SPIDER: Near-Optimal Non-Convex Optimization via Stochastic Path Integrated Differential Estimator

Cong Fang[1] * **Chris Junchi Li**[2] **Zhouchen Lin**[1†] **Tong Zhang**[2]
[1]Key Lab. of Machine Intelligence (MoE), School of EECS, Peking University
[2]Tencent AI Lab
{fangcong, zlin}@pku.edu.cn junchi.li.duke@gmail.com tongzhang@tongzhang-ml.org

Abstract

In this paper, we propose a new technique named *Stochastic Path-Integrated Differential EstimatoR* (SPIDER), which can be used to track many deterministic quantities of interests with significantly reduced computational cost. Combining SPIDER with the method of normalized gradient descent, we propose SPIDER-SFO that solve non-convex stochastic optimization problems using stochastic gradients only. We provide a few error-bound results on its convergence rates. Specially, we prove that the SPIDER-SFO algorithm achieves a gradient computation cost of $\mathcal{O}\left(\min(n^{1/2}\epsilon^{-2}, \epsilon^{-3})\right)$ to find an ϵ-approximate first-order stationary point. In addition, we prove that SPIDER-SFO nearly matches the algorithmic lower bound for finding stationary point under the gradient Lipschitz assumption in the finite-sum setting. Our SPIDER technique can be further applied to find an $(\epsilon, \mathcal{O}(\epsilon^{0.5}))$-approximate second-order stationary point at a gradient computation cost of $\tilde{\mathcal{O}}\left(\min(n^{1/2}\epsilon^{-2} + \epsilon^{-2.5}, \epsilon^{-3})\right)$.

1 Introduction

In this paper, we study the optimization problem

$$\underset{\mathbf{x}\in\mathbb{R}^d}{\text{minimize}} \quad f(\mathbf{x}) \equiv \mathbb{E}\left[F(\mathbf{x};\boldsymbol{\zeta})\right] \tag{1.1}$$

where the stochastic component $F(\mathbf{x};\boldsymbol{\zeta})$, indexed by some random vector $\boldsymbol{\zeta}$, is smooth and possibly *non-convex*. Non-convex optimization problem of form (1.1) contains many large-scale statistical learning tasks and is gaining tremendous popularity due to its favorable computational and statistical efficiency [5–7]. Typical examples of form (1.1) include principal component analysis, estimation of graphical models, as well as training deep neural networks [17]. The expectation-minimization structure of stochastic optimization problem (1.1) allows us to perform iterative updates and minimize the objective using its stochastic gradient $\nabla F(\mathbf{x};\boldsymbol{\zeta})$ as an estimator of its deterministic counterpart.

A special case of central interest is when the stochastic vector $\boldsymbol{\zeta}$ is finitely sampled. In such *finite-sum* (or *offline*) case, we denote each component function as $f_i(x)$ and (1.1) can be restated as

$$\underset{\mathbf{x}\in\mathbb{R}^d}{\text{minimize}} \quad f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}) \tag{1.2}$$

*This work was done while Cong Fang was a Research Intern with Tencent AI Lab.
†Corresponding author.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

where n is the number of individual functions. Another case is when n is reasonably large or even infinite, running across of the whole dataset is exhaustive or impossible. We refer it as the *online* (or *streaming*) case. For simplicity of notations we will study the optimization problem of form (1.2) in both finite-sum and online cases till the rest of this paper.

One important task for non-convex optimization is to search for, given the precision accuracy $\epsilon > 0$, an *ϵ-approximate first-order stationary point* $\mathbf{x} \in \mathbb{R}^d$ or $\|\nabla f(\mathbf{x})\| \leq \epsilon$. In this paper, we aim to propose a new technique, called the *Stochastic Path-Integrated Differential EstimatoR* (SPIDER), which enables us to construct an estimator that tracks a deterministic quantity with significantly lower sampling costs. As the readers will see, the SPIDER technique further allows us to design an algorithm with a faster rate of convergence for non-convex problem (1.2), in which we utilize the idea of *Normalized Gradient Descent* (NGD) [18, 26]. NGD is a variant of Gradient Descent (GD) where the stepsize is picked to be inverse-proportional to the norm of the full gradient. Compared to GD, NGD exemplifies faster convergence, especially in the neighborhood of stationary points [25]. However, NGD has been less popular due to its requirement of accessing the full gradient and its norm at each update. In this paper, we estimate and track the gradient and its norm via the SPIDER technique and then hybrid it with NGD. Measured by *gradient cost* which is the total number of computation of stochastic gradients, our proposed SPIDER-SFO algorithm achieves a faster rate of convergence in $\mathcal{O}(\min(n^{1/2}\epsilon^{-2}, \epsilon^{-3}))$ which outperforms the previous best-known results in both finite-sum [3][32] and online cases [24] by a factor of $\mathcal{O}(\min(n^{1/6}, \epsilon^{-0.333}))$.

For the task of finding stationary points for which we already achieved a faster convergence rate via our proposed SPIDER-SFO algorithm, a follow-up question to ask is: *is our proposed* SPIDER-*SFO algorithm optimal for an appropriate class of smooth functions?* In this paper, we provide an *affirmative* answer to this question in the finite-sum case. To be specific, inspired by a counterexample proposed by Carmon et al. [10] we are able to prove that the gradient cost upper bound of SPIDER-SFO algorithm matches the *algorithmic lower bound*. To put it differently, the gradient cost of SPIDER-SFO *cannot* be further improved for finding stationary points for some particular non-convex functions.

1.1 Related Works

In the recent years, there has been a surge of literatures in machine learning community that analyze the convergence property of non-convex optimization algorithms. Limited by space and our knowledge, we have listed all literatures that we believe are mostly related to this work. We refer the readers to the monograph by Jain et al. [19] and the references therein on recent general and model-specific convergence rate results on non-convex optimization.

SGD and Variance Reduction For the general problem of finding approximate stationary points, under the smoothness condition of $f(\mathbf{x})$, it is known that vanilla Gradient Descent (GD) and Stochastic Gradient Descent (SGD), which can be traced back to Cauchy [11] and Robbins & Monro [33] and achieve an ϵ-approximate stationary point with a gradient cost of $\mathcal{O}(\min(n\epsilon^{-2}, \epsilon^{-4}))$ [16, 26].

Recently, the convergence rate of GD and SGD have been improved by the variance-reduction type of algorithms [22, 34]. In special, the finite-sum Stochastic Variance-Reduced Gradient (SVRG) and online Stochastically Controlled Stochastic Gradient (SCSG), to the gradient cost of $\tilde{\mathcal{O}}(\min(n^{2/3}\epsilon^{-2}, \epsilon^{-3.333}))$ [3, 24, 32].

First-order method for finding approximate second-order stationary points It has been shown that for machine learning methods such as deep learning, approximate stationary points that have at least one negative Hessian direction, including saddle points and local maximizers, are often *not* sufficient and need to be avoided or escaped from [12, 15]. Recently, many literature study the problem of how to avoid or escape saddle points and achieve an (ϵ, δ)-approximate second-order stationary point $\mathbf{x}$ at a polynomial gradient cost, i.e. an $\mathbf{x} \in \mathbb{R}^d$ such that $\|\nabla f(\mathbf{x})\| \leq \epsilon, \lambda_{\min}(\nabla^2 f(\mathbf{x})) \geq -\delta$ [1, 2, 4, 8, 15, 18, 20, 21, 23, 25, 30, 31, 35, 38]. Among them, the group of authors Ge et al. [15], Jin et al. [20] proposed the noise-perturbed variants of Gradient Descent (PGD) and Stochastic

Gradient Descent (SGD) that escape from all saddle points and achieve an ϵ-approximate second-order stationary point in gradient cost of $\tilde{\mathcal{O}}(\min(n\epsilon^{-2}, poly(d)\epsilon^{-4}))$ stochastic gradients. Levy [25] proposed the noise-perturbed variant of NGD which yields faster evasion of saddle points than GD.

The breakthrough of gradient cost for finding second-order stationary points were achieved in 2016/2017, when the two recent lines of literatures, namely FastCubic [1] and CDHS [8] as well as their stochastic versions [2, 35], achieve a gradient cost of $\tilde{\mathcal{O}}(\min(n\epsilon^{-1.5}+n^{3/4}\epsilon^{-1.75}, \epsilon^{-3.5}))$ which serves as the best-known gradient cost for finding an $(\epsilon, \mathcal{O}(\epsilon^{0.5}))$-approximate second-order stationary point before the initial submission of this paper.[3] [4] In particular, Agarwal et al. [1], Tripuraneni et al. [35] converted the cubic regularization method for finding second-order stationary points [27] to stochastic- gradient based and stochastic-Hessian-vector-product-based methods, and Allen-Zhu [2], Carmon et al. [8] used a Negative-Curvature Search method to avoid saddle points. See also recent works by Reddi et al. [31] for related saddle-point-escaping methods that achieve similar rates for finding an approximate second-order stationary point.

Other concurrent works As the current work is carried out in its final phase, the authors became aware that an idea of resemblance was earlier presented in an algorithm named the *StochAstic Recursive grAdient algoritHm* (SARAH) [28, 29]. Despite the fact that both our SPIDER-SFO and theirs adopt the recursive stochastic gradient update framework and our SPIDER-SFO can be viewed as a variant of SARAH with normalization, our work differ from their works in two aspects:

(i) Our analysis techniques are totally different from the version of SARAH proposed by Nguyen et al. [28, 29]. Their version can be seen as a variant of gradient descent, while ours hybrids the SPIDER technique with normalized gradient descent. Moreover, Nguyen et al. [28, 29] adopt a large stepsize setting (in fact their goal was to design a memory-saving variant of SAGA [13]), while our SPIDER-SFO algorithm adopt a small stepsize that is proportional to ϵ. All these are essential elements of our superior achievements in convergence rates;

(ii) Our proposed SPIDER technique is a much more general variance-reduced estimation method for many quantities (*not* limited to gradients) and can be flexibly applied to numerous problems, e.g. stochastic zeroth-order method.

Soon after the initial submission to NIPS and arXiv release of this paper, we became aware that similar convergence rate results for stochastic first-order method were also achieved independently by the so-called SNVRG algorithm [39, 40]. SNVRG [40] obtains a gradient complexity of $\tilde{\mathcal{O}}(\min(n^{1/2}\epsilon^{-2}, \epsilon^{-3}))$ for finding an ϵ-approximate first-order stationary point and achieves a $\tilde{\mathcal{O}}(\epsilon^{-3.5})$ gradient cost for finding an $(\epsilon, \mathcal{O}(\epsilon^{0.5}))$-approximate second-order stationary point [39]. By exploiting the third-order smoothness, an SNVRG variant can also achieve an $(\epsilon, \mathcal{O}(\epsilon^{0.5}))$-approximate second-order stationary point in $\tilde{\mathcal{O}}(\epsilon^{-3})$ stochastic gradient costs [39].

1.2 Our Contributions

In this work, we propose the Stochastic Path-Integrated Differential Estimator (SPIDER) technique, which significantly avoids excessive access of stochastic oracles and reduces the time complexity. Such technique can be potential applied in many stochastic estimation problems.

(i) We propose the SPIDER-SFO algorithm (Algorithm 1) for finding approximate first-order stationary points for non-convex stochastic optimization problem (1.2), and prove the optimality of such rate in at least one case. Inspired by recent works Carmon et al. [8, 10], Johnson & Zhang [22] and independent of Zhou et al. [39, 40], this is the *first* time that the gradient cost of $\mathcal{O}(\min(n^{1/2}\epsilon^{-2}, \epsilon^{-3}))$ in both upper and lower (finite-sum only) bound for finding first-order stationary points for problem (1.2) were obtained.

[3] Allen-Zhu [2] also obtains a gradient cost of $\tilde{\mathcal{O}}(\epsilon^{-3.25})$ to achieve a (modified and weakened) $(\epsilon, \mathcal{O}(\epsilon^{0.25}))$-approximate second-order stationary point.

[4] Here and in many places afterwards, the gradient cost also includes the number of stochastic Hessian-vector product accesses, which has similar running time with computing per-access stochastic gradient.

(ii) Following Allen-Zhu & Li [4], Carmon et al. [8], Xu et al. [38], we propose SPIDER-SFO$^+$ algorithm for finding an approximate second-order stationary point for non-convex stochastic optimization problem. To best of our knowledge, this is also the *first* time that the gradient cost of $\tilde{\mathcal{O}}(\min(n^{1/2}\epsilon^{-2}+\epsilon^{-2.5},\epsilon^{-3}))$ achieved with standard assumptions. We leave the details of SFO in the long version of our paper: `https://arxiv.org/abs/1807.01695`

(iii) We propose a new and simpler analysis framework for proving convergence to approximate stationary points. One can flexibly apply our proof techniques to analyze others algorithms, e.g. SGD, SVRG [22], and SAGA [13].

