵ, η, p, $|\mathcal{S}_1|$, $|\mathcal{S}_2|$)

Input: initialization $\boldsymbol{x}_0$, accuracy ϵ, learning rate η, iteration interval p, mini-batch sizes $|\mathcal{S}_1|$ and $|\mathcal{S}_2|$.
for $k = 0$ to $K-1$ **do**
  if $\mathrm{mod}(k, p) = 0$ **then**
    Draw mini-batch $\mathcal{S}_1$ and compute $\boldsymbol{v}_k = \nabla f_{\mathcal{S}_1}(\boldsymbol{x}_k)$;
  else
    Draw mini-batch $\mathcal{S}_2$ and compute $\nabla f_{\mathcal{S}_2}(\boldsymbol{x}_k)$;
    $\boldsymbol{v}_k = \nabla f_{\mathcal{S}_2}(\boldsymbol{x}_k) - \mathrm{P}_{\boldsymbol{x}_{k-1}}^{\boldsymbol{x}_k}\left(\nabla f_{\mathcal{S}_2}(\boldsymbol{x}_{k-1}) - \boldsymbol{v}_{k-1}\right)$;
  end if
  $\boldsymbol{x}_{k+1} = \mathrm{Exp}_{\boldsymbol{x}_k}\left(-\eta_k \frac{\boldsymbol{v}_k}{\|\boldsymbol{v}_k\|}\right)$;
end for
Output: $\widetilde{\boldsymbol{x}}$ which is chosen uniformly at random from $\{\boldsymbol{x}_k\}_{k=0}^{K-1}$.

Lemma 1 tells that by properly selecting the mini-batch sizes $|\mathcal{S}_1|$ and $|\mathcal{S}_2|$, the accuracy of gradient estimate $\boldsymbol{v}_k$ can be controlled. Benefiting from the normalization step, we have $\mathrm{d}(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}) = \|\mathrm{Exp}_{\boldsymbol{x}_k}^{-1}(\boldsymbol{x}_{k+1})\| = \eta$. As a result, the gradient estimation error can be bounded as $\mathbb{E}\big[\|\boldsymbol{v}_k - \nabla f(\boldsymbol{x}_k)\|^2 \mid \boldsymbol{x}_{\widetilde{k}_0}, \cdots, \boldsymbol{x}_{\widetilde{k}_0+p-1}\big] \le \mathbb{I}_{\{|\mathcal{S}_1|<n\}}\frac{\sigma^2}{|\mathcal{S}_1|} + \frac{pL^2\eta^2}{|\mathcal{S}_2|}$. Based on this result, we are able to analyze the rate-of-convergence of R-SPIDER.

Finite-sum setting. We first consider problem (1) under finite-sum setting. By properly selecting parameters, we prove that at each iteration, the sequence $\{\boldsymbol{x}_k\}$ produced by Algorithm 1 can lead to sufficient decrease of the objective loss $f(\boldsymbol{x})$ when $\|\boldsymbol{v}_k\|$ is large. Based on this results, we further derive the iteration number of Algorithm 1 for computing an ϵ-accuracy solution. The result is formally summarized in Theorem 1.

Theorem 1. *Suppose Assumptions 1 and 2 hold. Let $s = \min\big(n, \frac{16\sigma^2}{\epsilon^2}\big)$, $p = n_0 s^{\frac{1}{2}}$, $\eta_k = \min\left(\frac{\epsilon}{2Ln_0}, \frac{\|\boldsymbol{v}_k\|}{4Ln_0}\right)$, $|\mathcal{S}_1| = s$, $|\mathcal{S}_2| = \frac{4s^{\frac{1}{2}}}{n_0}$ and $n_0 \in [1, 4s^{\frac{1}{2}}]$. Then for finite-sum problem (1), the sequence $\{\boldsymbol{x}_k\}$ produced by Algorithm 1 satisfies*

$$\mathbb{E}\left[f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k)\right] \le -\frac{\epsilon}{64Ln_0}\left(12\mathbb{E}[\|\boldsymbol{v}_k\|] - 7\epsilon\right).$$

Moreover, to achieve $\mathbb{E}[\|\nabla f(\widetilde{\boldsymbol{x}})\|] \leq \epsilon$, *Algorithm 1 will terminate at most* $\left(\frac{14Ln_0\Delta}{\epsilon^2}\right)$ *iterations in expectation, where* $\Delta = f(\boldsymbol{x}_0) - f(\boldsymbol{x}_*)$ *with* $\boldsymbol{x}_* = \operatorname{argmin}_{\boldsymbol{x}\in\mathcal{M}} f(\boldsymbol{x})$.

Proof. The result comes readily from geodesically L-gradient-Lipschitz of f and the variance bound in Lemma 1. See Appendix B.1 for a complete proof. □

Theorem 1 shows that Algorithm 1 only needs to run at most $\left(\frac{14Ln_0\Delta}{\epsilon^2}\right)$ iteration to compute an ϵ-accuracy solution $\widetilde{\boldsymbol{x}}$, *i.e.* $\mathbb{E}[\|\nabla f(\widetilde{\boldsymbol{x}})\|] \leq \epsilon$. This means the convergence rate of R-SPIDER is at the order of $\mathcal{O}\left(\frac{Ln_0\Delta}{\epsilon^2}\right)$. Besides, by one iteration loop of Algorithm 1, the objective value $f(\boldsymbol{x}_k)$ monotonously decreases in expectation when $\mathbb{E}[\|\boldsymbol{v}_k\|]$ is large, *e.g.* $\mathbb{E}[\|\boldsymbol{v}_k\|] \geq \frac{7\epsilon}{12}$. By comparison, Kasai *et al.* [10] only proved the sublinear convergence rate of the gradient norm $\mathbb{E}[\|\nabla f(\boldsymbol{x})\|^2]$ in R-SRG and did not reveal any convergence behavior of the objective $f(\boldsymbol{x})$. Moreover, Theorem 1 yields as a byproduct the benefits of mini-batching to R-SPIDER. Indeed, by controlling the parameter n_0 in R-SPIDER, the mini-batch size $|\mathcal{S}_2|$ at each iteration can range from 1 to $\min\left(4\sqrt{n}, \frac{16\sigma}{\epsilon}\right)$. Also, it can be seen from Theorem 1 that larger mini-batch size allows more aggressive step size η_k and thus leads to less necessary iterations to achieve an ϵ-accuracy solution. More specifically, the convergence rate bound $\mathcal{O}\left(\frac{Ln_0\Delta}{\epsilon^2}\right)$ indicates that at least in theory, increasing the mini-batch sizes in R-SPIDER provides linear speedups in parallel computing environment. In contrast, these important benefits of mini-batching are not explicitly analyzed in the existing Riemannian stochastic gradient algorithms [10, 11].

Based on Theorem 1, we can derive the IFO complexity of R-SPIDER for non-convex problems in Corollary 1.

Corollary 1. *Using the same assumptions and parameters in Theorem 1, the IFO complexity of Algorithm 1 is* $\mathcal{O}\left(\min\left(n + \frac{L\Delta\sqrt{n}}{\epsilon^2}, \frac{L\Delta\sigma}{\epsilon^3}\right)\right)$ *for achieving* $\mathbb{E}[\|\nabla f(\widetilde{\boldsymbol{x}})\|] \leq \epsilon$.

Proof. The result is obtained directly from a cumulation of IFOs at each step of iteration. See Appendix B.2. □

From Corollary 1, the IFO complexity of R-SPIDER for non-convex finite-sum problems is at the order of $\mathcal{O}\left(\frac{1}{\epsilon^2}\min\left(\sqrt{n}, \frac{1}{\epsilon}\right)\right)$. This result matches the state-of-the-art complexity bounds for general non-convex optimization problems in Euclidean space [25, 30]. Indeed, under Assumption 1, Fang *et al.* [25] proved that the lower IFO complexity bound for finite-sum problem (1) in Euclidean space is $\mathcal{O}\left(n + \frac{L\Delta\sqrt{n}}{\epsilon^2}\right)$ when the number n of the component function obeys $n \leq \mathcal{O}\left(\frac{L^2\Delta^2}{\epsilon^4}\right)$. In the sense that Euclidean space is a special case of Riemannian manifold, our IFO complexity $\mathcal{O}\left(n + \frac{L\Delta\sqrt{n}}{\epsilon^2}\right)$ for finite-sum problem (1) under Assumption 1 is nearly optimal. If Assumption 2 holds in addition, we can establish tighter IFO complexity $\mathcal{O}\left(\frac{1}{\epsilon^2}\min\left(\sqrt{n}, \frac{1}{\epsilon}\right)\right)$. This is because when the sample number n satisfies $n \geq \frac{16\sigma^2}{\epsilon^2}$, by sampling $|\mathcal{S}_1| = \frac{16\sigma^2}{\epsilon^2}$ and $|\mathcal{S}_2| = \frac{16\sigma}{n_0\epsilon}$, the gradient estimation error already satisfies $\mathbb{E}[\|\boldsymbol{v}_k - \nabla f(\boldsymbol{x}_k)\|^2] \leq \frac{\epsilon^2}{8}$. Accordingly, if $\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\boldsymbol{v}_k\| \leq 0.5\epsilon$ which is actually achieved after K iterations, then $\mathbb{E}[\|f(\widetilde{\boldsymbol{x}})\|] = \frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\nabla f(\boldsymbol{x}_k)\| \leq \frac{1}{K}\sum_{k=0}^{K-1}[\mathbb{E}\|\nabla f(\boldsymbol{x}_k) - \boldsymbol{v}_k\| + \mathbb{E}\|\boldsymbol{v}_k\|] \leq \epsilon$. So here it is only necessary to sample $|\mathcal{S}_1| = \frac{16\sigma^2}{\epsilon^2}$ data points instead of the entire set of n samples.

Kasai *et al.* [10] proved that the IFO complexity of R-SRG is at the order of $\mathcal{O}\left(n + \frac{L^2}{\epsilon^4}\right)$ to obtain an ϵ-accuracy solution. By comparison, we prove that R-SPIDER enjoys the complexity of $\mathcal{O}\left(\frac{1}{\epsilon^2}\min\left(\sqrt{n}, \frac{1}{\epsilon}\right)\right)$, which is at least lower than R-SRG by a factor of $\frac{1}{\epsilon}$. This is because the normalization step in R-SPIDER allows us to well control the gradient estimation error and thus avoids sampling too many redundant samples at each iteration, resulting in sharper IFO complexity. Zhang *et al.* [11] showed that R-SVRG has the IFO complexity $\mathcal{O}\left(n + \frac{\zeta^{1/2}n^{2/3}}{\epsilon^2}\right)$, where $\zeta \geq 1$ denotes the curvature parameter. Therefore, R-SPIDER improves over R-SVRG by a factor at least $n^{1/6}$ in IFO complexity. Note, here the curvature parameter ζ does not appear in our bounds, since we have avoided using the trigonometry inequality which characterizes the trigonometric geometric in Riemannian manifold [11, 16, 17].

The exponential mapping and parallel transport operators used in R-SPIDER are respectively classical instances of the more general concepts of retraction and vector transport [31, 32]. We note that under identical assumptions in [10], the convergence rate and IFO complexity bounds for R-SPIDER generalize well to the setting where exponential mapping and parallel transport are replaced by retraction and vector transport operators. Some specific ways of constructing retraction and vector transport are available in [31, 33, 34].

Online setting. Next we consider the online setting of problem (1). Similar to finite-sum setting, we prove in Theorem 2 that the objective $f(\boldsymbol{x})$ can be sufficiently decreased when the gradient norm is not too small.

Theorem 2. *Suppose Assumptions 1 and 2 hold. Let* $p = \frac{\sigma n_0}{\epsilon}$, $\eta_k = \min\left(\frac{\epsilon}{2Ln_0}, \frac{\|\boldsymbol{v}_k\|}{4Ln_0}\right)$, $|\mathcal{S}_1| = \frac{64\sigma^2}{\epsilon^2}$, $|\mathcal{S}_2| = \frac{4\sigma}{\epsilon n_0}$ *and* $n_0 \in [1, 4\sigma/\epsilon]$. *Then for problem (1) under online setting, the sequence* $\{\boldsymbol{x}_k\}$ *produced by Algorithm 1 satisfies*

$$\mathbb{E}\left[f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k)\right] \leq -\frac{\epsilon}{64Ln_0}\left(12\mathbb{E}[\|\boldsymbol{v}_k\|] - 7\epsilon\right).$$

Moreover, to achieve $\mathbb{E}[\|\nabla f(\widetilde{\boldsymbol{x}})\|] \leq \epsilon$, *Algorithm 1 will terminate at most* $\left(\frac{14Ln_0\Delta}{\epsilon^2}\right)$ *iterations in expectation,*

where $\Delta = f(\boldsymbol{x}_0) - f(\boldsymbol{x}_*)$ *with* $\boldsymbol{x}_* = \operatorname{argmin}_{\boldsymbol{x}\in\mathcal{M}} f(\boldsymbol{x})$.

Proof. The proof mimics that of Theorem 1 with proper adaptation to online setting. See Appendix B.3. □

As a direct consequence of this result, the following corollary establishes the IFO complexity of R-SPIDER for the online optimization.