Notation. Throughout this paper, we treat the parameters L, Δ, σ, and ρ, to be specified later as global constants. Let $\|\cdot\|$ denote the Euclidean norm of a vector or spectral norm of a square matrix. Denote $p_n = \mathcal{O}(q_n)$ for a sequence of vectors p_n and positive scalars q_n if there is a global constant C such that $|p_n| \le Cq_n$, and $p_n = \tilde{\mathcal{O}}(q_n)$ such C hides a poly-logarithmic factor of the parameters. Denote $p_n = \Omega(q_n)$ if there is a global constant C such that $|p_n| \ge Cq_n$. Let $\lambda_{\min}(\mathbf{A})$ denote the least eigenvalue of a real symmetric matrix $\mathbf{A}$. For fixed $K \ge k \ge 0$, let $\mathbf{x}_{k:K}$ denote the sequence $\{\mathbf{x}^k, \ldots, \mathbf{x}^K\}$. Let $[n] = \{1, \ldots, n\}$ and S denote the cardinality of a multi-set $\mathcal{S} \subset [n]$ of samples (a generic set that allows elements of multiple instances). For simplicity, we further denote the averaged sub-sampled stochastic estimator $\mathcal{B}_S := (1/S)\sum_{i\in\mathcal{S}} \mathcal{B}_i$ and averaged sub-sampled gradient $\nabla f_S := (1/S)\sum_{i\in\mathcal{S}} \nabla f_i$. Other notations are explained at their first appearance.

2 Stochastic Path-Integrated Differential Estimator: Core Idea

In this section, we present in detail the underlying idea of our Stochastic Path-Integrated Differential Estimator (SPIDER) technique behind the algorithm design. As the readers will see, such technique significantly avoids excessive access of stochastic oracle and reduces complexity, which is of independent interest and has potential applications in many stochastic estimation problems.

Let us consider an arbitrary deterministic vector quantity $Q(\mathbf{x})$. Assume that we observe a sequence $\hat{\mathbf{x}}_{0:K}$, and we want to dynamically track $Q(\hat{\mathbf{x}}^k)$ for $k = 0, 1, \ldots, K$. Assume further that we have an initial estimate $\tilde{Q}(\hat{\mathbf{x}}^0) \approx Q(\hat{\mathbf{x}}^0)$, and an unbiased estimate $\boldsymbol{\xi}_k(\hat{\mathbf{x}}_{0:k})$ of $Q(\hat{\mathbf{x}}^k) - Q(\hat{\mathbf{x}}^{k-1})$ such that for each $k = 1, \ldots, K$

$$\mathbb{E}\left[\boldsymbol{\xi}_k(\hat{\mathbf{x}}_{0:k}) \mid \hat{\mathbf{x}}_{0:k}\right] = Q(\hat{\mathbf{x}}^k) - Q(\hat{\mathbf{x}}^{k-1}).$$

Then we can integrate (in the discrete sense) the stochastic differential estimate as

$$\tilde{Q}(\hat{\mathbf{x}}_{0:K}) := \tilde{Q}(\hat{\mathbf{x}}^0) + \sum_{k=1}^{K} \boldsymbol{\xi}_k(\hat{\mathbf{x}}_{0:k}). \tag{2.1}$$

We call estimator $\tilde{Q}(\hat{\mathbf{x}}_{0:K})$ the *Stochastic Path-Integrated Differential EstimatoR*, or SPIDER for brevity. We conclude the following proposition which bounds the error of our estimator $\|\tilde{Q}(\hat{\mathbf{x}}_{0:K}) - Q(\hat{\mathbf{x}}^K)\|$, in terms of both expectation and high probability:

Proposition 1. *The martingale variance bound has*

$$\mathbb{E}\|\tilde{Q}(\hat{\mathbf{x}}_{0:K}) - Q(\hat{\mathbf{x}}^K)\|^2 = \mathbb{E}\|\tilde{Q}(\hat{\mathbf{x}}^0) - Q(\hat{\mathbf{x}}^0)\|^2 + \sum_{k=1}^{K} \mathbb{E}\|\boldsymbol{\xi}_k(\hat{\mathbf{x}}_{0:k}) - (Q(\hat{\mathbf{x}}^k) - Q(\hat{\mathbf{x}}^{k-1}))\|^2. \tag{2.2}$$

Proposition 1 can be easily concluded using the property of square-integrable martingales. Now, let $\mathcal{B}$ map any $\mathbf{x} \in \mathbb{R}^d$ to a random estimate $\mathcal{B}_i(\mathbf{x})$ such that, conditioning on the observed sequence $\mathbf{x}_{0:k}$, we have for each $k = 1, \ldots, K$,

$$\mathbb{E}\left[\mathcal{B}_i(\mathbf{x}^k) - \mathcal{B}_i(\mathbf{x}^{k-1}) \mid \mathbf{x}_{0:k}\right] = \mathcal{V}^k - \mathcal{V}^{k-1}. \tag{2.3}$$

At each step k let S_* be a subset that samples $\mathcal{S}_*$ elements in $[n]$ with replacement, and let the stochastic estimator $\mathcal{B}_{S_*} = (1/\mathcal{S}_*)\sum_{i\in S_*}\mathcal{B}_i$ satisfy

$$\mathbb{E}\|\mathcal{B}_i(\mathbf{x}) - \mathcal{B}_i(\mathbf{y})\|^2 \le L_{\mathcal{B}}^2\|\mathbf{x}-\mathbf{y}\|^2, \tag{2.4}$$

and $\|\mathbf{x}^k - \mathbf{x}^{k-1}\| \le \epsilon_1$ for all $k = 1,\dots,K$. Finally, we set our estimator $\mathcal{V}^k$ of $\mathcal{B}(\mathbf{x}^k)$ as

$$\mathcal{V}^k = \mathcal{B}_{S_*}(\mathbf{x}^k) - \mathcal{B}_{S_*}(\mathbf{x}^{k-1}) + \mathcal{V}^{k-1}.$$

Applying Proposition 1 immediately concludes the following lemma, which gives an error bound of the estimator $\mathcal{V}^k$ in terms of the second moment of $\|\mathcal{V}^k - \mathcal{B}(\mathbf{x}^k)\|$:

Lemma 1. *We have under the condition* (2.4) *that for all* $k = 1,\dots,K$,

$$\mathbb{E}\|\mathcal{V}^k - \mathcal{B}(\mathbf{x}^k)\|^2 \le \frac{kL_{\mathcal{B}}^2\epsilon_1^2}{\mathcal{S}_*} + \mathbb{E}\|\mathcal{V}^0 - \mathcal{B}(\mathbf{x}^0)\|^2. \tag{2.5}$$

It turns out that one can use SPIDER to track many quantities of interest, such as stochastic gradient, function values, zero-order estimate gradient, functionals of Hessian matrices, etc. Our proposed SPIDER-based algorithms in this paper take $\mathcal{B}_i$ as the stochastic gradient ∇f_i and the zeroth-order estimate gradient, separately.

3 SPIDER for Stochastic First-Order Method

In this section, we apply SPIDER to the Stochastic First-Order (SFO) method. We introduce the basic settings and assumptions in §3.1 and propose the main error-bound theorems for finding an ϵ-approximate first-order stationary point in §3.2. We conclude this section with the corresponding lower-bound result in §3.3.

3.1 Settings and Assumptions

We first introduce the formal definition of an approximate first-order stationary point as follows.

Definition 1. *We call* $\mathbf{x}\in\mathbb{R}^d$ *an* ϵ*-approximate first-order stationary point, or simply an FSP, if*

$$\|\nabla f(\mathbf{x})\| \le \epsilon. \tag{3.1}$$

For our purpose of analysis, we also pose the following assumption:

Assumption 1. *We assume the following*

(i) The $\Delta := f(\mathbf{x}^0) - f^* < \infty$ *where* $f^* = \inf_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x})$ *is the global infimum value of* $f(\mathbf{x})$*;*

(ii) The component function $f_i(\mathbf{x})$ *has an averaged* L*-Lipschitz gradient, i.e. for all* $\mathbf{x},\mathbf{y}$,

$$\mathbb{E}\|\nabla f_i(\mathbf{x}) - \nabla f_i(\mathbf{y})\|^2 \le L^2\|\mathbf{x}-\mathbf{y}\|^2;$$

(iii) (For online case only) the stochastic gradient has a finite variance bounded by $\sigma^2 < \infty$*, i.e.*

$$\mathbb{E}\left\|\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})\right\|^2 \le \sigma^2.$$

3.2 Upper Bound for Finding First-Order Stationary Points

Recall that NGD has iteration update rule

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \eta\frac{\nabla f(\mathbf{x}^k)}{\|\nabla f(\mathbf{x}^k)\|}, \tag{3.2}$$

where η is a constant step size. The NGD update rule (3.2) ensures $\|\mathbf{x}^{k+1} - \mathbf{x}^k\|$ being constantly equal to the stepsize η, and might fastly escape from saddle points and converge to a second-order

Algorithm 1 SPIDER-SFO: Input $\mathbf{x}^0$, q, S_1, S_2, n_0, ϵ, and $\tilde{\epsilon}$ (For finding first-order stationary point)

1: **for** $k = 0$ to K **do**
2: **if** $\mod(k, q) = 0$ **then**
3: Draw S_1 samples (or compute the full gradient for the finite-sum case), let $\mathbf{v}^k = \nabla f_{\mathcal{S}_1}(\mathbf{x}^k)$
4: **else**
5: Draw S_2 samples, and let $\mathbf{v}^k = \nabla f_{\mathcal{S}_2}(\mathbf{x}^k) - \nabla f_{\mathcal{S}_2}(\mathbf{x}^{k-1}) + \mathbf{v}^{k-1}$
6: **end if**

7: **OPTION I** ◇ for convergence rates in high probability
8: **if** $\|\mathbf{v}^k\| \leq 2\tilde{\epsilon}$ **then**
9: **return** $\mathbf{x}^k$
10: **else**
11: $\mathbf{x}^{k+1} = \mathbf{x}^k - \eta \cdot (\mathbf{v}^k / \|\mathbf{v}^k\|)$ where $\eta = \dfrac{\epsilon}{Ln_0}$
12: **end if**

13: **OPTION II** ◇ for convergence rates in expectation
14: $\mathbf{x}^{k+1} = \mathbf{x}^k - \eta^k \mathbf{v}^k$ where $\eta^k = \min\left(\dfrac{\epsilon}{Ln_0\|\mathbf{v}^k\|}, \dfrac{1}{2Ln_0}\right)$

15: **end for**

16: **OPTION I**: Return $\mathbf{x}^K$ ◇ however, this line is *not* reached with high probability

17: **OPTION II**: Return $\tilde{\mathbf{x}}$ chosen uniformly at random from $\{\mathbf{x}^k\}_{k=0}^{K-1}$

stationary point [25]. We propose SPIDER-SFO in Algorithm 1, which resembles a stochastic variant of NGD with the SPIDER technique applied, so that one can maintain an estimate of $\nabla f(\mathbf{x}^k)$ at a higher accuracy under limited gradient budgets.

To analyze the convergence rate of SPIDER-SFO, let us first consider the online case for Algorithm 1. We let the input parameters be

$$S_1 = \frac{2\sigma^2}{\epsilon^2}, \qquad S_2 = \frac{2\sigma}{\epsilon n_0}, \qquad \eta = \frac{\epsilon}{Ln_0}, \qquad \eta^k = \min\left(\frac{\epsilon}{Ln_0\|\mathbf{v}^k\|}, \frac{1}{2Ln_0}\right), \qquad q = \frac{\sigma n_0}{\epsilon}, \tag{3.3}$$

where $n_0 \in [1, 2\sigma/\epsilon]$ is a free parameter to choose.[5] In this case, $\mathbf{v}^k$ in Line 5 of Algorithm 1 is a SPIDER for $\nabla f(\mathbf{x}^k)$. To see this, recall $\nabla f_i(\mathbf{x}^{k-1})$ is the stochastic gradient drawn at step k and

$$\mathbb{E}\left[\nabla f_i(\mathbf{x}^k) - \nabla f_i(\mathbf{x}^{k-1}) \mid \mathbf{x}_{0:k}\right] = \nabla f(\mathbf{x}^k) - \nabla f(\mathbf{x}^{k-1}). \tag{3.4}$$

Plugging in $\mathcal{V}^k = \mathbf{v}^k$ and $\mathcal{B}_i = \nabla f_i$ in Lemma 1 of §2, we can use $\mathbf{v}^k$ in Algorithm 1 as the SPIDER and conclude the following lemma that is pivotal to our analysis.