Corollary 2. *Using the same assumptions and parameters in Theorem 2, the IFO complexity of Algorithm 1 is* $\mathcal{O}\left(\frac{L\sigma\Delta}{\epsilon^3}\right)$ *to achieve* $\mathbb{E}[\|\nabla f(\boldsymbol{x})\|] \leq \epsilon$.

Proof. See Appendix B.4 for a proof of this result. □

Bonnabel *et al.* [17] have also analyzed R-SGD under online setting, but only with asymptotic convergence guarantee obtained. By comparison, we for the first time establish non-asymptotic complexity bounds for Riemannian online non-convex optimization.

Algorithm 2 Riemannian Gradient Dominated SPIDER (R-GD-SPIDER)

1: **Input:** initial point $\widetilde{\boldsymbol{x}}_0$, initial accuracy ϵ_0, learning rate η^0, mini-batch sizes $|\mathcal{S}_1^0|$ and $|\mathcal{S}_2^0|$, iteration interval p^0, final accuracy ϵ
2: **for** $t = 1$ to T **do**
3: $\quad\widetilde{\boldsymbol{x}}_t = \text{R-SPIDER}(\widetilde{\boldsymbol{x}}_{t-1}, \epsilon_{t-1}, \eta^t, p^t, |\mathcal{S}_1^t|, |\mathcal{S}_2^t|)$.
4: $\quad$Set $\epsilon_t = 0.5\epsilon_{t-1}$, and $\eta^t, p^t, |\mathcal{S}_1^t|, |\mathcal{S}_2^t|$ properly.
5: **end for**
6: **Output:** $\widetilde{\boldsymbol{x}}_t$

3.3 On gradient dominated functions

We now turn to a special case of problem (1) with gradient dominated loss function as defined in Definition 1. For instance, the strongly geodesically convex (SGC) functions[1] are gradient dominated. Some non-strongly convex problems, *e.g.* ill-conditioned linear prediction and logistic regression [35], and Riemannian non-convex problems, *e.g.* PCA [11], also belong to gradient dominated functions. Please refer to [29, 35] for more instances of gradient dominated functions. To better fit gradient dominated functions, we develop the Riemannian gradient dominated SPIDER (R-GD-SPIDER) as a multi-stage variant of R-SPIDER. A high-level description of R-GD-SPIDER is outlined in Algorithm 2. The basic idea is to use more aggressive learning rates in early stage of processing and gradually shrink the learning rate in later stage. With the help of such a simulated annealing process, R-GD-SPIDER exhibits linear convergence behavior for finite-sum problems, as formally stated in Theorem 3. For the t-th iteration in Algorithm 2, R-SPIDER uses $|\mathcal{S}_1^t|$ and $|\mathcal{S}_{2,k}^t|$ samples to compute $\boldsymbol{v}_k$ for $k = \lfloor k/p \rfloor \cdot p$ and $k \neq \lfloor k/p \rfloor \cdot p$, respectively.

Theorem 3. *Suppose that function* $f(\boldsymbol{x})$ *is* τ*-gradient dominated, and Assumptions 1 and 2 hold. For finite-sum setting, at the* t*-th iteration, set* $\epsilon_0 = \frac{\sqrt{\Delta}}{2\sqrt{\tau}}$, $\epsilon_t = \frac{\epsilon_0}{2^t}$, $s_t = \min\left(n, \frac{32\sigma^2}{\epsilon_{t-1}^2}\right)$, $p^t = n_0^t\sqrt{s_t}$, $\eta_k^t = \frac{\|\boldsymbol{v}_k^t\|}{2Ln_0}$, $|\mathcal{S}_1^t| = s_t$, $|\mathcal{S}_{2,k}^t| = \min(\frac{8p^t\|\boldsymbol{v}_{k-1}^t\|^2}{(n_0^t)^2\epsilon_{t-1}^2}, n)$, *where* $n_0^t \in [1, \frac{8\sqrt{s_t}\|\boldsymbol{v}_{k-1}^t\|^2}{\epsilon_{t-1}^2}]$.
(1) *The sequence* $\{\widetilde{\boldsymbol{x}}_t\}$ *produced by Algorithm 2 satisfies*

$$\mathbb{E}\left[f(\widetilde{\boldsymbol{x}}_t) - f(\boldsymbol{x}_*)\right] \leq \frac{\Delta}{4^t} \quad \textit{and} \quad \mathbb{E}[\|\nabla f(\widetilde{\boldsymbol{x}}_t)\|] \leq \frac{1}{2^t}\sqrt{\frac{\Delta}{\tau}},$$

where $\Delta = f(\widetilde{\boldsymbol{x}}_0) - f(\boldsymbol{x}_*)$ *with* $\boldsymbol{x}_* = \operatorname{argmin}_{\boldsymbol{x}\in\mathcal{M}} f(\boldsymbol{x})$.
(2) *To achieve* $\mathbb{E}[\|\nabla f(\widetilde{\boldsymbol{x}}_T)\|] \leq \epsilon$, *in expectation the IFO complexity is* $\mathcal{O}\left(\min\left((n + \tau L\sqrt{n})\log\left(\frac{1}{\epsilon}\right), \frac{\tau L\sigma}{\epsilon}\right)\right)$.

Proof. The part (1) follows immediately from the update rule of ϵ_t. The part (2) can be proved by establishing the IFO bound $\min\left(n + \tau L\sqrt{n}, \frac{\tau L\sigma}{\epsilon_{t+1}}\right)$ for each stage t and then putting them together. See Appendix C.1. □

The main message conveyed by Theorem 3 is that R-GD-SPIDER enjoys a global linear rate of convergence and its IFO complexity is at the order of $\mathcal{O}\left(\min\left((n + \tau L\sqrt{n})\log\left(\frac{1}{\epsilon}\right), \frac{\tau L\sigma}{\epsilon}\right)\right)$. For R-SVRG with τ-gradient dominated functions, Zhang *et al.* [11] also established a linear convergence rate and an IFO complexity bound $\mathcal{O}\left((n + \tau L\zeta^{\frac{1}{2}}n^{\frac{2}{3}})\log\left(\frac{1}{\epsilon}\right)\right)$. As a comparison, our R-GD-SPIDER makes an improvement over R-SVRG in IFO complexity by a factor of $n^{\frac{1}{6}}$. For R-SRG [10], the corresponding IFO complexity is $\mathcal{O}\left((n + \tau^2L^2)\log\left(\frac{1}{\epsilon}\right)\right)$. Therefore, in terms of IFO complexity, R-GD-SPIDER is superior to R-SRG when the optimization accuracy ϵ is moderately small at a huge data size n.

Turning to the online setting, R-GD-SPIDER also converges linearly, as formally stated in Theorem 4.

Theorem 4. *Suppose that* $f(\boldsymbol{x})$ *is* τ*-gradient dominated, and Assumptions 1 and 2 hold. For online setting, at the* t*-th iteration, let* $\epsilon_0 = \frac{\sqrt{\Delta}}{2\sqrt{\tau}}$, $\epsilon_t = \frac{\epsilon_0}{2^t}$, $p_t = \frac{\sigma n_0^t}{\epsilon_{t-1}}$, $\eta_k^t = \frac{\|\boldsymbol{v}_k^t\|}{2Ln_0^t}$, $|\mathcal{S}_1^t| = \frac{32\sigma^2}{\epsilon_{t-1}^2}$, $|\mathcal{S}_{2,k}^t| = \frac{8\sigma\|\boldsymbol{v}_{k-1}^t\|^2}{\epsilon_{t-1}^3 n_0^t}$, *where* $n_0^t \in [1, \frac{8\sigma\|\boldsymbol{v}_{k-1}^t\|^2}{\epsilon_{t-1}^3}]$.
(1) *The sequence* $\{\widetilde{\boldsymbol{x}}_t\}$ *produced by Algorithm 2 satisfies*

$$\mathbb{E}\left[f(\widetilde{\boldsymbol{x}}_t) - f(\boldsymbol{x}_*)\right] \leq \frac{\Delta}{4^t} \quad \textit{and} \quad \mathbb{E}[\|\nabla f(\widetilde{\boldsymbol{x}}_t)\|] \leq \frac{1}{2^t}\sqrt{\frac{\Delta}{\tau}},$$

[1] A strongly geodesically convex function satisfies $f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \langle\nabla f(\boldsymbol{x}), \operatorname{Exp}_{\boldsymbol{x}}^{-1}(\boldsymbol{y})\rangle + \frac{\mu}{2}\|\operatorname{Exp}_{\boldsymbol{x}}^{-1}(\boldsymbol{y})\|^2, \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{M}$, for som $\mu > 0$, which immediately implies $f(\boldsymbol{x}) - f(\boldsymbol{x}_*) \leq \frac{1}{2\mu}\|\nabla f(\boldsymbol{x})\|^2$ by Cauchy-Schwarz inequality.

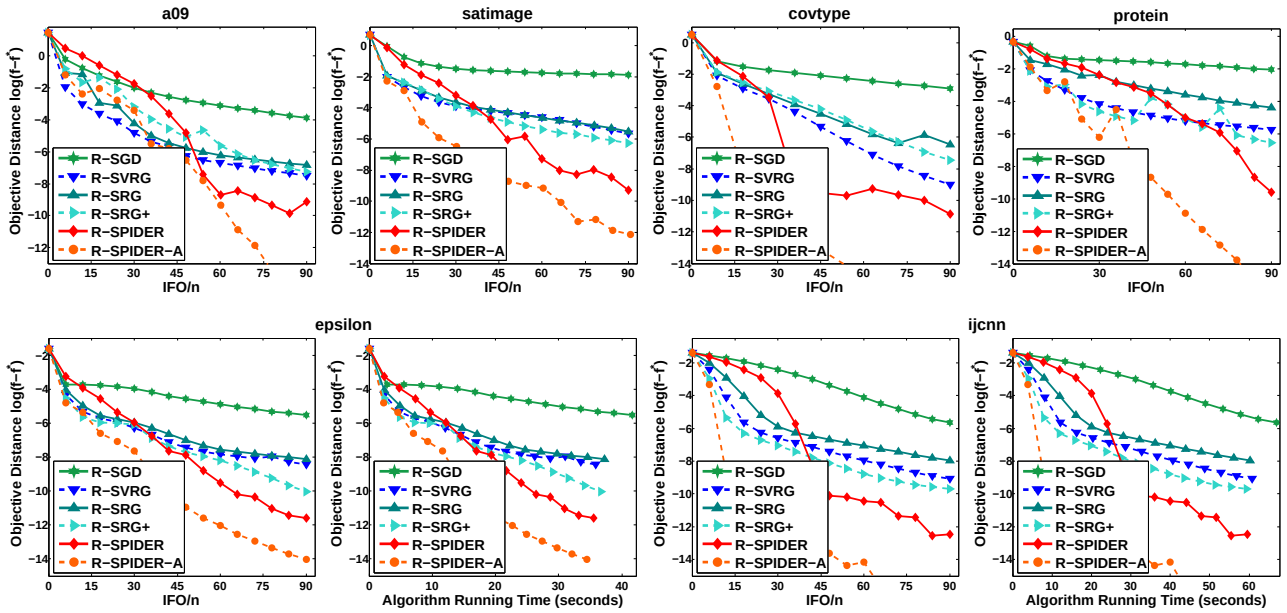


Figure 1: Comparison among Riemannian stochastic gradient algorithms on k-PCA problem.

where $\Delta = f(\widetilde{\boldsymbol{x}}_0) - f(\boldsymbol{x}_*)$ *with* $\boldsymbol{x}_* = \operatorname{argmin}_{\boldsymbol{x}\in\mathcal{M}} f(\boldsymbol{x})$. ***(2)*** *To achieve* $\mathbb{E}[\|\nabla f(\widetilde{\boldsymbol{x}}_T)\|] \leq \epsilon$, *in expectation the IFO complexity is* $\mathcal{O}\left(\frac{\tau L\sigma}{\epsilon}\right)$.

Proof. See Appendix C.2 for a proof of this result. □

Such a non-asymptotic convergence result is new to online Riemannian gradient dominated optimization.

4 Experiments

In this section, we compare the proposed R-SPIDER with several state-of-the-art Riemannian stochastic gradient algorithms, including R-SGD [17], R-SVRG [11, 18], R-SRG [10] and R-SRG+ [10]. We evaluate all the considered algorithms on two learning tasks: the k-PCA problem and the low-rank matrix completion problem. We run simulations on ten datasets, including six datasets from LibSVM, three face datasets (YaleB, AR and PIE) and one recommendation dataset (MovieLens-1M). The details of these datasets are described in Appendix D. For all the considered algorithms, we tune their hyper-parameters optimally.

A practical implementation of R-SPIDER. To achieve the IFO complexity in Corollary 1, it is suggested to set the learning rate as $\eta = \frac{\epsilon}{4Ln_0}$ where ϵ is the desired optimization accuracy. However, since in the initial epochs the computed point is far from the optimum to problem (1), using a tiny learning rate could usually be conservative. In contrast, by using a more aggressive learning rate at the initial optimization stage, we can expect stable but faster convergence behavior. Here for R-SPIDER we design a decaying learning rate with formulation $\eta_k = \alpha^{\lfloor \frac{k}{p} \rfloor} \cdot \beta$ and call it "R-SPIDER-A", where α and β are two constants. In our experiments, α is selected from $\{0.8, 0.85, 0.9, 0.95, 0.99\}$ and β from $\{5\times10^{-2}, 10^{-2}, 5\times10^{-3}, 10^{-3}\}$.

Evaluation on the k-PCA problem. Given n data points, k-PCA aims at computing their first k leading eigenvectors, which is formulated as $\min_{\boldsymbol{U}\in\mathsf{Gr}(k,d)} \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{a}_i^\top \boldsymbol{U}\boldsymbol{U}^T\boldsymbol{a}_i$, where $\boldsymbol{a}_i \in \mathbb{R}^d$ denotes the i-th sample vector and $\mathsf{Gr}(k,d) = \{\boldsymbol{U}\in\mathbb{R}^{d\times k} \mid \boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I}\}$ denotes the Grassmann manifold. For this problem, we can directly obtain the ground truth $\boldsymbol{U}^*$ by using singular value decomposition (SVD), and then use $f(\boldsymbol{U}^*)$ as optimal value f^* for sub-optimality estimation in Figures 1 and 2. In this experiment, we compute the first ten leading eigenvectors.

From the experimental results in Figure 1, one can observe that our R-SPIDER-A converges significantly faster than other algorithms and R-SPIDER can also quickly converge to a relatively high accuracy, e.g. 10^{-8}. In the initial epochs, R-SPIDER is comparable to other algorithms, showing relatively flat convergence behavior, mainly due to its very small learning rate and gradient normalization. Then along with more iterations, the computed solution becomes close to the optimum. Accordingly, the gradient begins to vanish and those considered algorithms without normalization tend to update the variable with small progress. In contrast, thanks to the normalization step, R-SPIDER moves more rapidly along the gradient descent direction and thus has sharper convergence curves. Meanwhile, R-SPIDER-A uses a relatively more aggressive learning rate in the initial epochs and decreases the learning rate along with more iterations. As a result, it exhibits

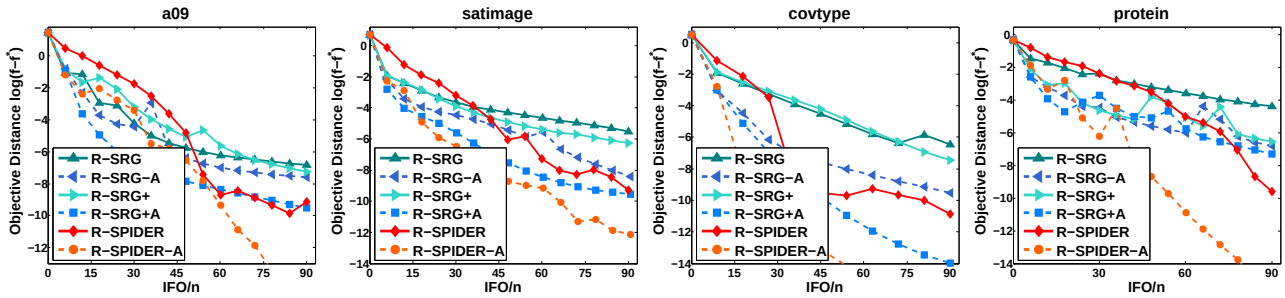


Figure 2: Comparison between R-SPIDER and R-SRG with adaptive learning rates on k-PCA problem.

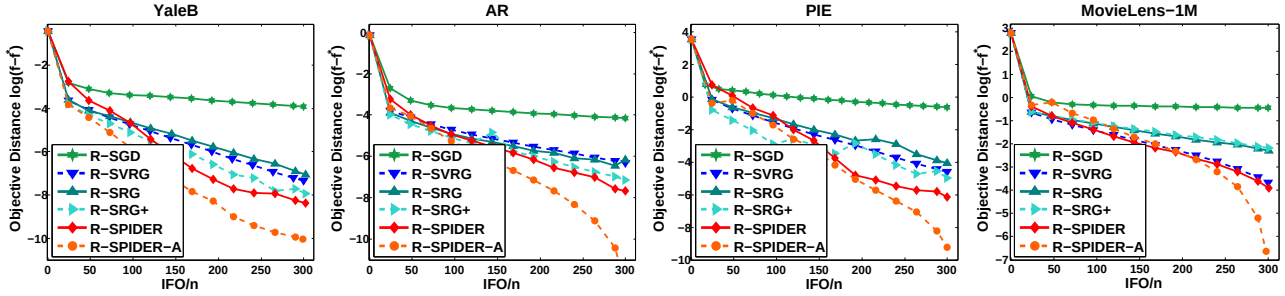


Figure 3: Comparison among Riemannian stochastic gradient algorithms on low-rank matrix completion problem.

the sharpest convergence behavior. On epsilon and ijcnn datasets (the bottom of Figure 1) we further plot the sub-optimality versus running-time curves. The main observations from this group of curves are consistent with those of IFO complexity, implying that the IFO complexity can comprehensively reflect the overall computational performance of a first-order Riemannian algorithm. See Figure 4 in Appendix D.2 for more experimental results on running time comparison.

In Figure 2, we compare R-SPIDER-A more closely with R-SRG-A and R-SRG+A which are respectively the counterparts of R-SRG and R-SRG+ with adaptive learning rate $\eta_k = \alpha(1 + \alpha\lambda_\alpha\lfloor\frac{k}{p}\rfloor)$ [10]. Here α and λ_α are tunable hyper-parameters. From the results, one can observe that the algorithm with adaptive learning rate usually outperforms the vanilla counterpart, which demonstrates the effectiveness of such an implementation trick. Moreover, R-SPIDER-A is consistently superior to R-SRG-A and R-SRG+A. See Figure 5 in Appendix D.3 for more results in this line.

Evaluation on the low-rank matrix completion (LRMC) problem. Given a low-rank incomplete observation $\boldsymbol{A} \in \mathbb{R}^{d\times n}$, the LRMC problem aims at exactly recovering $\boldsymbol{A}$. The mathematical formulation is $\min_{\boldsymbol{U}\in\mathsf{Gr}(k,d),\boldsymbol{G}\in\mathbb{R}^{k\times n}} \|\mathcal{P}_\Omega(\boldsymbol{A}) - \mathcal{P}_\Omega(\boldsymbol{UG})\|^2$, where the set Ω of locations corresponds to the observed entries, namely $(i,j) \in \Omega$ if $\boldsymbol{A}_{ij}$ is observed. $\mathcal{P}_\Omega$ is a linear operator that extracts entries in Ω and fills the entries not in Ω with zeros. The LRMC problem can be expressed equivalently as $\min_{\boldsymbol{U}\in\mathsf{Gr}(k,d),\boldsymbol{G}_i\in\mathbb{R}^k} \frac{1}{n}\sum_{i=1}^n \|\mathcal{P}_{\Omega_i}(\boldsymbol{A}_i) - \mathcal{P}_{\Omega_i}(\boldsymbol{UG}_i)\|^2$. Since there is no ground truth for the optimum, we run Riemannian GD sufficiently long until the gradient satisfies $\|\nabla f(\boldsymbol{x})\|/\|\boldsymbol{x}\| \le 10^{-8}$, and then use the output as an approximate optimal value f^* for sub-optimality estimation in Figure 3. We test the considered algorithms on YaleB, AR, PIE and MovieLens-1M, considering these data approximately lie on a union of low-rank subspaces [10, 36]. For face images, we randomly sample 30% pixels in each image as the observations and set $k = 30$. For MovieLens-1M, we use its one million ratings for 3,952 movies from 6,040 users as the observations and set $k = 100$.

From Figure 3, R-SPIDER-A and R-SPIDER show very similar behaviors as those in Figure 1. More specifically, R-SPIDER-A achieves fastest convergence rate, and R-SPIDER has similar convergence speed as other algorithms in the initial epochs and then runs faster along with more epochs. All these results confirm the superiority of R-SPIDER and R-SPIDER-A.

5 Conclusions

We proposed R-SPIDER, which is an efficient Riemannian gradient method for non-convex stochastic optimization on Riemannian manifolds. Compared to existing first-order Riemannian algorithms, R-SPIDER enjoys provably lower computational complexity bounds for finite-sum minimization. For online optimization, similar non-asymptotic bounds are established for R-SPIDER, which to our best knowledge has not been addressed in previous study. For the special case of gradient dominated functions, we further developed a variant of R-SPIDER with improved linear rate of convergence. Numerical results confirm the computational superiority of R-SPIDER over the state-of-the-arts.

References

[1] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987. 1

[2] M. Tan, I. Tsang, L. Wang, B. Vandereycken, and S. Pan. Riemannian pursuit for big matrix recovery. In *Proc. Int'l Conf. Machine Learning*, pages 1539–1547, 2014. 1

[3] B. Vandereycken. Low-rank matrix completion by Riemannian optimization. *SIAM Journal on Optimization*, 23(2):1214–1236, 2013. 1

[4] B. Mishra and R. Sepulchre. R3MC: A Riemannian three-factor algorithm for low-rank matrix completion. In *Proc. IEEE Conf. on Decision and Control*, pages 1137–1142, 2014. 1

[5] H. Kasai and B. Mishra. Low-rank tensor completion: a Riemannian manifold preconditioning approach. In *Proc. Int'l Conf. Machine Learning*, pages 1012–1021, 2016. 1

[6] A. Cherian and S. Sra. Riemannian dictionary learning and sparse coding for positive definite matrices. *IEEE trans. on Neural Networks and Learning Systems*, 28(12):2859–2871, 2017. 1

[7] J. Sun, Q. Qu, and J. Wright. Complete dictionary recovery over the sphere ii: Recovery by Riemannian trust-region method. *IEEE Trans. on Information Theory*, 63(2):885–914, 2017. 1

[8] R. Hosseini and S. Sra. Matrix manifold optimization for Gaussian mixtures. In *Proc. Conf. Neutral Information Processing Systems*, pages 910–918, 2015. 1

[9] G. Meyer, S. Bonnabel, and R. Sepulchre. Linear regression under fixed-rank constraints: a Riemannian approach. In *Proc. Int'l Conf. Machine Learning*, 2011. 1

[10] H. Kasai, H. Sato, and B. Mishra. Riemannian stochastic recursive gradient algorithm with retraction and vector transport and its convergence analysis. In *Proc. Int'l Conf. Machine Learning*, pages 2521–2529, 2018. 1, 2, 3, 4, 5, 6, 7, 8, 19

[11] H. Zhang, S. Reddi, and S. Sra. Riemannian SVRG: Fast stochastic optimization on Riemannian manifolds. In *Proc. Conf. Neutral Information Processing Systems*, pages 4592–4600, 2016. 1, 2, 3, 5, 6, 7

[12] E. Oja. Principal components, minor components, and linear neural networks. *Neural networks*, 5(6):927–935, 1992. 1

[13] J. da Cruz Neto, L. De Lima, and P. Oliveira. Geodesic algorithms in Riemannian geometry. *Balkan Journal of Geometry and its Applications*, 3(2):89–100, 1998. 1

[14] R. Badeau, B. David, and G. Richard. Fast approximated power iteration subspace tracking. *IEEE Trans. on Signal Processing*, 53(8):2931–2941, 2005. 1

[15] H. Zhang and S. Sra. An estimate sequence for geodesically convex optimization. In *Proc. Conf. on Learning Theory*, pages 1703–1723, 2018. 1

[16] H. Zhang and S. Sra. First-order methods for geodesically convex optimization. In *Proc. Conf. on Learning Theory*, pages 1617–1638, 2016. 1, 5

[17] S. Bonnabel. Stochastic gradient descent on Riemannian manifolds. *IEEE Trans. Automatic Control*, 58(9):2217–2229, 2013. 1, 2, 3, 5, 6, 7

[18] H. Kasai, H. Sato, and B. Mishra. Riemannian stochastic variance reduced gradient on Grassmann manifold. *arXiv preprint arXiv:1605.07367*, 2016. 1, 2, 3, 7

[19] H. Kasai, H. Sato, and B. Mishra. Riemannian stochastic quasi-Newton algorithm with variance reduction and its convergence analysis. *Prof. Int'l Conf. Artificial Intelligence and Statistics*, 2018. 1, 2, 3

[20] Y. Liu, F. Shang, J. Cheng, H. Cheng, and L. Jiao. Accelerated first-order methods for geodesically convex optimization on Riemannian manifolds. In *Proc. Conf. Neutral Information Processing Systems*, pages 4868–4877, 2017. 1

[21] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*. Springer Science & Business Media, 2006. 1

[22] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Proc. Conf. Neutral Information Processing Systems*, pages 315–323, 2013. 2

[23] L. Nguyen, J. Liu, K. Scheinberg, and M. Takáč. SARAH: A novel method for machine learning problems using stochastic recursive gradient. *Proc. Int'l Conf. Machine Learning*, 2018. 2

[24] L. Nguyen, J. Liu, K. Scheinberg, and M. Takáč. Stochastic recursive gradient algorithm for nonconvex optimization. *arXiv preprint arXiv:1705.07261*, 2017. 2

[25] C. Fang, C. Li, Z. Lin, and T. Zhang. SPIDER: Near-optimal non-convex optimization via stochastic path integrated differential estimator. *arXiv preprint arXiv:1807.01695*, 2018. 2, 4, 5, 11

[26] W. Huang, P. Absil, and K. Gallivan. A Riemannian symmetric rank-one trust-region method. *Mathematical Programming*, 150(2):179–216, 2015. 3