Lemma 2. *Set the parameters S_1, S_2, η, and q as in* (3.3)*, and $k_0 = \lfloor k/q \rfloor \cdot q$. Then under the Assumption 1, we have*

$$\mathbb{E}\left[\|\mathbf{v}^k - \nabla f(\mathbf{x}^k)\|^2 \mid \mathbf{x}_{0:k_0}\right] \leq \epsilon^2.$$

Here we compute the conditional expectation over the randomness of $x_{(k_0+1):k}$.

Lemma 2 shows that our SPIDER $\mathbf{v}^k$ of $\nabla f(\mathbf{x})$ maintains an error of $\mathcal{O}(\epsilon)$. Using this lemma, we are ready to present the following results for Stochastic First-Order (SFO) method for finding first-order stationary points of (1.2).

Theorem 1 (First-order stationary point, online setting, in expectation)**.** *Assume we are in the online case, let Assumption 1 holds, set the parameters S_1, S_2, η, and q as in* (3.3)*, and set $K = \lfloor (4L\Delta n_0)\epsilon^{-2} \rfloor + 1$. Then running Algorithm 1 with OPTION II for K iterations outputs a $\tilde{\mathbf{x}}$*

[5] When $n_0 = 1$, the mini-batch size is $2\sigma/\epsilon$, which is the largest mini-batch size that Algorithm 1 allows to choose.

satisfying

$$\mathbb{E}\left[\|\nabla f(\tilde{\mathbf{x}})\|\right] \leq 5\epsilon. \tag{3.5}$$

The gradient cost is bounded by $24L\Delta\sigma\cdot\epsilon^{-3}+2\sigma^2\epsilon^{-2}+4\sigma n_0^{-1}\epsilon^{-1}$ *for any choice of* $n_0\in[1,2\sigma/\epsilon]$. *Treating* Δ, L *and* σ *as positive constants, the stochastic gradient complexity is* $\mathcal{O}(\epsilon^{-3})$.

The relatively reduced minibatch size serves as the key ingredient for the superior performance of SPIDER-SFO. For illustrations, let us compare the sampling efficiency among SGD, SCSG and SPIDER-SFO in their special cases. With some involved analysis of the algorithms above, we can conclude that to ensure per-iteration sufficient decrease of $\Omega(\epsilon^2/L)$, we have

(i) for SGD the choice of mini-batch size is $\mathcal{O}\big(\sigma^2\cdot\epsilon^{-2}\big)$;

(ii) for SCSG [24] and Natasha2 [2] the mini-batch size is $\mathcal{O}\big(\sigma\cdot\epsilon^{-1.333}\big)$;

(iii) for our SPIDER-SFO, only a reduced mini-batch size of $\mathcal{O}\big(\sigma\cdot\epsilon^{-1}\big)$ is needed.

Turning to the finite-sum case, analogous to the online case we let

$$S_2=\frac{n^{1/2}}{n_0},\qquad \eta=\frac{\epsilon}{Ln_0},\qquad \eta^k=\min\left(\frac{\epsilon}{Ln_0\|\mathbf{v}^k\|},\frac{1}{2Ln_0}\right),\qquad q=n_0n^{1/2}, \tag{3.6}$$

where $n_0\in[1,n^{1/2}]$. In this case, one computes the full gradient $\mathbf{v}^k=\nabla f_{S_1}(\mathbf{x}^k)$ in Line 3 of Algorithm 1. We conclude our second upper-bound result:

Theorem 2 (First-order stationary point, finite-sum setting, in expectation)**.** *Assume we are in the finite-sum case, let Assumption 1 holds, set the parameters* $\mathcal{S}_2$, η^k, *and* q *as in* (3.6)*, set* $K=\big\lfloor(4L\Delta n_0)\epsilon^{-2}\big\rfloor+1$, *and let* $S_1=[n]$, *i.e. we obtain the full gradient in Line 3. Then running Algorithm 1 with OPTION II for* K *iterations outputs a* $\tilde{\mathbf{x}}$ *satisfying*

$$\mathbb{E}\|\nabla f(\tilde{\mathbf{x}})\|\leq 5\epsilon.$$

The gradient cost is bounded by $n+12(L\Delta)\cdot n^{1/2}\epsilon^{-2}+2n_0^{-1}n^{1/2}$ *for any choice of* $n_0\in[1,n^{1/2}]$. *Treating* Δ, L *and* σ *as positive constants, the stochastic gradient complexity is* $\mathcal{O}(n+n^{1/2}\epsilon^{-2})$.

3.3 Lower Bound for Finding First-Order Stationary Points

To conclude the optimality of our algorithm we need an algorithmic lower bound result [10, 37]. Consider the finite-sum case and any random algorithm $\mathcal{A}$ that maps functions $f:\mathbb{R}^d\to\mathbb{R}$ to a sequence of iterates in $\mathbb{R}^{d+1}$, with

$$[\mathbf{x}^k;i_k]=\mathcal{A}^{k-1}\big(\boldsymbol{\xi},\nabla f_{i_0}(\mathbf{x}^0),\nabla f_{i_1}(\mathbf{x}^1),\ldots,\nabla f_{i_{k-1}}(\mathbf{x}^{k-1})\big),\quad k\geq 1, \tag{3.7}$$

where $\mathcal{A}^k$ are measure mapping into $\mathbb{R}^{d+1}$, i_k is the individual function chosen by $\mathcal{A}$ at iteration k, and $\boldsymbol{\xi}$ is uniform random vector from $[0,1]$. And $[\mathbf{x}^0;i_0]=\mathcal{A}^0(\boldsymbol{\xi})$, where $\mathcal{A}^0$ is a measure mapping. The lower-bound result for solving (1.2) is stated as follows:

Theorem 3 (Lower bound for SFO for the finite-sum setting)**.** *For any* $L>0$, $\Delta>0$, *and* $2\leq n\leq O\left(\Delta^2L^2\cdot\epsilon^{-4}\right)$, *for any algorithm* $\mathcal{A}$ *satisfying* (3.7)*, there exists a dimension* $d=\tilde{\mathcal{O}}\big(\Delta^2L^2\cdot n^2\epsilon^{-4}\big)$, *and a function* f *satisfies Assumption 1 in the finite-sum case, such that in order to find a point* $\tilde{\mathbf{x}}$ *for which* $\|\nabla f(\tilde{\mathbf{x}})\|\leq\epsilon$, $\mathcal{A}$ *must cost at least* $\Omega\big(L\Delta\cdot n^{1/2}\epsilon^{-2}\big)$ *stochastic gradient accesses.*

Note the condition $n\leq\mathcal{O}(\epsilon^{-4})$ in Theorem 3 ensures that our lower bound $\Omega(n^{1/2}\epsilon^{-2})=\Omega(n+n^{1/2}\epsilon^{-2})$, and hence our upper bound in Theorem 1 matches the lower bound in Theorem 3 up to a constant factor of relevant parameters, and is hence *near-optimal*. Inspired by Carmon et al. [10], our proof of Theorem 3 utilizes a specific counterexample function that requires at least $\Omega(n^{1/2}\epsilon^{-2})$ stochastic gradient accesses. Note Carmon et al. [10] analyzed such counterexample in the deterministic case $n=1$ and we generalize such analysis to the finite-sum case $n\geq 1$.

Remark 1. *Note by setting* $n=\mathcal{O}(\epsilon^{-4})$ *the lower bound complexity in Theorem 3 can be as large as* $\Omega(\epsilon^{-4})$. *We emphasize that this does not violate the* $\mathcal{O}(\epsilon^{-3})$ *upper bound in the online case [Theorem*

1], since the counterexample established in the lower bound depends not on the stochastic gradient variance σ^2 specified in Assumption 1(iii), but on the component number n. To obtain the lower bound result for the online case with the additional Assumption 1(iii), with more efforts one might be able to construct a second counterexample that requires $\Omega(\epsilon^{-3})$ stochastic gradient accesses with the knowledge of σ instead of n. We leave this as a future work.

4 Further Extensions

Further extensions of our SPIDER technique can be successfully applied to reduce the complexity. Limited by space, we leave the details of the following important extensions in the long version of our paper at `https://arxiv.org/abs/1807.01695` .

Upper Bound for Finding First-Order Stationary Points, in High-Probability Under more stringent assumptions on the moments of stochastic gradients, our Algorithm 1 with OPTION I achieves a gradient cost of $\tilde{\mathcal{O}}(\min(n^{1/2}\epsilon^{-2}, \epsilon^{-3}))$ (note the additional polylogarithmic factor) with high probability. We detail the theorems and their proofs in the long version of our paper.

Second-Order Stationary Point To find a second-order stationary point with (3.1), we can fuse our SPIDER-SFO in Algorithm 1 (OPTION I taken) with a Negative-Curvature-Search (NC-Search) iteration. In the long version of our paper (and independent of [39]), we proved rigorously that a gradient cost of $\tilde{\mathcal{O}}(\min(n^{1/2}\epsilon^{-2} + \epsilon^{-2.5}, \epsilon^{-3}))$ can be achieved under standard assumptions:

Theorem 4 (Second-Order Stationary Point, Informal)**.** *There exists an algorithm such that under appropriate assumptions it takes to find a $(\epsilon, \sqrt{\rho\epsilon})$-second-order stationary point, we have for the online case, when $\epsilon \leq \rho\sigma^2$ the total number of stochastic gradient computations is $\tilde{\mathcal{O}}(\epsilon^{-3})$; For the finite-sum case, when $\epsilon \leq \rho n$, the total cost of gradient access is $\tilde{\mathcal{O}}(n\epsilon^{-1.5} + n^{1/2}\epsilon^{-2} + \epsilon^{-2.5})$.*

Zeroth-Order Stationary Point After the NIPS submission of this work, we propose a second application of our SPIDER technique to the stochastic zeroth-order method for problem (1.2) and achieves individual function accesses of $\mathcal{O}(\min(dn^{1/2}\epsilon^{-2}, d\epsilon^{-3}))$. To best of our knowledge, this is also the *first* time a complexity of individual function value accesses for non-convex problems has been improved to the aforementioned complexity using variance reduction techniques [22, 34].

5 Summary and Future Directions

We propose in this work the SPIDER method for non-convex optimization. Our SPIDER-type algorithms have update rules that are reasonably simple and achieve excellent convergence properties. However, there are still some important questions are left. For example, the lower bound results for finding a second-order stationary point are *not* complete. Specially, it is *not* yet clear if our $\tilde{\mathcal{O}}(\epsilon^{-3})$ for the online case and $\tilde{\mathcal{O}}(n^{1/2}\epsilon^{-2})$ for the finite-sum case gradient cost upper bound for finding a second-order stationary point (when $n \geq \Omega(\epsilon^{-1})$) is *optimal* or the gradient cost can be further improved, assuming both Lipschitz gradient and Lipschitz Hessian.

Acknowledgement The authors would like to thank Jeffrey Z. HaoChen for his help on the numerical experiments, thank an anonymous reviewer to point out a mistake in the original proof of Theorem 1 and thank Zeyuan Allen-Zhu and Quanquan Gu for relevant discussions and pointing out references Zhou et al. [39, 40], also Jianqiao Wangni for pointing out references Nguyen et al. [28, 29], and Zebang Shen, Ruoyu Sun, Haishan Ye, Pan Zhou for very helpful discussions and comments. Zhouchen Lin is supported by National Basic Research Program of China (973 Program, grant no. 2015CB352502), National Natural Science Foundation (NSF) of China (grant nos. 61625301 and 61731018), and Microsoft Research Asia.

References

[1] Agarwal, N., Allen-Zhu, Z., Bullins, B., Hazan, E., & Ma, T. (2017). Finding approximate local minima faster than gradient descent. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing* (pp. 1195–1199).: ACM.

[2] Allen-Zhu, Z. (2018). Natasha 2: Faster non-convex optimization than sgd. In *Advances in Neural Information Processing Systems*.

[3] Allen-Zhu, Z. & Hazan, E. (2016). Variance reduction for faster non-convex optimization. In *International Conference on Machine Learning* (pp. 699–707).

[4] Allen-Zhu, Z. & Li, Y. (2018). Neon2: Finding local minima via first-order oracles. In *Advances in Neural Information Processing Systems*.

[5] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010* (pp. 177–186). Springer.

[6] Bottou, L., Curtis, F. E., & Nocedal, J. (2018). Optimization methods for large-scale machine learning. *SIAM Review*, 60(2), 223–311.

[7] Bubeck, S. et al. (2015). Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4), 231–357.

[8] Carmon, Y., Duchi, J. C., Hinder, O., & Sidford, A. (2016). Accelerated methods for non-convex optimization. *To appear in SIAM Journal on Optimization, accepted*.

[9] Carmon, Y., Duchi, J. C., Hinder, O., & Sidford, A. (2017a). "Convex Until Proven Guilty": Dimension-free acceleration of gradient descent on non-convex functions. In *International Conference on Machine Learning* (pp. 654–663).

[10] Carmon, Y., Duchi, J. C., Hinder, O., & Sidford, A. (2017b). Lower bounds for finding stationary points i. *arXiv preprint arXiv:1710.11606*.

[11] Cauchy, A. (1847). Méthode générale pour la résolution des systemes déquations simultanées. *Comptes Rendus de l'Academie des Science*, 25, 536–538.

[12] Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., & Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems* (pp. 2933–2941).

[13] Defazio, A., Bach, F., & Lacoste-Julien, S. (2014). SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems* (pp. 1646–1654).

[14] Durrett, R. (2010). *Probability: Theory and Examples (4th edition)*. Cambridge University Press.

[15] Ge, R., Huang, F., Jin, C., & Yuan, Y. (2015). Escaping from saddle points – online stochastic gradient for tensor decomposition. In *Proceedings of The 28th Conference on Learning Theory* (pp. 797–842).

[16] Ghadimi, S. & Lan, G. (2013). Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4), 2341–2368.

[17] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

[18] Hazan, E., Levy, K., & Shalev-Shwartz, S. (2015). Beyond convexity: Stochastic quasi-convex optimization. In *Advances in Neural Information Processing Systems* (pp. 1594–1602).

[19] Jain, P., Kar, P., et al. (2017). Non-convex optimization for machine learning. *Foundations and Trends® in Machine Learning*, 10(3-4), 142–336.

[20] Jin, C., Ge, R., Netrapalli, P., Kakade, S. M., & Jordan, M. I. (2017a). How to escape saddle points efficiently. In *International Conference on Machine Learning* (pp. 1724–1732).

[21] Jin, C., Netrapalli, P., & Jordan, M. I. (2017b). Accelerated gradient descent escapes saddle points faster than gradient descent. *arXiv preprint arXiv:1711.10456*.

[22] Johnson, R. & Zhang, T. (2013). Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems* (pp. 315–323).

[23] Lee, J. D., Simchowitz, M., Jordan, M. I., & Recht, B. (2016). Gradient descent only converges to minimizers. In *Proceedings of The 29th Conference on Learning Theory* (pp. 1246–1257).

[24] Lei, L., Ju, C., Chen, J., & Jordan, M. I. (2017). Non-convex finite-sum optimization via scsg methods. In *Advances in Neural Information Processing Systems* (pp. 2345–2355).

[25] Levy, K. Y. (2016). The power of normalization: Faster evasion of saddle points. *arXiv preprint arXiv:1611.04831*.

[26] Nesterov, Y. (2004). *Introductory lectures on convex optimization: A basic course*, volume 87. Springer.

[27] Nesterov, Y. & Polyak, B. T. (2006). Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1), 177–205.

[28] Nguyen, L. M., Liu, J., Scheinberg, K., & Takáč, M. (2017a). SARAH: A novel method for machine learning problems using stochastic recursive gradient. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research* (pp. 2613–2621). International Convention Centre, Sydney, Australia: PMLR.

[29] Nguyen, L. M., Liu, J., Scheinberg, K., & Takáč, M. (2017b). Stochastic recursive gradient algorithm for nonconvex optimization. *arXiv preprint arXiv:1705.07261*.

[30] Paquette, C., Lin, H., Drusvyatskiy, D., Mairal, J., & Harchaoui, Z. (2018). Catalyst for gradient-based nonconvex optimization. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics* (pp. 613–622).

[31] Reddi, S., Zaheer, M., Sra, S., Poczos, B., Bach, F., Salakhutdinov, R., & Smola, A. (2018). A generic approach for escaping saddle points. In A. Storkey & F. Perez-Cruz (Eds.), *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research* (pp. 1233–1242). Playa Blanca, Lanzarote, Canary Islands: PMLR.

[32] Reddi, S. J., Hefny, A., Sra, S., Poczos, B., & Smola, A. (2016). Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning* (pp. 314–323).

[33] Robbins, H. & Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, (pp. 400–407).

[34] Schmidt, M., Le Roux, N., & Bach, F. (2017). Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2), 83–112.

[35] Tripuraneni, N., Stern, M., Jin, C., Regier, J., & Jordan, M. I. (2018). Stochastic cubic regularization for fast nonconvex optimization. In *Advances in Neural Information Processing Systems*.

[36] Woodworth, B. & Srebro, N. (2017). Lower bound for randomized first order convex optimization. *arXiv preprint arXiv:1709.03594*.

[37] Woodworth, B. E. & Srebro, N. (2016). Tight complexity bounds for optimizing composite objectives. In *Advances in Neural Information Processing Systems* (pp. 3639–3647).

[38] Xu, Y., Jin, R., & Yang, T. (2017). First-order stochastic algorithms for escaping from saddle points in almost linear time. *arXiv preprint arXiv:1711.01944*.

[39] Zhou, D., Xu, P., & Gu, Q. (2018a). Finding local minima via stochastic nested variance reduction. *arXiv preprint arXiv:1806.08782*.

[40] Zhou, D., Xu, P., & Gu, Q. (2018b). Stochastic nested variance reduction for nonconvex optimization. *arXiv preprint arXiv:1806.07811*.

A Analysis of SPIDER

In this and next sections, we sometimes denote for brevity that $\mathbb{E}_k[\cdot] = \mathbb{E}[\cdot \mid x_{0:k}]$, the expectation operator conditional on $x_{0:k}$, for an arbitrary $k \geq 0$. We focus on the proofs of Proposition 1 and Lemma 1.

A.1 Proof of Proposition 1

Proof of Proposition 1. It is straightforward to verify from the definition of $\tilde{Q}$ in (2.1) that

$$\tilde{Q}(\hat{\mathbf{x}}_{0:K}) - Q(\hat{\mathbf{x}}^K) = \tilde{Q}(\hat{\mathbf{x}}^0) - Q(\hat{\mathbf{x}}^0) + \sum_{k=1}^{K} \boldsymbol{\xi}_k(\hat{\mathbf{x}}_{0:k}) - (Q(\hat{\mathbf{x}}^k) - Q(\hat{\mathbf{x}}^{k-1}))$$

is a martingale, and hence (2.2) follows from the property of L^2 martingales [14]. □

A.2 Proof of Lemma 1

Proof of Lemma 1. For any $k > 0$, we have from Proposition 1 (by applying $\tilde{Q} = \mathcal{V}$)

$$\mathbb{E}_k\|\mathcal{V}^k - \mathcal{B}(\mathbf{x}^k)\|^2 = \mathbb{E}_k\|\mathcal{B}_{S_*}(\mathbf{x}^k) - \mathcal{B}(\mathbf{x}^k) - \mathcal{B}_{S_*}(\mathbf{x}^{k-1}) + \mathcal{B}(\mathbf{x}^{k-1})\|^2 + \|\mathcal{V}^{k-1} - \mathcal{B}(\mathbf{x}^{k-1})\|^2. \tag{A.1}$$

Then

$$\begin{aligned} & \mathbb{E}_k\|\mathcal{B}_{S_*}(\mathbf{x}^k) - \mathcal{B}(\mathbf{x}^k) - \mathcal{B}_{S_*}(\mathbf{x}^{k-1}) + \mathcal{B}(\mathbf{x}^{k-1})\|^2 \\ \overset{a}{=} & \frac{1}{\mathcal{S}_*}\mathbb{E}\|\mathcal{B}_i(\mathbf{x}^k) - \mathcal{B}(\mathbf{x}^k) - \mathcal{B}_i(\mathbf{x}^{k-1}) + \mathcal{B}(\mathbf{x}^{k-1})\|^2 \\ \overset{b}{\leq} & \frac{1}{\mathcal{S}_*}\mathbb{E}\|\mathcal{B}_i(\mathbf{x}^k) - \mathcal{B}_i(\mathbf{x}^{k-1})\|^2 \\ \overset{(2.4)}{\leq} & \frac{1}{\mathcal{S}_*}L_{\mathcal{B}}^2\mathbb{E}\|\mathbf{x}^k - \mathbf{x}^{k-1}\|^2 \leq \frac{L_{\mathcal{B}}^2\epsilon_1^2}{\mathcal{S}_*}, \end{aligned} \tag{A.2}$$

where in $\overset{a}{=}$ and $\overset{b}{\leq}$, we use Eq (2.3), and S_* are random sampled from $[n]$ with replacement. Combining (A.1) and (A.2), we have

$$\mathbb{E}_k\|\mathcal{V}^k - \mathcal{B}(\mathbf{x}^k)\|^2 \leq \frac{L_{\mathcal{B}}^2\epsilon_1^2}{\mathcal{S}_*} + \|\mathcal{V}^{k-1} - \mathcal{B}(\mathbf{x}^{k-1})\|^2. \tag{A.3}$$

Telescoping the above display for $k' = k-1, \ldots, 0$ and using the iterated law of expectation, we have

$$\mathbb{E}\|\mathcal{V}^k - \mathcal{B}(\mathbf{x}^k)\|^2 \leq \frac{kL_{\mathcal{B}}^2\epsilon_1^2}{\mathcal{S}_*} + \mathbb{E}\|\mathcal{V}^0 - \mathcal{B}(\mathbf{x}^0)\|^2. \tag{A.4}$$

□

A.3 Proof of Lemma 2

Proof of Lemma 2. For $k = k_0$, we have

$$\begin{aligned} & \mathbb{E}_{k_0}\|\mathbf{v}^{k_0} - \nabla f(\mathbf{x}^{k_0})\|^2 \\ = & \mathbb{E}_{k_0}\|\nabla f_{S_1}(\mathbf{x}^{k_0}) - \nabla f(\mathbf{x}^{k_0})\|^2 \leq \frac{\sigma^2}{S_1} = \frac{\epsilon^2}{2}. \end{aligned} \tag{A.5}$$

From Line 14 of Algorithm 1 we have for all $k \geq 0$,

$$\|\mathbf{x}^{k+1} - \mathbf{x}^k\| = \min\left(\frac{\epsilon}{Ln_0\|\mathbf{v}^k\|}, \frac{1}{2Ln_0}\right)\|\mathbf{v}^k\| \leq \frac{\epsilon}{Ln_0}. \tag{A.6}$$

Applying Lemma 1 with $\epsilon_1 = \epsilon/(Ln_0)$, $S_2 = 2\sigma/(\epsilon n_0)$, $K = k - k_0 \leq q = \sigma n_0/\epsilon$, we have

$$\mathbb{E}_{k_0}\|\mathbf{v}^k - \nabla f(\mathbf{x}^k)\|^2 \leq \frac{\sigma n_0 L^2}{\epsilon} \cdot \frac{\epsilon^2}{L^2 n_0^2} \cdot \frac{\epsilon n_0}{2\sigma} + \mathbb{E}_{k_0}\|\mathbf{v}^{k_0} - \nabla f(\mathbf{x}^{k_0})\|^2 \overset{(A.5)}{=} \epsilon^2, \tag{A.7}$$

completing the proof. □

B Proof of Expectation Results for FSP

This section devotes to the proofs of Theorems 1, 2. To prepare for them, we first conclude via standard analysis the following

Lemma 3. *Under the Assumption 1, setting $k_0 = \lfloor k/q \rfloor \cdot q$, we have*

$$\mathbb{E}_{k_0}\left[f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k)\right] \leq -\frac{\epsilon}{4Ln_0}\mathbb{E}_{k_0}\left\|\mathbf{v}^k\right\| + \frac{3\epsilon^2}{4n_0 L}. \tag{B.1}$$

Proof of Lemma 3. From Assumption 1 (ii), we have

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|^2 = \|\mathbb{E}_i\left(\nabla f_i(\mathbf{x}) - \nabla f_i(\mathbf{y})\right)\|^2 \leq \mathbb{E}_i\|\nabla f_i(\mathbf{x}) - \nabla f_i(\mathbf{y})\|^2 \leq L^2\|\mathbf{x} - \mathbf{y}\|^2. \tag{B.2}$$