[27] W. Huang, K. Gallivan, and P. Absil. A broyden class of quasi-Newton methods for Riemannian optimization. *SIAM Journal on Optimization*, 25(3):1660–1685, 2015. 3

[28] B. Polyak. Gradient methods for the minimisation of functionals. *USSR Computational Mathematics and Mathematical Physics*, 3(4):864–878, 1963. 3

[29] Y. Nesterov and B. Polyak. Cubic regularization of Newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006. 3, 6

[30] D. Zhou, P. Xu, and Q. Gu. Stochastic nested variance reduction for nonconvex optimization. *arXiv preprint arXiv:1806.07811*, 2018. 5

[31] R. Adler, J. Dedieu, J. Margulies, M. Martens, and M. Shub. Newton's method on Riemannian manifolds and a geometric model for the human spine. *IMA Journal of Numerical Analysis*, 22(3):359–390, 2002. 5

[32] P. Absil, R. Mahony, and R. Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009. 5

[33] P. Absil and J. Malick. Projection-like retractions on matrix manifolds. *SIAM Journal on Optimization*, 22(1):135–158, 2012. 5

[34] Z. Wen and W. Yin. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1-2):397–434, 2013. 5

[35] H. Karimi, J. Nutini, and M. Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 795–811. Springer, 2016. 6

[36] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM*, 58(3):11, 2011. 8

[37] A. Georghiades, P. Belhumeur, and D. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23:643–660, Jun. 2001. 18

[38] A. Martinez and R. Benavente. The AR face database. *CVC Tech. Rep. 24*, Jun. 1998. 18

[39] T. Sim, S. Baker, and M. Bsat. The CMU pose, illumination, and expression database. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25:1615–1618, Dec. 2003. 18

Faster First-Order Methods for Stochastic Non-Convex Optimization on Riemannian Manifolds (Supplementary File)

This supplementary document contains the technical proofs of convergence results and some additional numerical results of the manuscript entitled "Faster First-Order Methods for Stochastic Non-convex Optimization on Riemannian Manifolds". It is structured as follows. The proof of the key lemma, namely Lemma 1 in Section 3.2, is presented in Appendix A. Then Appendix B.1 provides the proofs of the main results for general finite-sum non-convex problems in Section 3.2, including Theorem 1 and Corollary 1. Next, Appendix B.3 gives the proof of the results for online setting, including Theorem 2 and Corollary 2. For gradient dominated results in Section 3.3, including Theorems 3 and 4, are given in Appendix C.1. Finally, the detailed descriptions of datasets and more experimental results are provided in Appendix D.

A Proofs of Lemma 1

Before proving Lemma 1, we first present an useful lemma from [25]. Let $Q(\boldsymbol{x})$ denote arbitrary determinstic vector and $\xi_k(\boldsymbol{x}_{0:k})$ denote the unbiased estimate $Q(\boldsymbol{x}_k) - Q(\boldsymbol{x}_{k-1})$. Namely, $\mathbb{E}[\xi_k(\boldsymbol{x}_{0:k})|\boldsymbol{x}_{0:k}] = Q(\boldsymbol{x}_k) - Q(\boldsymbol{x}_{k-1})$. Then we aim to use the stochastic differential estimate to approximate $Q(\boldsymbol{x}_k)$ as follows:

$$\tilde{Q}(\boldsymbol{x}_k) = \tilde{Q}(\boldsymbol{x}_0) + \sum_{i=1}^{k} \xi_i(\boldsymbol{x}_{0:i}),$$

where $\tilde{Q}(\boldsymbol{x}_0)$ is the estimation of $Q(\boldsymbol{x}_0)$.

Lemma 2. *[25] For any vector h, we have*

$$\mathbb{E}\|\tilde{Q}(\boldsymbol{x}_k) - Q(\boldsymbol{x}_k)\|^2 \leq \mathbb{E}\|\tilde{Q}(\boldsymbol{x}_0) - Q(\boldsymbol{x}_0)\|^2 + \sum_{i=1}^{k} \mathbb{E}\|\xi_i(\boldsymbol{x}_{0:i}) - (Q(\boldsymbol{x}_i) - Q(\boldsymbol{x}_{i-1}))\|^2.$$

Let $\mathcal{A}_i$ map any vector $\boldsymbol{x}$ to a random vector esimate $\mathcal{A}_i(\boldsymbol{x})$ such that

$$\mathbb{E}[\mathcal{A}_i(\boldsymbol{x}_k) - \mathcal{A}_i(\boldsymbol{x}_{k-1})|\boldsymbol{x}_{0:k}] = \mathcal{V}_k - \mathcal{V}_{k-1}, \tag{4}$$

where $\mathcal{V}_k$ is defined below. Assume $\mathcal{A}_{\mathcal{S}} = \frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} \mathcal{A}_i$ where $\mathcal{S}$ denote the sampled data of sample number $|\mathcal{S}|$. Besides, $\mathcal{A}_i$ satisfies

$$\mathbb{E}_i\|\mathcal{A}_i(\boldsymbol{x}) - \mathcal{A}_i(\boldsymbol{y})\|_2^2 \leq L^2\|\mathrm{Exp}_{\boldsymbol{x}}^{-1}(\boldsymbol{y})\|^2.$$

Then we define $\mathcal{V}_k = \mathcal{A}_{\mathcal{S}}(\boldsymbol{x}_k) - \mathcal{A}_{\mathcal{S}}(\boldsymbol{x}_{k-1}) + \mathcal{V}_{k-1}$ and $\mathcal{V}_0$ is the estimate of $\mathcal{A}(\boldsymbol{x}_0)$. Based on Lemma 2, we can further conclude:

Lemma 3. *Assume $d(\boldsymbol{x}_{k-1}, \boldsymbol{x}_k) = \|Exp_{\boldsymbol{x}_k}^{-1}(\boldsymbol{x}_{k-1})\| = \rho_{k-1}$. Then we have*

$$\mathbb{E}\|\mathcal{V}_k - \mathcal{A}(\boldsymbol{x}_k)\|^2 \leq \mathbb{E}\|\mathcal{V}_0 - \mathcal{A}(\boldsymbol{x}_0)\|^2 + L^2\sum_{i=1}^{t}\mathbb{I}_{\{|\mathcal{S}_i|<n\}}\frac{\rho_{i-1}^2}{|\mathcal{S}_i|}. \tag{5}$$

Proof. The proof here mimics that of Lemma 4 in [25]. For completeness, we provide the proof. Assume for the

k-th sampling, the seleced sample set is denoted by $\mathcal{S}_k$. Then, we have

$$\begin{aligned}
\mathbb{E}\|\mathcal{V}_k - \mathcal{A}(\boldsymbol{x}_k)\|^2 =& \mathbb{E}\|\mathcal{A}_{\mathcal{S}_k}(\boldsymbol{x}_k) - \mathcal{A}_{\mathcal{S}_k}(\boldsymbol{x}_{k-1}) + \mathcal{V}_{k-1} - \mathcal{A}(\boldsymbol{x}_k)\|^2 \\
=& \mathbb{E}\|\mathcal{A}_{\mathcal{S}_k}(\boldsymbol{x}_k) - \mathcal{A}_{\mathcal{S}_k}(\boldsymbol{x}_{k-1}) - \mathcal{A}(\boldsymbol{x}_k) + \mathcal{A}(\boldsymbol{x}_{k-1}) + \mathcal{V}_{k-1} - \mathcal{A}(\boldsymbol{x}_{k-1})\|^2 \\
=& \mathbb{E}\|\mathcal{A}_{\mathcal{S}_k}(\boldsymbol{x}_k) - \mathcal{A}_{\mathcal{S}_k}(\boldsymbol{x}_{k-1}) - \mathcal{A}(\boldsymbol{x}_k) + \mathcal{A}(\boldsymbol{x}_{k-1})\|^2 + \mathbb{E}\|\mathcal{V}_{k-1} - \mathcal{A}(\boldsymbol{x}_{k-1})\|^2 \\
&+ \mathbb{E}\langle \mathcal{A}_{\mathcal{S}_k}(\boldsymbol{x}_k) - \mathcal{A}_{\mathcal{S}_k}(\boldsymbol{x}_{k-1}) - \mathcal{A}(\boldsymbol{x}_k) + \mathcal{A}(\boldsymbol{x}_{k-1}), \mathcal{V}_{k-1} - \mathcal{A}(\boldsymbol{x}_{k-1})\rangle \\
\overset{①}{=}& \mathbb{E}\|\mathcal{A}_{\mathcal{S}_k}(\boldsymbol{x}_k) - \mathcal{A}_{\mathcal{S}_k}(\boldsymbol{x}_{k-1}) - \mathcal{A}(\boldsymbol{x}_k) + \mathcal{A}(\boldsymbol{x}_{k-1})\|^2 + \mathbb{E}\|\mathcal{V}_{k-1} - \mathcal{A}(\boldsymbol{x}_{k-1})\|^2 \\
=& \frac{1}{|\mathcal{S}_k|}\mathbb{E}\|\mathcal{A}_i(\boldsymbol{x}_k) - \mathcal{A}_i(\boldsymbol{x}_{k-1}) - \mathcal{A}(\boldsymbol{x}_k) + \mathcal{A}(\boldsymbol{x}_{k-1})\|^2 + \mathbb{E}\|\mathcal{V}_{k-1} - \mathcal{A}(\boldsymbol{x}_{k-1})\|^2 \\
\overset{②}{\leq}& \frac{1}{|\mathcal{S}_k|}\mathbb{E}\|\mathcal{A}_i(\boldsymbol{x}_k) - \mathcal{A}_i(\boldsymbol{x}_{k-1})\|^2 + \mathbb{E}\|\mathcal{V}_{k-1} - \mathcal{A}(\boldsymbol{x}_{k-1})\|^2 \\
\overset{③}{\leq}& \frac{L^2}{|\mathcal{S}_k|}\|\mathrm{Exp}_{\boldsymbol{x}_k}^{-1}(\boldsymbol{x}_{k-1})\|^2 + \mathbb{E}\|\mathcal{V}_{k-1} - \mathcal{A}(\boldsymbol{x}_{k-1})\|^2 \\
\leq& \frac{L^2\rho_{k-1}^2}{|\mathcal{S}_k|} + \mathbb{E}\|\mathcal{V}_{k-1} - \mathcal{A}(\boldsymbol{x}_{k-1})\|^2,
\end{aligned}$$

where ① holds since $\mathbb{E}\langle \mathcal{A}_{\mathcal{S}_k}(\boldsymbol{x}_k) - \mathcal{A}_{\mathcal{S}_k}(\boldsymbol{x}_{k-1}) - \mathcal{A}(\boldsymbol{x}_k) + \mathcal{A}(\boldsymbol{x}_{k-1}), \mathcal{V}_{k-1} - \mathcal{A}(\boldsymbol{x}_{k-1})\rangle = \mathbf{0}$ in which the expectation is taken on the random set $\mathcal{S}_k$ ($\mathcal{V}_{k-1} - \mathcal{A}(\boldsymbol{x}_{k-1})$ is constant); ② holds due to $\mathbb{E}\|\boldsymbol{x} - \mathbb{E}(\boldsymbol{x})\|^2 \leq \mathbb{E}\|\boldsymbol{x}\|^2$; ③ holds since $f_i(\boldsymbol{x})$ is L-gradient Lipschitz, namely $\mathbb{E}_i\|\nabla f_i(\boldsymbol{x}) - \Gamma_{\boldsymbol{y}}^{\boldsymbol{x}}(\nabla f_i(\boldsymbol{y}))\|_2^2 \leq L\|\mathrm{Exp}_{\boldsymbol{x}}^{-1}(\boldsymbol{y})\|^2$. Notice, when $|\mathcal{S}_k| \geq n$, in ①, we have $\mathbb{E}\|\mathcal{A}_{\mathcal{S}_k}(\boldsymbol{x}_k) - \mathcal{A}_{\mathcal{S}_k}(\boldsymbol{x}_{k-1}) - \mathcal{A}(\boldsymbol{x}_k) + \mathcal{A}(\boldsymbol{x}_{k-1})\|^2 + \mathbb{E}\|\mathcal{V}_{k-1} - \mathcal{A}(\boldsymbol{x}_{k-1})\|^2 = 0$. In this case, we can obtain $\mathbb{E}\|\mathcal{V}_k - \mathcal{A}(\boldsymbol{x}_k)\|^2 = \mathbb{E}\|\mathcal{V}_{k-1} - \mathcal{A}(\boldsymbol{x}_{k-1})\|^2$. Therefore, consider these two cases and sum up $k = 0, 1, \cdots, t$, we have

$$\mathbb{E}\|\mathcal{V}(\boldsymbol{x}_t) - \mathcal{A}(\boldsymbol{x}_t)\|^2 \leq \mathbb{E}\|\mathcal{V}_0 - \mathcal{A}(\boldsymbol{x}_0)\|^2 + L^2\sum_{i=1}^{t}\mathbb{I}_{\{|\mathcal{S}_i|<n\}}\frac{\rho_{i-1}^2}{|\mathcal{S}_i|}.$$

The proof is completed. □

Lemma 4. *Suppose Assumptions 1 and 2 hold. Let $k_0 = \lfloor k/p \rfloor$ and $\widetilde{k}_0 = k_0 p$. Assume that for $k = \lfloor k/p \rfloor \cdot p$, we select $|\mathcal{S}_1|$ samples to estimate $\boldsymbol{v}_k$ and for $k \neq \lfloor k/p \rfloor \cdot p$, we select $|\mathcal{S}_{2,k}|$ samples to estimate $\boldsymbol{v}_k$. Then the estimation error between the full Riemannian gradient $\nabla f(\boldsymbol{x}_k)$ and its estimate $\boldsymbol{v}_k$ in Algorithm 1 is bounded as*

$$\mathbb{E}\left[\|\boldsymbol{v}_k - \nabla f(\boldsymbol{x}_k)\|^2 \mid \boldsymbol{x}_{\widetilde{k}_0}, \cdots, \boldsymbol{x}_{\widetilde{k}_0+p-1}\right] \leq \mathbb{I}_{\{|\mathcal{S}_1|<n\}}\frac{\sigma^2}{|\mathcal{S}_1|} + L^2\sum_{i=\widetilde{k}_0}^{\widetilde{k}_0+p-1}\mathbb{I}_{\{|\mathcal{S}_{2,i+1}|<n\}}\frac{d^2(\boldsymbol{x}_i, \boldsymbol{x}_{i+1})}{|\mathcal{S}_{2,i+1}|},$$

where $d(\boldsymbol{x}_i, \boldsymbol{x}_{i+1})$ is the distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i+1}$.