So $f(\mathbf{x})$ has L-Lipschitz continuous gradient, then

$$\begin{aligned}
f(\mathbf{x}^{k+1}) &\leq f(\mathbf{x}^k) + \left\langle \nabla f(\mathbf{x}^k), \mathbf{x}^{k+1} - \mathbf{x}^k \right\rangle + \frac{L}{2}\left\|\mathbf{x}^{k+1} - \mathbf{x}^k\right\|^2 \\
&= f(\mathbf{x}^k) - \eta^k \left\langle \nabla f(\mathbf{x}^k), \mathbf{v}^k \right\rangle + \frac{L(\eta^k)^2}{2}\left\|\mathbf{v}^k\right\|^2 \\
&= f(\mathbf{x}^k) - \eta^k\left(1 - \frac{\eta^k L}{2}\right)\left\|\mathbf{v}^k\right\|^2 - \eta^k\left\langle \nabla f(\mathbf{x}^k) - \mathbf{v}^k, \mathbf{v}^k \right\rangle \\
&\overset{a}{\leq} f(\mathbf{x}^k) - \eta^k\left(\frac{1}{2} - \frac{\eta^k L}{2}\right)\left\|\mathbf{v}^k\right\|^2 + \frac{\eta^k}{2}\left\|\mathbf{v}^k - \nabla f(\mathbf{x}^k)\right\|^2,
\end{aligned} \tag{B.3}$$

where in $\overset{a}{\leq}$, we applied Cauchy-Schwarz inequality. Since $\eta^k = \min\left(\frac{\epsilon}{Ln_0\|\mathbf{v}^k\|}, \frac{1}{2Ln_0}\right) \leq \frac{1}{2Ln_0} \leq \frac{1}{2L}$, we have

$$\eta^k\left(\frac{1}{2} - \frac{\eta^k L}{2}\right)\left\|\mathbf{v}^k\right\|^2 \geq \frac{1}{4}\eta^k\left\|\mathbf{v}^k\right\|^2 = \frac{\epsilon^2}{8n_0 L}\min\left(2\left\|\frac{\mathbf{v}^k}{\epsilon}\right\|, \left\|\frac{\mathbf{v}^k}{\epsilon}\right\|^2\right) \overset{a}{\geq} \frac{\epsilon\|\mathbf{v}^k\| - 2\epsilon^2}{4n_0 L}, \tag{B.4}$$

where in $\overset{a}{\geq}$, we use $V(x) = \min\left(|x|, \frac{x^2}{2}\right) \geq |x| - 2$ for all x. Hence

$$\begin{aligned}
f(\mathbf{x}^{k+1}) &\leq f(\mathbf{x}^k) - \frac{\epsilon\|\mathbf{v}^k\|}{4Ln_0} + \frac{\epsilon^2}{2n_0 L} + \frac{\eta^k}{2}\left\|\mathbf{v}^k - \nabla f(\mathbf{x}^k)\right\|^2 \\
&\overset{\eta^k \leq \frac{1}{2Ln_0}}{\leq} f(\mathbf{x}^k) - \frac{\epsilon\|\mathbf{v}^k\|}{4Ln_0} + \frac{\epsilon^2}{2n_0 L} + \frac{1}{4Ln_0}\left\|\mathbf{v}^k - \nabla f(\mathbf{x}^k)\right\|^2.
\end{aligned} \tag{B.5}$$

Taking expectation on the above display and using Lemma 2, we have

$$\mathbb{E}_{k_0} f(\mathbf{x}^{k+1}) - \mathbb{E}_{k_0} f(\mathbf{x}^k) \leq -\frac{\epsilon}{4Ln_0}\mathbb{E}_{k_0}\left\|\mathbf{v}^k\right\| + \frac{3\epsilon^2}{4Ln_0}. \tag{B.6}$$

□

The proof is done via the following lemma:

Lemma 4. *Under Assumption 1, for all $k \geq 0$, we have*

$$\mathbb{E}\|\nabla f(\mathbf{x}^k)\| \leq \mathbb{E}\|\mathbf{v}^k\| + \epsilon. \tag{B.7}$$

Proof. By taking the total expectation in Lemma 2, we have

$$\mathbb{E}\|\mathbf{v}^k - \nabla f(\mathbf{x}^k)\|^2 \leq \epsilon^2. \tag{B.8}$$

Then by Jensen's inequality

$$\left(\mathbb{E}\|\mathbf{v}^k - \nabla f(\mathbf{x}^k)\|\right)^2 \leq \mathbb{E}\|\mathbf{v}^k - \nabla f(\mathbf{x}^k)\|^2 \leq \epsilon^2.$$

So using triangle inequality

$$\begin{aligned} \mathbb{E}\|\nabla f(\mathbf{x}^k)\| &= \mathbb{E}\|\mathbf{v}^k - (\mathbf{v}^k - \nabla f(\mathbf{x}^k))\| \\ &\leq \mathbb{E}\|\mathbf{v}^k\| + \mathbb{E}\|\mathbf{v}^k - \nabla f(\mathbf{x}^k)\| \leq \mathbb{E}\|\mathbf{v}^k\| + \epsilon. \end{aligned} \tag{B.9}$$

This completes our proof. □

Now, we are ready to prove Theorem 1.

Proof of Theorem 1. Taking full expectation on Lemma 3, and telescoping the results from $k = 0$ to $K - 1$, we have

$$\frac{\epsilon}{4Ln_0}\sum_{k=0}^{K-1}\mathbb{E}\|\mathbf{v}^k\| \leq f(\mathbf{x}^0) - \mathbb{E}f(\mathbf{x}^K) + \frac{3K\epsilon^2}{4Ln_0} \overset{\mathbb{E}f(\mathbf{x}^K)\geq f^*}{\leq} \Delta + \frac{3K\epsilon^2}{4Ln_0}. \tag{B.10}$$

Diving $\frac{\epsilon}{4Ln_0}K$ both sides of (B.10), and using $K = \lfloor \frac{4L\Delta n_0}{\epsilon^2} \rfloor + 1 \geq \frac{4L\Delta n_0}{\epsilon^2}$, we have

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\mathbf{v}^k\| \leq \Delta \cdot \frac{4Ln_0}{\epsilon}\frac{1}{K} + 3\epsilon \leq 4\epsilon. \tag{B.11}$$

Then from the choose of $\tilde{\mathbf{x}}$, we have

$$\mathbb{E}\|\nabla f(\tilde{\mathbf{x}})\| = \frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\nabla f(\mathbf{x}^k)\| \overset{\text{(B.7)}}{\leq} \frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\mathbf{v}^k\| + \epsilon \overset{\text{(B.11)}}{\leq} 5\epsilon. \tag{B.12}$$

To compute the gradient cost, note in each q iterations we access for one time S_1 stochastic gradients and for q times of $2S_2$ stochastic gradients, and hence the cost is

$$\begin{aligned} \left\lceil K \cdot \frac{1}{q} \right\rceil S_1 + 2KS_2 &\overset{S_1 = qS_2}{\leq} 3K \cdot S_2 + S_1 \\ &\leq 3\left(\frac{4Ln_0\Delta}{\epsilon^2}\right)\frac{2\sigma}{\epsilon n_0} + \frac{2\sigma^2}{\epsilon^2} + 2S_2 \\ &= \frac{24L\sigma\Delta}{\epsilon^3} + \frac{2\sigma^2}{\epsilon^2} + \frac{4\sigma}{n_0\epsilon}. \end{aligned} \tag{B.13}$$

This concludes a gradient cost of $24L\Delta\sigma\epsilon^{-3} + 2\sigma^2\epsilon^{-2} + 4\sigma n_0^{-1}\epsilon^{-1}$. □

Proof of Theorem 2. For Lemma 2, we have

$$\mathbb{E}_{k_0}\|\mathbf{v}^{k_0} - \nabla f(\mathbf{x}^{k_0})\|^2 = \mathbb{E}_{k_0}\|\nabla f(\mathbf{x}^{k_0}) - \nabla f(\mathbf{x}^{k_0})\|^2 = 0. \tag{B.14}$$

With the above display, applying Lemma 1 with $\epsilon_1 = \frac{\epsilon}{Ln_0}$, and $S_2 = \frac{n^{1/2}}{\epsilon n_0}$, $K = k - k_0 \leq q = n_0 n^{1/2}$, we have

$$\mathbb{E}_{k_0}\|\mathbf{v}^k - \nabla f(\mathbf{x}^k)\|^2 \leq n_0 n^{1/2} L^2 \cdot \frac{\epsilon^2}{L^2 n_0^2} \cdot \frac{\epsilon n_0}{n^{1/2}} + \mathbb{E}_{k_0}\|\mathbf{v}^{k_0} - \nabla f(\mathbf{x}^{k_0})\|^2 \overset{\text{(A.5)}}{=} \epsilon^2. \tag{B.15}$$

So Lemma 2 holds. Then from the same technique of online case, we can obtain (A.6) and (4), and (B.12). The gradient cost analysis is computed as:

$$\begin{aligned}\left\lceil K \cdot \frac{1}{q} \right\rceil S_1 + 2KS_2 &\overset{S_1=qS_2}{\leq} 3K \cdot S_2 + S_1 \\ &\leq 3\left(\frac{4Ln_0\Delta}{\epsilon^2}\right)\frac{n^{1/2}}{n_0} + n + 2S_2 \\ &= \frac{12(L\Delta)\cdot n^{1/2}}{\epsilon^2} + n + \frac{2n^{1/2}}{n_0}. \end{aligned} \tag{B.16}$$

This concludes a gradient cost of $n + 12(L\Delta)\cdot n^{1/2}\epsilon^{-2} + 2n_0^{-1}n^{1/2}$. □

C Proof of Theorem 3 for Lower Bound

Our proof is a direct extension of Carmon et al. [10]. Before we drill into the proof of Theorem 3, we first introduce the hard instance $\tilde{f}_K$ with $K \geq 1$ constructed by Carmon et al. [10].

$$\hat{f}_K(\mathbf{x}) \coloneqq -\Psi(1)\Phi(x_1) + \sum_{i=2}^{K}\left[\Psi(-x_{i-1})\Phi(-x_i) - \Psi(x_{i-1})\Phi(x_i)\right], \tag{C.1}$$

where the component functions are

$$\Psi(x) \coloneqq \begin{cases} 0 & x \leq \frac{1}{2} \\ \exp\left(1 - \frac{1}{(2x-1)^2}\right) & x > \frac{1}{2} \end{cases} \tag{C.2}$$

and

$$\Phi(x) \coloneqq \sqrt{e}\int_{-\infty}^{x} e^{-\frac{t^2}{2}}, \tag{C.3}$$

where x_i denote the value of i-th coordinate of $\mathbf{x}$, with $i \in [d]$. $\hat{f}_K(\mathbf{x})$ constructed by Carmon et al. [10] is a zero-chain function, that is for every $i \in [d]$, $\nabla_i f(\mathbf{x}) = 0$ whenever $x_{i-1} = x_i = x_{i+1}$. So any deterministic algorithm can only recover "one" dimension in each iteration [10]. In addition, it satisfies that : If $|x_i| \leq 1$ for any $i \leq K$,

$$\left\|\nabla \hat{f}_K(\mathbf{x})\right\| \geq 1. \tag{C.4}$$

Then to handle random algorithms, Carmon et al. [10] further consider the following extensions:

$$\tilde{f}_{K,\mathbf{B}^K}(\mathbf{x}) = \hat{f}_K\left((\mathbf{B}^K)^{\mathrm{T}}\rho(\mathbf{x})\right) + \frac{1}{10}\|\mathbf{x}\|^2 = \hat{f}_K\left(\left\langle \mathbf{b}^{(1)}, \rho(\mathbf{x})\right\rangle, \ldots, \left\langle \mathbf{b}^{(K)}, \rho(\mathbf{x})\right\rangle\right) + \frac{1}{10}\|\mathbf{x}\|^2, \tag{C.5}$$

where $\rho(\mathbf{x}) = \frac{\mathbf{x}}{\sqrt{1+\|\mathbf{x}\|^2/R^2}}$ and $R = 230\sqrt{K}$, $\mathbf{B}^K$ is chosen uniformly at random from the space of orthogonal matrices $\mathcal{O}(d,K) = \{\mathbf{D} \in \mathbb{R}^{d\times K} | \mathbf{D}^\top\mathbf{D} = I_K\}$. The function $\tilde{f}_{K,\mathbf{B}}(\mathbf{x})$ satisfies the following:

(i)

$$\tilde{f}_{K,\mathbf{B}^K}(\mathbf{0}) - \inf_{\mathbf{x}} \tilde{f}_{K,\mathbf{B}^K}(\mathbf{x}) \leq 12K. \tag{C.6}$$

(ii) $\tilde{f}_{K,\mathbf{B}^K}(\mathbf{x})$ has constant l (independent of K and d) Lipschitz continuous gradient.

(iii) if $d \geq 52 \cdot 230^2 K^2 \log(\frac{2K^2}{p})$, for any algorithm $\mathcal{A}$ solving (1.2) with $n = 1$, and $f(\mathbf{x}) = \tilde{f}_{K,\mathbf{B}^K}(\mathbf{x})$, then with probability $1 - p$,

$$\left\|\nabla \tilde{f}_{K,\mathbf{B}^K}(\mathbf{x}^k)\right\| \geq \frac{1}{2}, \quad \text{for every } k \leq K. \tag{C.7}$$

The above properties found by Carmon et al. [10] is very technical. One can refer to Carmon et al. [10] for more details.

Proof of Theorem 3. Our lower bound theorem proof is as follows. The proof mirrors Theorem 2 in Carmon et al. [10] by further taking the number of individual function n into account. Set

$$f_i(\mathbf{x}) \coloneqq \frac{ln^{1/2}\epsilon^2}{L}\tilde{f}_{K,\mathbf{B}_i^K}(\mathbf{C}_i^{\mathrm{T}}\mathbf{x}/b) = \frac{ln^{1/2}\epsilon^2}{L}\left(\hat{f}_K\left((\mathbf{B}_i^K)^{\mathrm{T}}\rho(\mathbf{C}_i^{\mathrm{T}}\mathbf{x}/b)\right) + \frac{1}{10}\left\|\mathbf{C}_i^{\mathrm{T}}\mathbf{x}/b\right\|^2\right), \tag{C.8}$$

and

$$f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}). \tag{C.9}$$

where $\mathbf{B}^{nK} = [\mathbf{B}_1^K, \ldots, \mathbf{B}_n^K]$ is chosen uniformly at random from the space of orthogonal matrices $\mathcal{O}(d, K) = \{\mathbf{D} \in \mathbb{R}^{(d/n)\times(nK)} | \mathbf{D}^\top\mathbf{D} = I_{(nK)}\}$, with each $\mathbf{B}_i^K \in \{\mathbf{D} \in \mathbb{R}^{(d/n)\times(K)} | \mathbf{D}^\top\mathbf{D} = I_{(K)}\}$, $i \in [n]$, $\mathbf{C} = [\mathbf{C}_1, \ldots, \mathbf{C}_n]$ is an arbitrary orthogonal matrices $\mathcal{O}(d, K) = \{\mathbf{D} \in \mathbb{R}^{d\times d} | \mathbf{D}^\top\mathbf{D} = I_d\}$, with each $\mathbf{C}_i^K \in \{\mathbf{D} \in \mathbb{R}^{(d)\times(d/n)} | \mathbf{D}^\top\mathbf{D} = I_{(d/n)}\}$, $i \in [n]$. $K = \frac{\Delta L}{12ln^{1/2}\epsilon^2}$, with $n \leq \frac{144\Delta^2 L^2}{l^2\epsilon^4}$ (to ensure $K \geq 1$), $b = \frac{l\epsilon}{L}$, and $R = \sqrt{230K}$. We first verify that $f(\mathbf{x})$ satisfies Assumption 1 (i). For Assumption 1 (i), from (C.6), we have

$$f(\mathbf{0}) - \inf_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) \leq \frac{1}{n}\sum_{i=1}^{n}(f_i(\mathbf{0}) - \inf_{\mathbf{x}\in\mathbb{R}^d} f_i(\mathbf{x})) \leq \frac{ln^{1/2}\epsilon^2}{L}12K = \frac{ln^{1/2}\epsilon^2}{L}\frac{12\Delta L}{12ln^{1/2}\epsilon^2} = \Delta^{6}.$$

For Assumption 1(ii), for any i, using the $\tilde{f}_{K,\mathbf{B}_i^K}$ has l-Lipschitz continuous gradient, we have

$$\left\|\nabla \tilde{f}_{K,\mathbf{B}_i^K}(\mathbf{C}_i^{\mathrm{T}}\mathbf{x}/b) - \nabla \tilde{f}_{K,\mathbf{B}_i^K}(\mathbf{C}_i^{\mathrm{T}}\mathbf{y}/b)\right\|^2 \leq l^2\left\|\mathbf{C}_i^{\mathrm{T}}(\mathbf{x}-\mathbf{y})/b\right\|^2, \tag{C.10}$$

Because $\|\nabla f_i(\mathbf{x}) - \nabla f_i(\mathbf{y})\|^2 = \left\|\frac{ln^{1/2}\epsilon^2}{Lb}\mathbf{C}_i\left(\nabla \tilde{f}_{K,\mathbf{B}_i^K}(\mathbf{C}_i^{\mathrm{T}}\mathbf{x}/b) - \nabla \tilde{f}_{K,\mathbf{B}_i^K}(\mathbf{C}_i^{\mathrm{T}}\mathbf{y}/b)\right)\right\|^2$, and using $\mathbf{C}_i^\top\mathbf{C}_i = I_{d/n}$, we have

$$\|\nabla f_i(\mathbf{x}) - \nabla f_i(\mathbf{y})\|^2 \leq \left(\frac{ln^{1/2}\epsilon^2}{L}\right)^2\frac{l^2}{b^4}\left\|\mathbf{C}_i^{\mathrm{T}}(\mathbf{x}-\mathbf{y})\right\|^2 = nL^2\left\|\mathbf{C}_i^{\mathrm{T}}(\mathbf{x}-\mathbf{y})\right\|^2, \tag{C.11}$$

where we use $b = \frac{l\epsilon}{L}$. Summing $i = 1, \ldots, n$ and using each $\mathbf{C}_i$ are orthogonal matrix, we have

$$\mathbb{E}\|\nabla f_i(\mathbf{x}) - \nabla f_i(\mathbf{y})\|^2 \leq L^2\|\mathbf{x}-\mathbf{y}\|^2. \tag{C.12}$$

Then with

$$d \geq 2\max(9n^3K^2, 12n^2KR^2)\log\left(\frac{2n^3K^2}{p}\right) + n^2K \sim \mathcal{O}\left(\frac{n^2\Delta^2L^2}{\epsilon^4}\log\left(\frac{n^2\Delta^2L^2}{\epsilon^4 p}\right)\right),$$

from Lemma 2 of Carmon et al. [10] (or similarly Lemma 7 of Woodworth & Srebro [37] and Theorem 3 of Woodworth & Srebro [36], also refer to Lemma 5 in the end of the paper), with probability at least $1 - p$, after $T = \frac{nK}{2}$ iterations (at the end of iteration $T - 1$), for all I_i^{T-1} with $i \in [d]$, if $I_i^{T-1} < K$, then for any $j_i \in \{I_i^{T-1} + 1, \ldots, K\}$, we have $\left\langle \mathbf{b}_{i,j_i}, \rho(\mathbf{C}_i^{\mathrm{T}}\mathbf{x}/b)\right\rangle \leq \frac{1}{2}$, where I_i^{T-1} denotes that the algorithm $\mathcal{A}$ has called individual function i with I_i^{T-1} times ($\sum_{i=1}^n I_i^{T-1} = T$)

[6] If $\mathbf{x}^0 \neq \mathbf{0}$, we can simply translate the counter example as $f'(\mathbf{x}) = f(\mathbf{x} - \mathbf{x}^0)$, then $f'(\mathbf{x}^0) - \inf_{\mathbf{x}\in\mathbb{R}^d} f'(\mathbf{x}) \leq \Delta$.

at the end of iteration $T-1$, and $\mathbf{b}_{i,j}$ denotes the j-th column of $\mathbf{B}_i^K$. However, from (C.7), if $\left\langle \mathbf{b}_{i,j_i}, \rho(\mathbf{C}_i^{\mathrm{T}}\mathbf{x}/b)\right\rangle \leq \frac{1}{2}$, we will have $\|\nabla \tilde{f}_{K,\mathbf{B}_i^K}(\mathbf{C}_i^{\mathrm{T}}\mathbf{x}/b)\| \geq \frac{1}{2}$. So f_i can be solved only after K times calling it.

From the above analysis, for any algorithm $\mathcal{A}$, after running $T = \frac{nK}{2} = \frac{\Delta L n^{1/2}}{24l\epsilon^2}$ iterations, at least $\frac{n}{2}$ functions cannot be solved (the worst case is when $\mathcal{A}$ exactly solves $\frac{n}{2}$ functions), so

$$\begin{aligned}\left\|\nabla f(\mathbf{x}^{nK/2})\right\|^2 &= \frac{1}{n^2}\left\|\sum_{i \text{ not solved}} \frac{ln^{1/2}\epsilon^2}{Lb}\mathbf{C}_i \nabla \tilde{f}_{K,\mathbf{B}_i^K}(\mathbf{C}_i^{\mathrm{T}}\mathbf{x}^{nK/2}/b)\right\|^2 \\ &\overset{a}{=} \frac{1}{n^2}\sum_{i \text{ not solved}}\left\|n^{1/2}\epsilon \nabla \tilde{f}_{K,\mathbf{B}_i^K}(\mathbf{C}_i^{\mathrm{T}}\mathbf{x}^{nK/2}/b)\right\|^2 \overset{\text{(C.7)}}{\geq} \frac{\epsilon^2}{8},\end{aligned} \tag{C.13}$$

where in $\overset{a}{=}$, we use $\mathbf{C}_i^{\top}\mathbf{C}_j = \mathbf{0}_{d/n}$, when $i \neq j$, and $\mathbf{C}_i^{\top}\mathbf{C}_i = I_{d/n}$. □

Lemma 5. *Let $\{\mathbf{x}\}_{0:T}$ with $T = \frac{nK}{2}$ is informed by a certain algorithm in the form* (3.7)*. Then when $d \geq 2\max(9n^3K^2, 12n^3KR^2)\log(\frac{2n^2K^2}{p}) + n^2K$, with probability $1-p$, at each iteration $0 \leq t \leq T$, $\mathbf{x}^t$ can only recover one coordinate.*

Proof. The proof is essentially same to [10] and [36]. We give a proof here. Before the poof, we give the following definitions:

1. Let i^t denotes that at iteration t, the algorithm choses the i^t-th individual function.
2. Let I_i^t denotes the total times that individual function with index i has been called before iteration k. We have $I_i^0 = 0$ with $i \in [n]$, $i \neq i^t$, and $I_{i^0}^0 = 1$. And for $t \geq 1$,
$$I_i^t = \begin{cases} I_i^{t-1} + 1, & i = i_t. \\ I_i^{t-1}, & \text{otherwise.}\end{cases} \tag{C.14}$$
3. Let $\mathbf{y}_i^t = \rho(\mathbf{C}_i^{\mathrm{T}}\mathbf{x}^t) = \frac{\mathbf{C}_i^{\mathrm{T}}\mathbf{x}^t}{\sqrt{R^2 + \|\mathbf{C}_i^{\mathrm{T}}\mathbf{x}^t\|^2}}$ with $i \in [n]$. We have $\mathbf{y}_i^t \in \mathbb{R}^{d/n}$ and $\|\mathbf{y}_i^t\| \leq R$.
4. Set $\boldsymbol{\mathcal{V}}_i^t$ be the set that $\left(\bigcup_{i=1}^n \left\{\mathbf{b}_{i,1}, \cdots \mathbf{b}_{i,\min(K,I_i^t)}\right\}\right) \bigcup \left\{\mathbf{y}_i^0, \mathbf{y}_i^1, \cdots, \mathbf{y}_i^t\right\}$, where $\mathbf{b}_{i,j}$ denotes the j-th column of $\mathbf{B}_i^K$.
5. Set $\boldsymbol{\mathcal{U}}_i^t$ be the set of $\left\{\mathbf{b}_{i,\min(K,I_i^{t-1}+1)}, \cdots, \mathbf{b}_{i,K}\right\}$ with $i \in [n]$. $\boldsymbol{\mathcal{U}}^t = \bigcup_{i=1}^n \boldsymbol{\mathcal{U}}_i^t$. And set $\tilde{\boldsymbol{\mathcal{U}}}_i^t = \left\{\mathbf{b}_{i,\min(K,1)}, \cdots, \mathbf{b}_{i,\min(K,I_i^{t-1})}\right\}$. $\tilde{\boldsymbol{\mathcal{U}}}^t = \bigcup_{i=1}^n \tilde{\boldsymbol{\mathcal{U}}}_i^t$.
6. Let $\boldsymbol{\mathcal{P}}_i^t \in \mathcal{R}^{(d/n)\times(d/n)}$ denote the projection operator to the span of $\mathbf{u} \in \boldsymbol{\mathcal{V}}_i^t$. And let $\boldsymbol{\mathcal{P}}_i^{t\perp}$ denote its orthogonal complement.