Proof. Here we construct an auxiliary sequence

$$\begin{aligned}
\widehat{\boldsymbol{v}}_t =& \sum_{i=1}^{t}\left(\mathrm{P}_{\boldsymbol{x}_i}^{\boldsymbol{x}_k}(\nabla f_{\mathcal{S}_2}(\boldsymbol{x}_i)) - \mathrm{P}_{\boldsymbol{x}_{i-1}}^{\boldsymbol{x}_k}(\nabla f_{\mathcal{S}_1}(\boldsymbol{x}_{i-1}))\right) + \mathrm{P}_{\boldsymbol{x}_0}^{\boldsymbol{x}_k}(\nabla f_{\mathcal{S}_1}(\boldsymbol{x}_0)) \\
=& \mathrm{P}_{\boldsymbol{x}_t}^{\boldsymbol{x}_k}(\nabla f_{\mathcal{S}_2}(\boldsymbol{x}_t)) - \mathrm{P}_{\boldsymbol{x}_{t-1}}^{\boldsymbol{x}_k}(\nabla f_{\mathcal{S}_2}(\boldsymbol{x}_{t-1})) + \widehat{\boldsymbol{v}}_{t-1},
\end{aligned}$$

where $\boldsymbol{x}_k$ is a given point and $\widehat{\boldsymbol{v}}_0 = \mathrm{P}_{\boldsymbol{x}_0}^{\boldsymbol{x}_k}(\nabla f_{\mathcal{S}_1}(\boldsymbol{x}_0))$. In this way, let $\mathcal{A}_{\mathcal{S}}(\boldsymbol{x}_t) = \mathrm{P}_{\boldsymbol{x}_t}^{\boldsymbol{x}_k}\left(\nabla f_{\mathcal{S}_2^t}(\boldsymbol{x}_t)\right)$. Then we have $\widehat{\boldsymbol{v}}_t = \mathcal{A}_{\mathcal{S}}(\boldsymbol{x}_t) - \mathcal{A}_{\mathcal{S}}(\boldsymbol{x}_{t-1}) + \widehat{\boldsymbol{v}}_{t-1}$. Accordingly, we can obtain

$$\begin{aligned}
\mathbb{E}_i\|\mathcal{A}_i(\boldsymbol{x}_t) - \mathcal{A}_i(\boldsymbol{x}_{t-1})\|_2^2 =& \mathbb{E}_i\left\|\mathrm{P}_{\boldsymbol{x}_t}^{\boldsymbol{x}_k}(\nabla f_i(\boldsymbol{x}_t)) - \mathrm{P}_{\boldsymbol{x}_{t-1}}^{\boldsymbol{x}_k}(\nabla f_i(\boldsymbol{x}_{t-1}))\right\|^2 \\
=& \mathbb{E}_i\left\|\mathrm{P}_{\boldsymbol{x}_{t-1}}^{\boldsymbol{x}_k}\left(\mathrm{P}_{\boldsymbol{x}_t}^{\boldsymbol{x}_{t-1}}(\nabla f_i(\boldsymbol{x}_t))\right) - \mathrm{P}_{\boldsymbol{x}_{t-1}}^{\boldsymbol{x}_k}(\nabla f_i(\boldsymbol{x}_{t-1}))\right\|^2 \\
=& \mathbb{E}_i\left\|\mathrm{P}_{\boldsymbol{x}_{t-1}}^{\boldsymbol{x}_k}\left(\mathrm{P}_{\boldsymbol{x}_t}^{\boldsymbol{x}_{t-1}}(\nabla f_i(\boldsymbol{x}_t)) - \nabla f_i(\boldsymbol{x}_{t-1})\right)\right\|^2 \\
\overset{①}{=}& \mathbb{E}_i\left\|\mathrm{P}_{\boldsymbol{x}_t}^{\boldsymbol{x}_{t-1}}(\nabla f_i(\boldsymbol{x}_t)) - \nabla f_i(\boldsymbol{x}_{t-1})\right\|^2 \\
\leq& L^2\|\mathrm{Exp}_{\boldsymbol{x}_{t-1}}^{-1}(\boldsymbol{x}_t)\|^2,
\end{aligned}$$

where ① holds as the parallel transport $\mathrm{P}_{\boldsymbol{x}}^{\boldsymbol{y}}$ preserves the norm. On the other hand, all $\mathcal{A}_i(\boldsymbol{x}_t)$ $(t=0,\cdots,k)$ are located in the tangent space at the point $\boldsymbol{x}_k$. Thus, Lemma 3 is applicable to the sequence $\widehat{\boldsymbol{v}}_t$.

Let $k_0=\lfloor K/p \rfloor$. For simplicity, we use $\mathcal{V}_0,\mathcal{V}_1,\cdots,\mathcal{V}_k$ to respectively denote $\mathcal{V}_{k_0},\mathcal{V}_{k_0+1},\cdots,\mathcal{V}_{k_0+k}$. For $\mathcal{V}_0$, we have $\mathcal{V}_0=\mathrm{P}_{\boldsymbol{x}_{k_0}}^{\boldsymbol{x}_k}\left(\nabla f_{\mathcal{S}_1}(\boldsymbol{x}_{k_0})\right)$. Then it yields

$$\begin{aligned}\mathbb{E}\|\mathcal{V}_0-\mathcal{A}(\boldsymbol{x}_0)\|^2=&\mathbb{E}\|\mathrm{P}_{\boldsymbol{x}_{k_0}}^{\boldsymbol{x}_k}\left(\nabla f_{\mathcal{S}_1}(\boldsymbol{x}_{k_0})\right)-\mathrm{P}_{\boldsymbol{x}_{k_0}}^{\boldsymbol{x}_k}\left(\nabla f(\boldsymbol{x}_{k_0})\right)\|^2\\=&\mathbb{E}\|\nabla f_{\mathcal{S}_1}(\boldsymbol{x}_{k_0})-\nabla f(\boldsymbol{x}_{k_0})\|^2\\=&\frac{1}{|\mathcal{S}_1|}\mathbb{E}\|\nabla f_i(\boldsymbol{x}_{k_0})-\nabla f(\boldsymbol{x}_{k_0})\|^2\\\overset{①}{\leq}&\frac{\sigma^2}{|\mathcal{S}_1|},\end{aligned}$$

where ① holds since the gradient variance is bounded in Assumption 2. On the other hand, since $\boldsymbol{x}_{k+1}=\mathrm{Exp}_{\boldsymbol{x}_k}\left(-\eta_k\frac{\boldsymbol{v}_k}{\|\boldsymbol{v}_k\|}\right)$, we have

$$\mathrm{d}^2\left(\boldsymbol{x}_{k+1},\boldsymbol{x}_k\right)=\left\|\mathrm{Exp}_{\boldsymbol{x}_k}^{-1}\left(\boldsymbol{x}_{k+1}\right)\right\|.$$

Therefore, we have

$$\mathbb{E}\|\widehat{\boldsymbol{v}}_t-\mathrm{P}_{\boldsymbol{x}_t}^{\boldsymbol{x}_k}\left(\nabla f(\boldsymbol{x}_t)\right)\|^2\leq L^2\sum_{i=0}^{t-1}\mathbb{I}_{\{|\mathcal{S}_{2,i+1}|<n\}}\frac{\mathrm{d}^2\left(\boldsymbol{x}_i,\boldsymbol{x}_{i+1}\right)}{|\mathcal{S}_{2,i+1}|}+\frac{\sigma^2}{|\mathcal{S}_1|}.$$

By setting $t=k$ and noting $t\leq p$ for each epoch, we establish

$$\mathbb{E}\|\boldsymbol{v}_k-\nabla f(\boldsymbol{x}_k)\|^2=\mathbb{E}\|\widehat{\boldsymbol{v}}_k-\mathrm{P}_{\boldsymbol{x}_k}^{\boldsymbol{x}_k}\left(\nabla f(\boldsymbol{x}_k)\right)\|^2\leq\frac{\sigma^2}{|\mathcal{S}_1|}+L^2\sum_{i=k_0}^{k_0+p-1}\mathbb{I}_{\{|\mathcal{S}_{2,i+1}|<n\}}\frac{\mathrm{d}^2\left(\boldsymbol{x}_i,\boldsymbol{x}_{i+1}\right)}{|\mathcal{S}_{2,i+1}|}.$$

Notice, when we sample all n samples, we have $\mathbb{E}\|\mathcal{V}_0-\mathcal{A}(x_0)\|^2=0$ and thus

$$\mathbb{E}\|\boldsymbol{v}_k-\nabla f(\boldsymbol{x}_k)\|^2\leq L^2\sum_{i=k_0}^{k_0+p-1}\mathbb{I}_{\{|\mathcal{S}_{2,i+1}|<n\}}\frac{\mathrm{d}^2\left(\boldsymbol{x}_i,\boldsymbol{x}_{i+1}\right)}{|\mathcal{S}_{2,i+1}|}.$$

So by combining the two case together, we can obtain the result in Lemma 1. The proof is completed. □

Now we are ready to prove Lemma 1.

Proof of Lemma 1. To prove Lemma 1, we can directly set $|\mathcal{S}_{2,k}|$ in Lemma 4 as $|\mathcal{S}_2|$ in Lemma 1 and obtain the results in Lemma 1. The proof is completed. □

B Proof of the Results in Section 3.2

B.1 Proof of Theorem 1

Proof. For brevity, let $\widetilde{\eta}_k=\frac{\eta_k}{\|\boldsymbol{v}_k\|}$. Then by using the L-gradient Lipschitz, we have

$$\begin{aligned}f(\boldsymbol{x}_{k+1})\leq&f(\boldsymbol{x}_k)+\langle\nabla f(\boldsymbol{x}_k),\mathrm{Exp}_{\boldsymbol{x}_k}^{-1}\left(\boldsymbol{x}_{k+1}\right)\rangle+\frac{L}{2}\|\mathrm{Exp}_{\boldsymbol{x}_k}^{-1}\left(\boldsymbol{x}_{k+1}\right)\|^2\\\leq&f(\boldsymbol{x}_k)-\widetilde{\eta}_k\langle\nabla f(\boldsymbol{x}_k),\boldsymbol{v}_k\rangle+\frac{\widetilde{\eta}_k^2L}{2}\|\boldsymbol{v}_k\|^2\\\leq&f(\boldsymbol{x}_k)-\widetilde{\eta}_k\langle\nabla f(\boldsymbol{x}_k)-\boldsymbol{v}_k,\boldsymbol{v}_k\rangle-\widetilde{\eta}_k\left(1-\frac{\widetilde{\eta}_kL}{2}\right)\|\boldsymbol{v}_k\|^2\\\leq&f(\boldsymbol{x}_k)-\widetilde{\eta}_k\langle\nabla f(\boldsymbol{x}_k)-\boldsymbol{v}_k,\boldsymbol{v}_k\rangle-\widetilde{\eta}_k\left(1-\frac{\widetilde{\eta}_kL}{2}\right)\|\boldsymbol{v}_k\|^2\\\leq&f(\boldsymbol{x}_k)+\frac{\widetilde{\eta}_k}{2}\|\nabla f(\boldsymbol{x}_k)-\boldsymbol{v}_k\|^2-\frac{\widetilde{\eta}_k}{2}\left(1-\widetilde{\eta}_kL\right)\|\boldsymbol{v}_k\|^2.\end{aligned}\tag{6}$$

Since we have $\boldsymbol{x}_{k+1} = \mathrm{Exp}_{\boldsymbol{x}_k}^{-1}\left(-\eta_k \frac{\boldsymbol{v}_k}{\|\boldsymbol{v}_k\|}\right)$, we can obtain

$$\mathrm{d}\left(\boldsymbol{x}_{k+1}, \boldsymbol{x}_k\right) = \eta_k = \min\left(\frac{\epsilon}{2Ln_0}, \frac{\|\boldsymbol{v}_k\|}{4Ln_0}\right) \leq \frac{\epsilon}{2Ln_0}. \tag{7}$$