Because $\mathcal{A}^t$ performs measurable mapping, the above terms are all measurable on $\boldsymbol{\xi}$ and $\mathbf{B}^{nK}$, where $\boldsymbol{\xi}$ is the random vector in $\mathcal{A}$. It is clear that if for all $0 \leq t \leq T$ and $i \in [n]$, we have

$$\left|\left\langle \mathbf{u}, \mathbf{y}_i^t\right\rangle\right| < \frac{1}{2}, \quad \text{for all } \mathbf{u} \in \boldsymbol{\mathcal{U}}_i^t. \tag{C.15}$$

then at each iteration, we can only recover one index, which is our destination. To prove that (C.15) holds with probability at least $1-p$, we consider a more hard event $\boldsymbol{\mathcal{G}}^t$ as

$$\boldsymbol{\mathcal{G}}^t = \left\{\left|\left\langle \mathbf{u}, \boldsymbol{\mathcal{P}}_i^{(t-1)\perp}\mathbf{y}_i^t\right\rangle\right| \leq a\|\boldsymbol{\mathcal{P}}_i^{(t-1)\perp}\mathbf{y}_i^t\| \mid \mathbf{u} \in \boldsymbol{\mathcal{U}}^t \text{ (not } \boldsymbol{\mathcal{U}}_i^t),\ i \in [n]\right\}, \quad t \geq 1, \tag{C.16}$$

with $a = \min\left(\frac{1}{3(T+1)}, \frac{1}{2(1+\sqrt{3T})R}\right)$. And $G^{\leq t} = \bigcap_{j=0}^t \boldsymbol{\mathcal{G}}^j$.

We first show that if $\mathcal{G}^{\leq T}$ happens, then (C.15) holds for all $0 \leq t \leq T$. For $0 \leq t \leq T$, and $i \in [n]$, if $\boldsymbol{\mathcal{U}}_i^t = \varnothing$, (C.15) is right; otherwise for any $\mathbf{u} \in \boldsymbol{\mathcal{U}}_i^t$, we have

$$\begin{aligned} & \left|\left\langle \mathbf{u}, \mathbf{y}_i^t \right\rangle\right| \\ \leq & \left|\left\langle \mathbf{u}, \boldsymbol{\mathcal{P}}_i^{(t-1)\perp} \mathbf{y}_i^t \right\rangle\right| + \left|\left\langle \mathbf{u}, \boldsymbol{\mathcal{P}}_i^{(t-1)} \mathbf{y}_i^t \right\rangle\right| \\ \leq & a\|\boldsymbol{\mathcal{P}}_i^{(t-1)\perp} \mathbf{y}_i^t\| + \left|\left\langle \mathbf{u}, \boldsymbol{\mathcal{P}}_i^{t-1} \mathbf{y}_i^t \right\rangle\right| \leq aR + R\left\|\boldsymbol{\mathcal{P}}_i^{t-1}\mathbf{u}\right\|, \end{aligned} \tag{C.17}$$

where in the last inequality, we use $\|\boldsymbol{\mathcal{P}}_i^{(t-1)\perp} \mathbf{y}_i^t\| \leq \|\mathbf{y}_i^{(t-1)}\| \leq R$.

If $t = 0$, we have $\boldsymbol{\mathcal{P}}_i^{t-1} = \mathbf{0}_{d/n \times d/n}$, then $\left\|\boldsymbol{\mathcal{P}}_i^{t-1}\mathbf{u}\right\| = 0$, so (C.15) holds. When $t \geq 1$, suppose at $t-1$, $\mathcal{G}^{\leq t}$ happens then (C.15) holds for all 0 to $t-1$. Then we need to prove that $\|\boldsymbol{\mathcal{P}}_i^{t-1}\mathbf{u}\| \leq b = \sqrt{3T}a$ with $\mathbf{u} \in \boldsymbol{\mathcal{U}}_i^t$ and $i \in [n]$. Instead, we prove a stronger results: $\|\boldsymbol{\mathcal{P}}_i^{t-1}\mathbf{u}\| \leq b = \sqrt{3T}a$ with all $\mathbf{u} \in \boldsymbol{\mathcal{U}}^t$ and $i \in [n]$. Again, When $t = 0$, we have $\|\boldsymbol{\mathcal{P}}_i^{t-1}\mathbf{u}\| = 0$, so it is right, when $t \geq 1$, by Graham-Schmidt procedure on $\mathbf{y}_i^0, \mathbf{b}_{i_0,\min(I_{i^0}^0,K)}, \cdots, \mathbf{y}_i^{t-1}, \mathbf{b}_{i_{t-1},\min(I_{i^{t-1}}^{t-1},K)}$, we have

$$\left\|\boldsymbol{\mathcal{P}}_i^{t-1}\mathbf{u}\right\|^2 = \sum_{z=0}^{t-1}\left|\left\langle \frac{\boldsymbol{\mathcal{P}}_i^{(z-1)\perp}\mathbf{y}_i^z}{\|\boldsymbol{\mathcal{P}}_i^{(z-1)\perp}\mathbf{y}_i^z\|}, \mathbf{u} \right\rangle\right|^2 + \sum_{z=0,\ I_{i^z}^z \leq K}^{t-1}\left|\left\langle \frac{\hat{\boldsymbol{\mathcal{P}}}_i^{(z-1)\perp}\mathbf{b}_{i_z,I_{i^z}^z}}{\|\hat{\boldsymbol{\mathcal{P}}}_i^{(z-1)\perp}\mathbf{b}_{i_z,I_{i^z}^z}\|}, \mathbf{u} \right\rangle\right|^2, \tag{C.18}$$

where

$$\hat{\boldsymbol{\mathcal{P}}}_i^{(z-1)} = \boldsymbol{\mathcal{P}}_i^{(z-1)} + \frac{\left(\boldsymbol{\mathcal{P}}_i^{(z-1)\perp}\mathbf{y}_i^z\right)\left(\boldsymbol{\mathcal{P}}_i^{(z-1)\perp}\mathbf{y}_i^z\right)^{\mathrm{T}}}{\left\|\boldsymbol{\mathcal{P}}_i^{(z-1)\perp}\mathbf{y}_i^z\right\|^2}.$$

Using $\mathbf{b}_{i_z,I_{i^z}^z} \perp \mathbf{u}$ for all $\mathbf{u} \in \boldsymbol{\mathcal{U}}^t$, we have

$$\begin{aligned} & \left|\left\langle \hat{\boldsymbol{\mathcal{P}}}_i^{(z-1)\perp}\mathbf{b}_{i_z,I_{i^z}^z}, \mathbf{u} \right\rangle\right| \\ = & \left|0 - \left\langle \hat{\boldsymbol{\mathcal{P}}}_i^{(z-1)}\mathbf{b}_{i_z,I_{i^z}^z}, \mathbf{u} \right\rangle\right| \\ \leq & \left|\left\langle \boldsymbol{\mathcal{P}}_i^{(z-1)}\mathbf{b}_{i_z,I_{i^z}^z}, \mathbf{u} \right\rangle\right| + \left|\left\langle \frac{\boldsymbol{\mathcal{P}}_i^{(z-1)\perp}\mathbf{y}_i^z}{\|\boldsymbol{\mathcal{P}}_i^{(z-1)\perp}\mathbf{y}_i^z\|}, \mathbf{b}_{i_z,I_{i^z}^z} \right\rangle \left\langle \frac{\boldsymbol{\mathcal{P}}_i^{(z-1)\perp}\mathbf{y}_i^z}{\|\boldsymbol{\mathcal{P}}_i^{(z-1)\perp}\mathbf{y}_i^z\|}, \mathbf{u} \right\rangle\right|. \end{aligned} \tag{C.19}$$

For the first term in the right hand of (C.19), by induction, we have

$$\left|\left\langle \boldsymbol{\mathcal{P}}_i^{(z-1)}\mathbf{b}_{i_z,I_{i^z}^z}, \mathbf{u} \right\rangle\right| = \left|\left\langle \boldsymbol{\mathcal{P}}_i^{(z-1)}\mathbf{b}_{i_z,I_{i^z}^z}, \boldsymbol{\mathcal{P}}_i^{(z-1)}\mathbf{u} \right\rangle\right| \leq b^2. \tag{C.20}$$

For the second term in the right hand of (C.19), by assumption (C.16), we have

$$\left|\left\langle \frac{\boldsymbol{\mathcal{P}}_i^{(z-1)\perp}\mathbf{y}_i^z}{\|\boldsymbol{\mathcal{P}}_i^{(z-1)\perp}\mathbf{y}_i^z\|}, \mathbf{b}_{i_z,I_{i^z}^z} \right\rangle \left\langle \frac{\boldsymbol{\mathcal{P}}_i^{(z-1)\perp}\mathbf{y}_i^z}{\|\boldsymbol{\mathcal{P}}_i^{(z-1)\perp}\mathbf{y}_i^z\|}, \mathbf{u} \right\rangle\right| \leq a^2. \tag{C.21}$$

Also, we have

$$\begin{aligned} & \left\|\hat{\boldsymbol{\mathcal{P}}}_i^{(z-1)\perp}\mathbf{b}_{i_z,I_{i^z}^z}\right\|^2 \\ = & \|\mathbf{b}_{i_z,I_{i^z}^z}\|^2 - \left\|\hat{\boldsymbol{\mathcal{P}}}_i^{(z-1)}\mathbf{b}_{i_z,I_{i^z}^z}\right\|^2 \\ = & \|\mathbf{b}_{i_z,I_{i^z}^z}\|^2 - \left\|\boldsymbol{\mathcal{P}}_i^{(z-1)}\mathbf{b}_{i_z,I_{i^z}^z}\right\|^2 - \left|\left\langle \frac{\boldsymbol{\mathcal{P}}_i^{(z-1)\perp}\mathbf{y}_i^z}{\|\boldsymbol{\mathcal{P}}_i^{(z-1)\perp}\mathbf{y}_i^z\|}, \mathbf{b}_{i_z,I_{i^z}^z} \right\rangle\right|^2 \\ \geq & 1 - b^2 - a^2. \end{aligned} \tag{C.22}$$

Substituting (C.19) and (C.22) into (C.18), for all $\mathbf{u} \in \boldsymbol{\mathcal{U}}^t$, we have

$$\begin{aligned} \left\|\boldsymbol{\mathcal{P}}_i^{t-1}\mathbf{u}\right\|^2 &\leq ta^2 + t\frac{(a^2+b^2)^2}{1-(a^2+b^2)} \\ &\overset{a^2+b^2\leq(3T+1)a^2\leq a}{\leq} Ta^2 + T\frac{a^2}{1-a} \overset{a\leq 1/2}{\leq} 3Ta^2 = b^2. \end{aligned} \tag{C.23}$$

Thus for (C.17), $t \geq 1$, because $\mathbf{u} \in \boldsymbol{\mathcal{U}}_i^t \subseteq \boldsymbol{\mathcal{U}}^t$, we have

$$\left|\left\langle \mathbf{u}, \mathbf{y}_i^t \right\rangle\right| \leq (a+b)R \overset{a\leq\frac{1}{2(1+\sqrt{3T})R}}{\leq} \frac{1}{2}. \tag{C.24}$$

This shows that if $\boldsymbol{\mathcal{G}}^{\leq T}$ happens, (C.15) holds for all $0 \leq t \leq T$. Then we prove that $\mathbb{P}(\boldsymbol{\mathcal{G}}^{\leq T}) \geq 1-p$. We have

$$\mathbb{P}\left((\boldsymbol{\mathcal{G}}^{\leq T})^c\right) = \sum_{t=0}^{T} \mathbb{P}\left((\boldsymbol{\mathcal{G}}^{\leq t})^c \mid \boldsymbol{\mathcal{G}}^{<t}\right). \tag{C.25}$$

We give the following definition:

1. Denote $\hat{i}^t$ be the sequence of $i_{0:t-1}$. Let $\hat{\mathcal{S}}^t$ be the set that contains all possible ways of $\hat{i}^t$ ($|\hat{\mathcal{S}}^t| \leq n^t$).
2. Let $\tilde{\mathbf{U}}_{\hat{i}^t}^j = [\mathbf{b}_{j,1}, \cdots, \mathbf{b}_{j,\min(K,I_j^{t-1})}]$ with $j \in [n]$, and $\tilde{\mathbf{U}}_{\hat{i}^t} = [\tilde{\mathbf{U}}_{\hat{i}^t}^1, \cdots, \tilde{\mathbf{U}}_{\hat{i}^t}^n]$. $\tilde{\mathbf{U}}_{\hat{i}^t}$ is analogous to $\tilde{\boldsymbol{\mathcal{U}}}^t$, but is a matrix.
3. Let $\mathbf{U}_{\hat{i}^t}^j = [\mathbf{b}_{j,\min(K,I_j^t)}; \cdots; \mathbf{b}_{j,K}]$ with $j \in [n]$, and $\mathbf{U}_{\hat{i}^t} = [\mathbf{U}_{\hat{i}^t}^1, \cdots, \mathbf{U}_{\hat{i}^t}^n]$. $\mathbf{U}_{\hat{i}^t}$ is analogous to $\boldsymbol{\mathcal{U}}^t$, but is a matrix. Let $\bar{\mathbf{U}} = [\tilde{\mathbf{U}}_{\hat{i}^t}, \mathbf{U}_{\hat{i}^t}]$.