Now we consider the two cases: (1) k is not an integer multiple of p; (2) k is an integer multiple of p. We can consider case (1) as follows. If $s = n$, then by Lemma 1 and Eqn. (7), we have

$$\mathbb{E}\|\boldsymbol{v}_k - \nabla f(\boldsymbol{x}_k)\|^2 \leq \frac{L^2}{|\mathcal{S}_2|}\sum_{i=k_0}^{k_0+p-1}\mathrm{d}^2\left(\boldsymbol{x}_i, \boldsymbol{x}_{i+1}\right) \leq \frac{pL^2}{|\mathcal{S}_2|}\frac{\epsilon^2}{4L^2n_0^2} = n_0 s^{\frac{1}{2}} L^2 \frac{n_0}{4s^{\frac{1}{2}}}\frac{\epsilon^2}{4L^2n_0^2} = \frac{1}{16}\epsilon^2.$$

If $s = \frac{16\sigma^2}{\epsilon^2}$, then Lemma 1 gives

$$\mathbb{E}\|\boldsymbol{v}_k - \nabla f(\boldsymbol{x}_k)\|^2 \leq \frac{pL^2}{|\mathcal{S}_2|}\frac{\epsilon^2}{4L^2n_0^2} + \frac{\sigma^2}{|\mathcal{S}_1|} = n_0 s^{\frac{1}{2}} L^2 \frac{\epsilon^2}{4L^2n_0^2}\frac{n_0}{4s^{\frac{1}{2}}} + \frac{\epsilon^2}{16} = \frac{1}{8}\epsilon^2. \tag{8}$$

For case (2), namely when k is an integer multiple of p, we have $\mathbb{E}\|\boldsymbol{v}_k - \nabla f(\boldsymbol{x}_k)\|^2 \leq \frac{pL^2\eta_k^2}{|\mathcal{S}_2|} + \frac{\sigma^2}{|\mathcal{S}_1|} = 0 + \frac{\epsilon^2}{16} \leq \frac{1}{8}\epsilon^2$. At the same time, since $\eta_k = \min\left(\frac{\epsilon}{2Ln_0}, \frac{\|\boldsymbol{v}_k\|}{4Ln_0}\right)$, we have $\widetilde{\eta}_k = \frac{\eta_k}{\|\boldsymbol{v}_k\|} = \min\left(\frac{\epsilon}{2Ln_0\|\boldsymbol{v}_k\|}, \frac{1}{4Ln_0}\right) \leq \frac{1}{4Ln_0}$ and

$$\widetilde{\eta}_k(1-\widetilde{\eta}_k L)\|\boldsymbol{v}_k\|^2 \geq \frac{3\widetilde{\eta}_k}{4}\|\boldsymbol{v}_k\|^2 = \frac{3}{8}\min\left(\frac{\epsilon}{Ln_0\|\boldsymbol{v}_k\|}, \frac{1}{2Ln_0}\right)\|\boldsymbol{v}_k\|^2 = \frac{3\epsilon^2}{16Ln_0}\min\left(\frac{2\|\boldsymbol{v}_k\|}{\epsilon}, \frac{\|\boldsymbol{v}_k\|^2}{\epsilon^2}\right) \overset{①}{\geq} \frac{3\epsilon(2\|\boldsymbol{v}_k\| - \epsilon)}{16Ln_0},$$

where ① uses $x^2 \geq 2|x| - 1$ for $\forall x$. So by taking expectation, we have

$$\mathbb{E}\left[f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k)\right] \leq \frac{1}{2}\frac{1}{4Ln_0}\frac{\epsilon^2}{8} - \frac{1}{2}\frac{3\epsilon(2\|\boldsymbol{v}_k\| - \epsilon)}{16Ln_0} = -\frac{\epsilon}{64Ln_0}\left(12\mathbb{E}\|\boldsymbol{v}_k\| - 7\epsilon\right).$$

In this way, we have

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\boldsymbol{v}_k\| \leq \frac{7\epsilon}{64} + \frac{16Ln_0}{3K\epsilon}\mathbb{E}\left[f(\boldsymbol{x}_0) - f(\boldsymbol{x}_K)\right] \leq \frac{7\epsilon}{64} + \frac{16Ln_0\Delta}{3K\epsilon},$$

where we use $\mathbb{E}\left[f(\boldsymbol{x}_0) - f(\boldsymbol{x}_K)\right] \leq \mathbb{E}\left[f(\boldsymbol{x}_0) - f(\boldsymbol{x}_*)\right] \leq f(\boldsymbol{x}_0) - f(\boldsymbol{x}_*) \leq \Delta$. It means that after running at most $K = \frac{14Ln_0\Delta}{\epsilon^2}$ iterations, the algorithm will terminate, since

$$\mathbb{E}\|\nabla f(\widetilde{\boldsymbol{x}})\| = \frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\nabla f(\boldsymbol{x}_k)\| \leq \frac{1}{K}\sum_{k=0}^{K-1}\left[\mathbb{E}\|\nabla f(\boldsymbol{x}_k) - \boldsymbol{v}_k\| + \mathbb{E}\|\boldsymbol{v}_k\|\right] \overset{①}{\leq} \frac{1}{K}\sum_{k=0}^{K-1}\sqrt{\mathbb{E}\|\nabla f(\boldsymbol{x}_k) - \boldsymbol{v}_k\|^2} + \frac{\epsilon}{2} \overset{②}{\leq} \epsilon,$$

where ① uses the Jensen's inequality; ② holds since $\mathbb{E}\|\nabla f(\boldsymbol{x}) - \boldsymbol{v}_k\|^2 \leq \frac{\epsilon^2}{8}$ in Eqn. (8). The proof is completed. The proof is completed. □

B.2 Proof of Corollary 1

Proof. According to Theorem 1, we know that after running at most $K = \frac{14Ln_0\Delta}{\epsilon^2}$ iterations, the algorithm will terminate. In this way, we can compute the stochastic gradient complexity as

$$\mathcal{O}\left(\frac{K}{p}|\mathcal{S}_1| + K|\mathcal{S}_2|\right) = \mathcal{O}\left(\frac{Ln_0\Delta}{\epsilon^2}\left(s\frac{1}{n_0 s^{1/2}} + \frac{s^{1/2}}{2n_0}\right)\right) = \mathcal{O}\left(\min\left(n + \frac{L\Delta\sqrt{n}}{\epsilon^2}, \frac{L\Delta\sigma}{\epsilon^3}\right)\right).$$

The proof is completed. □

B.3 Proof of Theorem 2

Proof. For brevity, let $\widetilde{\eta}_k = \frac{\eta_k}{\|\boldsymbol{v}_k\|}$. From Eqn. (6), we can obtain the following inequality:

$$f(\boldsymbol{x}_{k+1}) \leq f(\boldsymbol{x}_k) + \frac{\widetilde{\eta}_k}{2}\|\nabla f(\boldsymbol{x}_k) - \boldsymbol{v}_k\|^2 - \frac{\widetilde{\eta}_k}{2}\left(1 - \widetilde{\eta}_k L\right)\|\boldsymbol{v}_k\|^2. \tag{9}$$

Now we consider the two cases: (1) k is not an integer multiple of p; (2) k is an integer multiple of p. We can consider case (1) as follows. By setting $p = \frac{\sigma n_0}{\epsilon}$, $\eta_k = \min\left(\frac{\epsilon}{2Ln_0}, \frac{\|\boldsymbol{v}_k\|}{4Ln_0}\right)$, $|\mathcal{S}_1| = \frac{16\sigma^2}{\epsilon^2}$, $\mathcal{S}_2 = \frac{\sigma}{2\epsilon n_0}$, where $n_0 \in [1, 2\sigma/\epsilon]$, Lemma 1 gives

$$\mathbb{E}\|\boldsymbol{v}_k - \nabla f(\boldsymbol{x}_k)\|^2 \leq \frac{pL^2\eta_k^2}{|\mathcal{S}_2|} + \frac{\sigma^2}{|\mathcal{S}_1|} = \frac{\sigma n_0}{\epsilon}L^2\frac{\epsilon^2}{4L^2n_0^2}\frac{\epsilon n_0}{4\sigma} + \frac{\epsilon^2}{16} = \frac{1}{8}\epsilon^2. \tag{10}$$

For case (2), namely when k is an integer multiple of p, we have $\mathbb{E}\|\boldsymbol{v}_k - \nabla f(\boldsymbol{x}_k)\|^2 \leq \frac{pL^2\eta^2}{|\mathcal{S}_2|} + \frac{\sigma^2}{|\mathcal{S}_1|} = 0 + \frac{\epsilon^2}{16} \leq \frac{1}{8}\epsilon^2$. Then similar to proof in Sec. B.1, since $\eta_k = \min\left(\frac{\epsilon}{2Ln_0}, \frac{\|\boldsymbol{v}_k\|}{4Ln_0}\right)$, we have $\widetilde{\eta}_k = \frac{\eta_k}{\|\boldsymbol{v}_k\|} = \min\left(\frac{\epsilon}{2Ln_0\|\boldsymbol{v}_k\|}, \frac{1}{4Ln_0}\right) \leq \frac{1}{4Ln_0}$ and

$$\widetilde{\eta}_k(1 - \widetilde{\eta}_k L)\|\boldsymbol{v}_k\|^2 \geq \frac{3\widetilde{\eta}_k}{4}\|\boldsymbol{v}_k\|^2 = \frac{3}{8}\min\left(\frac{\epsilon}{Ln_0\|\boldsymbol{v}_k\|}, \frac{1}{2Ln_0}\right)\|\boldsymbol{v}_k\|^2 = \frac{3\epsilon^2}{16Ln_0}\min\left(\frac{2\|\boldsymbol{v}_k\|}{\epsilon}, \frac{\|\boldsymbol{v}_k\|^2}{\epsilon^2}\right) \overset{①}{\geq} \frac{3\epsilon(2\|\boldsymbol{v}_k\| - \epsilon)}{16Ln_0},$$

where ① uses $x^2 \geq 2|x| - 1$ for $\forall x$.

So by taking expectation, we have

$$\mathbb{E}\left[f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k)\right] \leq \frac{1}{2}\frac{1}{4Ln_0}\frac{\epsilon^2}{8} - \frac{1}{2}\frac{3\epsilon(2\|\boldsymbol{v}_k\| - \epsilon)}{16Ln_0} = -\frac{\epsilon}{64Ln_0}\left(12\mathbb{E}\|\boldsymbol{v}_k\| - 7\epsilon\right).$$

In this way, we have

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\boldsymbol{v}_k\| \leq \frac{7\epsilon}{64} + \frac{16Ln_0}{3K\epsilon}\mathbb{E}\left[f(\boldsymbol{x}_0) - f(\boldsymbol{x}_K)\right] \leq \frac{7\epsilon}{64} + \frac{16Ln_0\Delta}{3K\epsilon},$$

where we use $\mathbb{E}\left[f(\boldsymbol{x}_0) - f(\boldsymbol{x}_K)\right] \leq \mathbb{E}\left[f(\boldsymbol{x}_0) - f(\boldsymbol{x}_*)\right] \leq f(\boldsymbol{x}_0) - f(\boldsymbol{x}_*) \leq \Delta$. It means that after running at most $K = \frac{14Ln_0\Delta}{\epsilon^2}$ iterations, the algorithm will terminate, since

$$\mathbb{E}\|\nabla f(\widetilde{\boldsymbol{x}})\| = \frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\nabla f(\boldsymbol{x}_k)\| \leq \frac{1}{K}\sum_{k=0}^{K-1}\left[\mathbb{E}\|\nabla f(\boldsymbol{x}_k) - \boldsymbol{v}_k\| + \mathbb{E}\|\boldsymbol{v}_k\|\right] \overset{①}{\leq} \frac{1}{K}\sum_{k=0}^{K-1}\sqrt{\mathbb{E}\|\nabla f(\boldsymbol{x}_k) - \boldsymbol{v}_k\|^2} + \frac{\epsilon}{2} \overset{②}{\leq} \epsilon,$$

where ① uses the Jensen's inequality; ② holds since $\mathbb{E}\|\nabla f(\boldsymbol{x}) - \boldsymbol{v}_k\|^2 \leq \frac{\epsilon^2}{8}$ in Eqn. (8). The proof is completed. The proof is completed. □

B.4 Proof of Corollary 2

Proof. We adopt similar proof sketch of Corollary 1. According to Theorem 2, we know that after running at most $K = \frac{14Ln_0\Delta}{\epsilon^2}$ iterations, the algorithm will terminate. In this way, we can compute the stochastic gradient complexity as

$$\mathcal{O}\left(\frac{K}{p}|\mathcal{S}_1| + K|\mathcal{S}_2|\right) = \mathcal{O}\left(\frac{Ln_0\Delta}{\epsilon^2}\left(\frac{\sigma^2}{\epsilon^2}\frac{\epsilon}{\sigma n_0} + \frac{\sigma}{\epsilon n_0}\right)\right) = \mathcal{O}\left(\frac{L\sigma\Delta}{\epsilon^3}\right).$$

The proof is completed. □

C Proofs of the Results in Section 3.3

Before proving Theorems 3 and 4, we first prove Lemma 5 which is a key lemma to prove Theorems 3 and 4.

Lemma 5. *Assume function $f(\boldsymbol{x})$ is τ-gradient dominated. Let $\mathcal{E}$ denotes the event:*

$$\mathcal{E}=\left\{\mathbb{E}\|\nabla f(\widetilde{\boldsymbol{x}})\|^2\leq\epsilon^2 \quad \textit{and} \quad \mathbb{E}\left[f(\widetilde{\boldsymbol{x}})-f(\boldsymbol{x}_*)\right]\leq\tau\epsilon^2.\right\}$$

(1) For online-setting, we have $p=\frac{\sigma n_0}{\epsilon}$, $\eta_k=\frac{\|\boldsymbol{v}_k\|}{2Ln_0}$, $|\mathcal{S}_1|=\frac{32\sigma^2}{\epsilon^2}$, $|\mathcal{S}_{2,k}|=\frac{8\sigma\|\boldsymbol{v}_{k-1}\|^2}{\epsilon^3 n_0}$. To let the event $\mathcal{E}$ happen, Algorithm 1 runs at most $K=\frac{64Ln_0\Delta}{\epsilon^2}$ iterations and the IFO complexity is

$$\mathcal{O}\left(\frac{L\Delta\sigma}{\epsilon^3}\right), \quad \textit{where } \widetilde{\Delta}=f(\boldsymbol{x}_0)-f(\boldsymbol{x}_*).$$

(2) For finite-sum setting, we let $s=\min\left(n,\frac{32\sigma^2}{\epsilon^2}\right)$, $p=n_0 s^{\frac{1}{2}}$, $\eta_k=\frac{\|\boldsymbol{v}_k\|}{2Ln_0}$, $|\mathcal{S}_1|=s$, $|\mathcal{S}_{2,k}|=\min\left(\frac{8p\|\boldsymbol{v}_{k-1}\|^2}{n_0^2\epsilon^2},n\right)$. To let the event $\mathcal{E}$ happen, Algorithm 1 runs at most $K=\frac{64Ln_0\Delta}{\epsilon^2}$ iterations and the IFO complexity is

$$\mathcal{O}\left(\min\left(n+\frac{L\Delta\sqrt{n}}{\epsilon^2},\frac{L\Delta\sigma}{\epsilon^3}\right)\right), \quad \textit{where } \Delta=f(\boldsymbol{x}_0)-f(\boldsymbol{x}_*).$$

Proof. For brevity, let $\widetilde{\eta}_k=\frac{\eta_k}{\|\boldsymbol{v}_k\|}=\frac{1}{2Ln_0}$. Then similar to Eqn. (6), by using the L-gradient Lipschitz, we have

$$\begin{aligned}
f(\boldsymbol{x}_{k+1})\leq& f(\boldsymbol{x}_k)+\langle\nabla f(\boldsymbol{x}_k),\mathrm{Exp}_{\boldsymbol{x}_k}^{-1}(\boldsymbol{x}_{k+1})\rangle+\frac{L}{2}\|\mathrm{Exp}_{\boldsymbol{x}_k}^{-1}(\boldsymbol{x}_{k+1})\|^2\\
\leq& f(\boldsymbol{x}_k)-\widetilde{\eta}_k\langle\nabla f(\boldsymbol{x}_k),\boldsymbol{v}_k\rangle+\frac{\widetilde{\eta}_k^2L}{2}\|\boldsymbol{v}_k\|^2\\
\leq& f(\boldsymbol{x}_k)-\widetilde{\eta}_k\langle\nabla f(\boldsymbol{x}_k)-\boldsymbol{v}_k,\boldsymbol{v}_k\rangle-\widetilde{\eta}_k\left(1-\frac{\widetilde{\eta}_kL}{2}\right)\|\boldsymbol{v}_k\|^2\\
\leq& f(\boldsymbol{x}_k)-\widetilde{\eta}_k\langle\nabla f(\boldsymbol{x}_k)-\boldsymbol{v}_k,\boldsymbol{v}_k\rangle-\widetilde{\eta}_k\left(1-\frac{\widetilde{\eta}_kL}{2}\right)\|\boldsymbol{v}_k\|^2\\
\leq& f(\boldsymbol{x}_k)+\frac{\widetilde{\eta}_k}{2}\|\nabla f(\boldsymbol{x}_k)-\boldsymbol{v}_k\|^2-\frac{\widetilde{\eta}_k}{2}\left(1-\widetilde{\eta}_kL\right)\|\boldsymbol{v}_k\|^2\\
\overset{①}{\leq}& f(\boldsymbol{x}_k)+\frac{1}{4Ln_0}\|\nabla f(\boldsymbol{x}_k)-\boldsymbol{v}_k\|^2-\frac{1}{8Ln_0}\|\boldsymbol{v}_k\|^2,
\end{aligned}$$

where ① holds since $n_0\geq 1$. By summing up this equation from 0 to $K-1$ and taking expectation, we can obtain

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\boldsymbol{v}_k\|^2\leq\frac{2}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\boldsymbol{v}_k-\nabla f(\boldsymbol{x}_k)\|^2+\frac{8Ln_0}{K}\left[f(\boldsymbol{x}_0)-f(\boldsymbol{x}_K)\right]\overset{①}{\leq}\frac{2}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\boldsymbol{v}_k-\nabla f(\boldsymbol{x}_k)\|^2+\frac{8Ln_0\Delta}{K},$$

where ① uses $\mathbb{E}\left[f(\boldsymbol{x}_0)-f(\boldsymbol{x}_K)\right]\leq\mathbb{E}\left[f(\boldsymbol{x}_0)-f(\boldsymbol{x}_*)\right]\leq f(\boldsymbol{x}_0)-f(\boldsymbol{x}_*)\leq\Delta$.

Now we use Lemma 4 to bound each $\mathbb{E}\|\boldsymbol{v}_k-\nabla f(\boldsymbol{x}_k)\|^2$ for both online and finite-sum setting. For online-setting, we have $p=\frac{\sigma n_0}{\epsilon}$, $\eta_k=\frac{\|\boldsymbol{v}_k\|}{2Ln_0}$, $|\mathcal{S}_1|=\frac{32\sigma^2}{\epsilon^2}$, $|\mathcal{S}_{2,k}|=\frac{8\sigma\|\boldsymbol{v}_{k-1}\|^2}{\epsilon^3 n_0}$. From Lemma 4, we can establish

$$\mathbb{E}\|\boldsymbol{v}_k-\nabla f(\boldsymbol{x}_k)\|^2\leq\mathbb{I}_{\{|\mathcal{S}_1|<n\}}\frac{\sigma^2}{|\mathcal{S}_1|}+L^2\sum_{i=k_0}^{k_0+p-1}\mathbb{I}_{\{|\mathcal{S}_{2,i+1}|<n\}}\frac{\mathrm{d}^2(\boldsymbol{x}_i,\boldsymbol{x}_{i+1})}{|\mathcal{S}_{2,i+1}|}\leq\sigma^2\frac{\epsilon^2}{32\sigma^2}+L^2\sum_{i=k_0}^{k_0+p-1}\frac{\|\boldsymbol{v}_i\|^2}{4L^2n_0^2}\frac{\epsilon^3n_0}{8\sigma\|\boldsymbol{v}_i\|^2}\leq\frac{\epsilon^2}{16},$$

where we use $\mathrm{d}^2(\boldsymbol{x}_{k+1},\boldsymbol{x}_k)=\left\|\mathrm{Exp}_{\boldsymbol{x}_k}^{-1}(\boldsymbol{x}_{k+1})\right\|^2=\eta_k^2$ since $\boldsymbol{x}_{k+1}=\mathrm{Exp}_{\boldsymbol{x}_k}\left(-\eta_k\frac{\boldsymbol{v}_k}{\|\boldsymbol{v}_k\|}\right)$. For finite-sum setting, we let $s=\min\left(n,\frac{32\sigma^2}{\epsilon^2}\right)$, $p=n_0s^{\frac{1}{2}}$, $\eta_k=\frac{\|\boldsymbol{v}_k\|}{2Ln_0}$, $|\mathcal{S}_1|=s$, $|\mathcal{S}_{2,k}|=\min\left(\frac{8p\|\boldsymbol{v}_{k-1}\|^2}{n_0^2\epsilon^2},n\right)$. In this case, we also have

$$\mathbb{E}\|\boldsymbol{v}_k-\nabla f(\boldsymbol{x}_k)\|^2\leq\mathbb{I}_{\{|\mathcal{S}_1|<n\}}\frac{\sigma^2}{|\mathcal{S}_1|}+L^2\sum_{i=k_0}^{k_0+p-1}\mathbb{I}_{\{|\mathcal{S}_{2,i+1}|<n\}}\frac{\mathrm{d}^2(\boldsymbol{x}_i,\boldsymbol{x}_{i+1})}{|\mathcal{S}_{2,i+1}|}\leq\sigma^2\frac{\epsilon^2}{32\sigma^2}+L^2\sum_{i=k_0}^{k_0+p-1}\frac{\|\boldsymbol{v}_i\|^2}{4L^2n_0^2}\frac{n_0^2\epsilon^2}{8p\|\boldsymbol{v}_i\|^2}\leq\frac{\epsilon^2}{16}.$$

Meanwhile, we set $K=\frac{64Ln_0\Delta}{\epsilon^2}$, which gives

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\boldsymbol{v}_k\|^2\leq\frac{2}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\boldsymbol{v}_k-\nabla f(\boldsymbol{x}_k)\|^2+\frac{8Ln_0\Delta}{K}\leq\frac{\epsilon^2}{4}.$$

It means that after running at most $K = \frac{14Ln_0\Delta}{\epsilon^2}$ iterations, the algorithm will terminate, since

$$\mathbb{E}\|\nabla f(\widetilde{\boldsymbol{x}})\|^2 = \frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\nabla f(\boldsymbol{x}_k)\|^2 \leq \frac{1}{K}\sum_{k=0}^{K-1}\left[2\mathbb{E}\|\nabla f(\boldsymbol{x}_k) - \boldsymbol{v}_k\|^2 + 2\mathbb{E}\|\boldsymbol{v}_k\|^2\right] \leq \epsilon^2.$$

Then we use the definition of τ-gradient dominated function, we have

$$\mathbb{E}[f(\widetilde{\boldsymbol{x}}) - f(\boldsymbol{x}_*)] = \frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}[f(\boldsymbol{x}_k) - f(\boldsymbol{x}_*)] \leq \frac{\tau}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\nabla f(\boldsymbol{x}_k)\|^2 = \tau\mathbb{E}\|\nabla f(\widetilde{\boldsymbol{x}})\|^2 \leq \tau\epsilon^2.$$

Now consider the IFO complexity for both online and finite-sum settings. For online setting, its IFO complexity is

$$\mathcal{O}\left(\frac{K}{p}|\mathcal{S}_1| + \sum_{k=0}^{K-1}\mathbb{E}|\mathcal{S}_{2,k}|\right) = \mathcal{O}\left(\frac{L\Delta\sigma}{\epsilon^3} + \frac{\sigma}{n_0\epsilon^3}\sum_{k=0}^{K-1}\mathbb{E}\|\boldsymbol{v}_k\|^2\right) \leq \mathcal{O}\left(\frac{L\Delta\sigma}{\epsilon^3} + \frac{\sigma}{n_0\epsilon^3}K\cdot\frac{\epsilon^2}{4}\right) = \mathcal{O}\left(\frac{L\Delta\sigma}{\epsilon^3}\right).$$

similarly, we can compute the expectation IFO complexity for finite-sum setting:

$$\mathcal{O}\left(\frac{K}{p}|\mathcal{S}_1| + \sum_{k=0}^{K-1}\mathbb{E}|\mathcal{S}_{2,k}|\right) = \mathcal{O}\left(\min\left(n + \frac{L\Delta\sqrt{n}}{\epsilon^2}, \frac{L\Delta\sigma}{\epsilon^3}\right)\right).$$

The proof is completed. □

C.1 Proof of Theorems 3

Now we are ready to prove Theorem 3.

Proof. We first consider the t iteration in Algorithm 2. By Lemma 5, we obtain that by using ϵ_{t-1} with proper other parameters, the IFO complexity of Algorithm 1 for computing $\mathbb{E}[\|\nabla f(\widetilde{\boldsymbol{x}}_t)\|^2] \leq \epsilon_{t-1}^2$ is

$$\mathcal{O}\big(\min\big(n + \frac{L\Delta_t\sqrt{n}}{\epsilon_{t-1}^2}, \frac{L\Delta_t\sigma}{\epsilon_{t-1}^3}\big)\big),$$

when the parameters satisfy $s_t = \min\big(n, \frac{32\sigma^2}{\epsilon_{t-1}^2}\big)$, $p^t = n_0^t s_t^{\frac{1}{2}}$, $\eta_k^t = \frac{\|\boldsymbol{v}_k^t\|}{2Ln_0}$, $|\mathcal{S}_1^t| = s_t$, $|\mathcal{S}_{2,k}^t| = \min(\frac{8p^t\|\boldsymbol{v}_{k-1}^t\|^2}{(n_0^t)^2\epsilon_{t-1}^2}, n)$ and $K^t = \frac{64Ln_0^t\Delta^t}{\epsilon_{t-1}^2}$. Then the initial point $\boldsymbol{x}_0$ at the t iteration is the output $\widetilde{\boldsymbol{x}}_{t-1}$ of the $(t-1)$-th iteration, which gives the distance $\Delta_t = \mathbb{E}[f(\boldsymbol{x}_0) - f(\boldsymbol{x}_*)] = \mathbb{E}[f(\widetilde{\boldsymbol{x}}_{t-1}) - f(\boldsymbol{x}_*)] \leq \tau\epsilon_{t-2}^2$ by using Lemma 5. On the other hand, $\epsilon_t = \frac{\epsilon_0}{2^t}$. So the IFO complexity of the t-th iteration is

$$\mathcal{O}\big(\min\big(n + \frac{L\Delta_t\sqrt{n}}{\epsilon_{t-1}^2}, \frac{L\Delta_t\sigma}{\epsilon_{t-1}^3}\big)\big) = \mathcal{O}\big(\min\big(n + \frac{L\tau\epsilon_{t-2}^2\sqrt{n}}{\epsilon_{t-1}^2}, \frac{L\sigma\tau\epsilon_{t-2}^2}{\epsilon_{t-1}^3}\big)\big) = \mathcal{O}\left(\min\left(n + \tau L\sqrt{n}, \frac{\tau L\sigma}{\epsilon_{t-1}}\right)\right).$$

So to achieve $\epsilon_T \leq \frac{\epsilon_0}{2^T} \leq \epsilon$, T satisfies $T \geq \log\left(\frac{\epsilon_0}{\epsilon}\right)$. So for the T iterations, the total complexity is

$$\mathcal{O}\left(\min\left(\left(n + \tau L\sqrt{n}\right)\log\left(\frac{1}{\epsilon}\right), \tau L\sigma\sum_{t=1}^{T}\frac{1}{\epsilon_{t-1}}\right)\right) = \mathcal{O}\left(\min\left(\left(n + \tau L\sqrt{n}\right)\log\left(\frac{1}{\epsilon}\right), \frac{\tau L\sigma}{\epsilon}\right)\right).$$

Meanwhile, we can obtain

$$\mathbb{E}\|\nabla f(\widetilde{\boldsymbol{x}}_t)\| \leq \sqrt{\mathbb{E}\|\nabla f(\widetilde{\boldsymbol{x}}_t)\|^2} \leq \epsilon_{t-1} = \frac{\epsilon_0}{2^{t-1}} = \frac{1}{2^t}\sqrt{\frac{\Delta}{\tau}} \quad \text{and} \quad \mathbb{E}\left[f(\widetilde{\boldsymbol{x}}_t) - f(\boldsymbol{x}_*)\right] \leq \tau\epsilon_{t-1}^2 = \frac{\tau\epsilon_0^2}{4^{t-1}} = \frac{\Delta}{4^t},$$

where we set $\epsilon_0 = \frac{1}{2}\sqrt{\frac{\Delta}{\tau}}$. The proof is completed. □

C.2 Proof of Theorem 4

Proof. The proof here is very similar to the strategy in Section C.1 for proving Theorem 3. The main idea is to use the result in Lemma 5, to achieve

$$\mathbb{E}\|\nabla f(\widetilde{\boldsymbol{x}})\|^2 \leq \epsilon^2 \quad \text{and} \quad \mathbb{E}\left[f(\widetilde{\boldsymbol{x}}) - f(\boldsymbol{x}_*)\right] \leq \tau\epsilon^2,$$

the IFO complexity is

$$\mathcal{O}\left(\frac{L\Delta\sigma}{\epsilon^3}\right), \quad \text{where } \widetilde{\Delta} = f(\boldsymbol{x}_0) - f(\boldsymbol{x}_*).$$

Then following the proof in Section C.1 for proving Theorem 3, we can obtain the IFO complexity for achieving $\mathbb{E}\|\nabla f(\widetilde{\boldsymbol{x}}_t)\|^2 \leq \epsilon_{t-1}^2$:

$$\mathcal{O}\left(\frac{\tau L\sigma}{\epsilon}\right),$$

when the parameters obey $p_t = \frac{\sigma n_0^t}{\epsilon_{t-1}}$, $\eta_k^t = \frac{\|\boldsymbol{v}_k^t\|}{2Ln_0^t}$, $|\mathcal{S}_1^t| = \frac{32\sigma^2}{\epsilon_{t-1}^2}$, $|\mathcal{S}_{2,k}^t| = \frac{8\sigma\|\boldsymbol{v}_{k-1}^t\|^2}{\epsilon_{t-1}^3 n_0^t}$, and $K^t = \frac{64Ln_0^t\Delta^t}{\epsilon_{t-1}^2}$.

Meanwhile, we can obtain

$$\mathbb{E}\|\nabla f(\widetilde{\boldsymbol{x}}_t)\| \leq \sqrt{\mathbb{E}\|\nabla f(\widetilde{\boldsymbol{x}}_t)\|^2} \leq \epsilon_{t-1} = \frac{\epsilon_0}{2^{t-1}} = \frac{1}{2^t}\sqrt{\frac{\Delta}{\tau}} \quad \text{and} \quad \mathbb{E}\left[f(\widetilde{\boldsymbol{x}}_t) - f(\boldsymbol{x}_*)\right] \leq \tau\epsilon_{t-1}^2 = \frac{\tau\epsilon_0^2}{4^{t-1}} = \frac{\Delta}{4^t},$$

where we set $\epsilon_0 = \frac{1}{2}\sqrt{\frac{\Delta}{\tau}}$. The proof is completed. □

D More Experimental Results

D.1 Descriptions of Testing Datasets

We first briefly introduce the ten testing datasets in the manuscript. Among them, there are six datasets, including a9a, satimage, covtype, protein, ijcnn1 and epsilon, that are provided in the LibSVM website[1]. We also evaluate our algorithms on the three datasets: YaleB [37], AR [38] and PIE [39], which are very commonly used face classification datasets. Finally, we also test those algorithms on a movie recommendation dataset, namely MovieLens-1M[2]. Their detailed information is summarized in Table 2. From it we can observe that these datasets are different from each other due to their feature dimension, training samples, and class numbers, *etc.*

Table 2: Descriptions of the ten testing datasets.

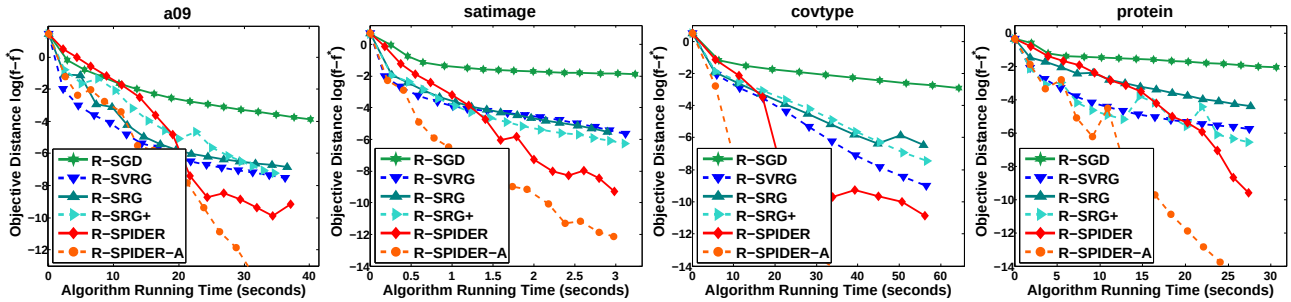
	#class	#sample	#feature		#class	#sample	#feature
a9a	2	32,561	123	epsilon	2	40,000	2000
satimage	6	4,435	36	YaleB	38	2,414	2,016
covtype	2	581,012	54	AR	100	2,600	1,200
protein	3	14,895	357	PIE	64	11,554	1,024
ijcnn1	2	49,990	22	MovieLens-1M	—	6,040	3,706

D.2 Comparison of Algorithm Running Time

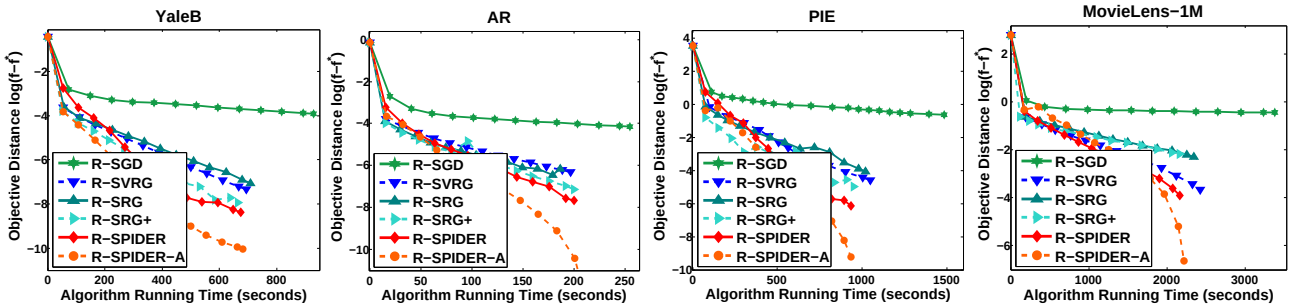
In this subsection, we present more experimental results to show the algorithm running time comparison among the compared algorithms in the manuscript. The experimental results in Figure 1 only provides the algorithm running time comparison of the ijcnn and epsilon datasets. Here we provide the comparison of all remaining datasets in Figure 4 which respond to Figures 1 and 3 in the manuscript. From the curves of comparison of optimality gap vs. algorithm running time, one can observe that our R-SPIDER-A is the fastest method and R-SPIDER can also quickly converge to a relatively high accuracy, *e.g.* 10^{-8}. We have discussed these results in the manuscript. Besides, all these results are consistent with the curves of the comparison of optimality gap vs. IFO, since the IFO complexity can comprehensively reflect the overall computational performance of a first-order Riemannian algorithm.

[1] https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/

[2] https://grouplens.org/datasets/movielens/1m/



(a) Comparison among Riemannian stochastic gradient algorithms on k-PCA problem.



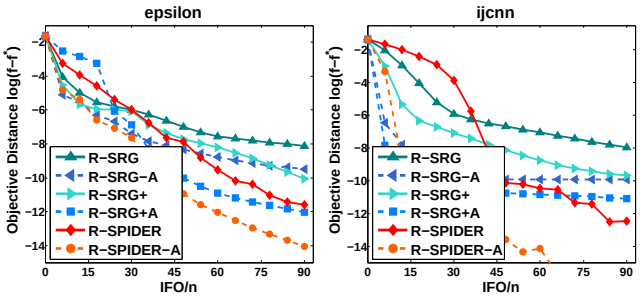
(b) Comparison among Riemannian stochastic gradient algorithms on low-rank matrix completion problem.

Figure 4: Comparison of algorithm running time of Riemannian stochastic gradient algorithms.

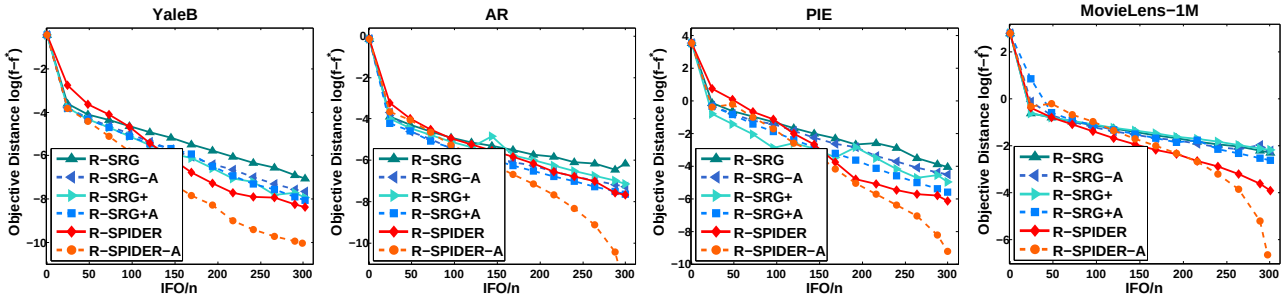
D.3 Comparison between Riemannian Stochastic Gradient Algorithms with Adaptive Learning Rate

Here we provide more comparison among our proposed R-SPIDER-A, R-SRG-A and R-SRG+A. R-SRG-A and R-SRG+A are respectively the counterparts of R-SRG and R-SRG+ with adaptive learning rate of formulation $\eta_k = \alpha(1 + \alpha\lambda_\alpha \lfloor \frac{k}{p} \rfloor)$ [10]. Notice, the reason that we do not compare all algorithms together is to avoid too many curves in one figure, leading to poor readability.

By observing Figure 5, we can find that the algorithm with adaptive learning rate usually outperforms the vanilla counterpart, which demonstrates the effectiveness of the strategy of adaptive learning rate. Moreover, R-SPIDER-A also consistently shows sharpest convergence behaviors compared with R-SRG-A and R-SRG+A. All these results are consistent with the experimental results in the manuscript. All results shows the advantages of our proposed R-SPIDER and R-SPIDER-A.



(a) Comparison among Riemannian stochastic gradient algorithms on k-PCA problem.



(b) Comparison among Riemannian stochastic gradient algorithms on low-rank matrix completion problem.

Figure 5: More comparison between R-SPIDER and R-SRG with adaptive learning rates.