We have that

$$\begin{aligned} &\mathbb{P}\left((\boldsymbol{\mathcal{G}}^{\leq t})^c \mid \boldsymbol{\mathcal{G}}^{<t}\right) \\ =& \sum_{\hat{i}_0^t \in \hat{\mathcal{S}}^t} \mathbb{E}_{\boldsymbol{\xi}, \mathbf{U}_{\hat{i}_0^t}} \left(\mathbb{P}\left((\boldsymbol{\mathcal{G}}^{\leq t})^c \mid \boldsymbol{\mathcal{G}}^{<t}, \hat{i}^t = \hat{i}_0^t, \boldsymbol{\xi}, \mathbf{U}_{\hat{i}_0^t}\right) \mathbb{P}\left(\hat{i}^t = \hat{i}_0^t \mid \boldsymbol{\mathcal{G}}^{<t}, \boldsymbol{\xi}, \mathbf{U}_{\hat{i}_0^t}\right)\right). \end{aligned} \tag{C.26}$$

For $\sum_{\hat{i}_0^t \in \hat{\mathcal{S}}^t} \mathbb{E}_{\boldsymbol{\xi}, \mathbf{U}_{\hat{i}_0^t}} \mathbb{P}\left(\hat{i}^t = \hat{i}_0^t \mid \boldsymbol{\mathcal{G}}^{<t}, \boldsymbol{\xi}, \mathbf{U}_{\hat{i}_0^t}\right) = \sum_{\hat{i}_0^t \in \hat{\mathcal{S}}^t} \mathbb{P}\left(\hat{i}^t = \hat{i}_0^t \mid \boldsymbol{\mathcal{G}}^{<t}\right) = 1$, in the rest, we show that the probability $\mathbb{P}\left((\boldsymbol{\mathcal{G}}^{\leq t})^c \mid \boldsymbol{\mathcal{G}}^{<t}, \hat{i}^t = \hat{i}_0^t, \boldsymbol{\xi} = \boldsymbol{\xi}_0, \tilde{\mathbf{U}}_{\hat{i}_0^t} = \tilde{\mathbf{U}}_0, \right)$ for all $\xi_0, \tilde{\mathbf{U}}_0$ is small. By union bound, we have

$$\begin{aligned} &\mathbb{P}\left((\boldsymbol{\mathcal{G}}^{\leq t})^c \mid \boldsymbol{\mathcal{G}}^{<t}, \hat{i}^t = \hat{i}_0^t, \boldsymbol{\xi} = \boldsymbol{\xi}_0, \tilde{\mathbf{U}}_{\hat{i}_0^t} = \tilde{\mathbf{U}}_0\right) \\ \leq& \sum_{i=1}^{n} \sum_{\mathbf{u} \in \boldsymbol{\mathcal{U}}^t} \mathbb{P}\left(\left\langle \mathbf{u}, \boldsymbol{\mathcal{P}}_i^{(t-1)\perp} \mathbf{y}_i^t \right\rangle \geq a \|\boldsymbol{\mathcal{P}}_i^{(t-1)\perp} \mathbf{y}_i^t\| \mid \boldsymbol{\mathcal{G}}^{<t}, \hat{i}^t = \hat{i}_0^t, \boldsymbol{\xi} = \boldsymbol{\xi}_0, \tilde{\mathbf{U}}_{\hat{i}_0^t} = \tilde{\mathbf{U}}_0\right). \end{aligned} \tag{C.27}$$

Note that $\hat{i}_0^t$ is a constant. Because given $\boldsymbol{\xi}$ and $\tilde{\mathbf{U}}_{\hat{i}_0^t}$, under $G^{\leq t}$, both $\boldsymbol{\mathcal{P}}_i^{(t-1)}$ and $\mathbf{y}_i^t$ are known. We prove

$$\mathbb{P}\left(\mathbf{U}_{\hat{i}_0^t} = \mathbf{U}_0 \mid \boldsymbol{\mathcal{G}}^{<t}, \hat{i}^t = \hat{i}_0^t, \boldsymbol{\xi} = \boldsymbol{\xi}_0, \tilde{\mathbf{U}}_{\hat{i}_0^t} = \tilde{\mathbf{U}}_0\right) = \mathbb{P}\left(\mathbf{U}_{\hat{i}_0^t} = \mathbf{Z}_i \mathbf{U}_0 \mid \boldsymbol{\mathcal{G}}^{<t}, \hat{i}^t = \hat{i}_0^t, \boldsymbol{\xi} = \boldsymbol{\xi}_0, \tilde{\mathbf{U}}_{\hat{i}_0^t} = \tilde{\mathbf{U}}_0\right) \tag{C.28}$$

where $\mathbf{Z}_i \in \mathbb{R}^{d/n \times d/n}$, $\mathbf{Z}_i^{\mathrm{T}}\mathbf{Z}_i = \mathbf{I}_d$, and $\mathbf{Z}_i\mathbf{u} = \mathbf{u} = \mathbf{Z}_i^{\mathrm{T}}\mathbf{u}$ for all $\mathbf{u} \in \boldsymbol{\mathcal{V}}_i^{t-1}$. In this way, $\frac{\boldsymbol{\mathcal{P}}_i^{(t-1)\perp}\mathbf{u}}{\|\boldsymbol{\mathcal{P}}_i^{(t-1)\perp}\mathbf{u}\|}$ has uniformed distribution on the unit space. To prove it, we have

$$\begin{aligned}
&\mathbb{P}\left(\mathbf{U}_{\hat{i}_0^t} = \mathbf{U}_0 \mid \boldsymbol{\mathcal{G}}^{<t}, \hat{i}^t = \hat{i}_0^t, \boldsymbol{\xi} = \boldsymbol{\xi}_0, \tilde{\mathbf{U}}_{\hat{i}_0^t} = \tilde{\mathbf{U}}_0\right) \\
= \;& \frac{\mathbb{P}(\mathbf{U}_{\hat{i}_0^t} = \mathbf{U}_0, \boldsymbol{\mathcal{G}}^{<t}, \hat{i}^t = \hat{i}_0^t, \boldsymbol{\xi} = \boldsymbol{\xi}_0, \tilde{\mathbf{U}}_{\hat{i}_0^t} = \tilde{\mathbf{U}}_0)}{\mathbb{P}(\boldsymbol{\mathcal{G}}^{<t}, \hat{i}^t = \hat{i}_0^t, \boldsymbol{\xi} = \boldsymbol{\xi}_0, \tilde{\mathbf{U}}_{\hat{i}_0^t} = \tilde{\mathbf{U}}_0)} \\
= \;& \frac{\mathbb{P}(\boldsymbol{\mathcal{G}}^{<t}, \hat{i}^t = \hat{i}_0^t \mid \boldsymbol{\xi} = \boldsymbol{\xi}_0, \mathbf{U}_{\hat{i}_0^t} = \mathbf{U}_0, \tilde{\mathbf{U}}_{\hat{i}_0^t} = \tilde{\mathbf{U}}_0) p(\boldsymbol{\xi} = \boldsymbol{\xi}_0, \mathbf{U}_{\hat{i}_0^t} = \mathbf{U}_0, \tilde{\mathbf{U}}_{\hat{i}_0^t} = \tilde{\mathbf{U}}_0)}{\mathbb{P}(\boldsymbol{\mathcal{G}}^{<t}, \hat{i}^t = \hat{i}_0^t, \boldsymbol{\xi} = \boldsymbol{\xi}_0, \tilde{\mathbf{U}}_{\hat{i}_0^t} = \tilde{\mathbf{U}})}
\end{aligned} \tag{C.29}$$

And

$$\begin{aligned}
&\mathbb{P}\left(\mathbf{U}_{\hat{i}_0^t} = \mathbf{Z}_i\mathbf{U}_0 \mid \boldsymbol{\mathcal{G}}^{<t}, \hat{i}^t = \hat{i}_0^t, \boldsymbol{\xi} = \boldsymbol{\xi}_0, \tilde{\mathbf{U}}_{\hat{i}_0} = \tilde{\mathbf{U}}_0\right) \\
= \;& \frac{\mathbb{P}(\boldsymbol{\mathcal{G}}^{<t}, \hat{i}^t = \hat{i}_0^t \mid \boldsymbol{\xi} = \boldsymbol{\xi}_0, \mathbf{U}_{\hat{i}_0^t} = \mathbf{U}_0, \tilde{\mathbf{U}}_{\hat{i}_0^t} = \mathbf{Z}_i\tilde{\mathbf{U}}_0) p(\boldsymbol{\xi} = \boldsymbol{\xi}_0, \mathbf{U}_{\hat{i}_0^t} = \mathbf{Z}_i\mathbf{U}_0, \tilde{\mathbf{U}}_{\hat{i}_0^t} = \tilde{\mathbf{U}}_0)}{\mathbb{P}(\boldsymbol{\mathcal{G}}^{<t}, \hat{i}^t = \hat{i}_0^t, \boldsymbol{\xi} = \boldsymbol{\xi}_0, \tilde{\mathbf{U}}_{\hat{i}_0^t} = \tilde{\mathbf{U}}_0)}
\end{aligned} \tag{C.30}$$

For $\boldsymbol{\xi}$ and $\bar{\mathbf{U}}$ are independent. And $p(\bar{\mathbf{U}}) = p(\mathbf{Z}_i\bar{\mathbf{U}})$, we have $p(\boldsymbol{\xi} = \boldsymbol{\xi}_0, \mathbf{U}_{\hat{i}_0^t} = \mathbf{U}_0, \tilde{\mathbf{U}}_{\hat{i}_0^t} = \tilde{\mathbf{U}}_0) = p(\boldsymbol{\xi} = \boldsymbol{\xi}_0, \mathbf{U}_{\hat{i}_0^t} = \mathbf{Z}_i\mathbf{U}_0, \tilde{\mathbf{U}}_{\hat{i}_0^t} = \tilde{\mathbf{U}}_0)$. Then we prove that if $\boldsymbol{\mathcal{G}}^{<t}$ and $\hat{i}^t = \hat{i}_0^t$ happens under $\mathbf{U}_{\hat{i}_0^t} = \mathbf{U}_0, \boldsymbol{\xi} = \boldsymbol{\xi}_0, \tilde{\mathbf{U}}_{\hat{i}_0^t} = \tilde{\mathbf{U}}_0$, if and only if $\boldsymbol{\mathcal{G}}^{<t}$ and $\hat{i}^t = \hat{i}_0^t$ happen under $\mathbf{U}_{\hat{i}_0^t} = \mathbf{Z}_i\mathbf{U}_0, \boldsymbol{\xi} = \boldsymbol{\xi}_0, \tilde{\mathbf{U}}_{\hat{i}_0^t} = \tilde{\mathbf{U}}_0$.

Suppose at iteration $l-1$ with $l \leq t$, we have the result. At iteration l, suppose $\boldsymbol{\mathcal{G}}^{<l}$ and $\hat{i}^l = \hat{i}_0^l$ happen, given $\mathbf{U}_{\hat{i}_0^t} = \mathbf{U}_0, \boldsymbol{\xi} = \boldsymbol{\xi}_0, \tilde{\mathbf{U}}_{\hat{i}_0^t} = \tilde{\mathbf{U}}_0$. Let $\mathbf{x}'$ and $(\hat{i}')^j$ are generated by $\boldsymbol{\xi} = \boldsymbol{\xi}_0, \mathbf{U}_{\hat{i}_0^t} = \mathbf{Z}_i\mathbf{U}_0, \tilde{\mathbf{U}}_{\hat{i}_0^t} = \tilde{\mathbf{U}}_0$. Because $\boldsymbol{\mathcal{G}}^{<l}$ happens, thus at each iteration, we can only recover one index until $l-1$. Then $(\mathbf{x}')^j = \mathbf{x}^j$ and $(\hat{i}')^j = \hat{i}^j$. with $j \leq l$. By induction, we only need to prove that $\boldsymbol{\mathcal{G}}^{l-1'}$ will happen. Let $\mathbf{u} \in \boldsymbol{\mathcal{U}}^{l-1}$, and $i \in [n]$, we have

$$\left|\left\langle \mathbf{Z}_i\mathbf{u}, \frac{\boldsymbol{\mathcal{P}}_i^{(l-2)\perp}\mathbf{y}_i^{l-1}}{\|\boldsymbol{\mathcal{P}}_i^{(l-2)\perp}\mathbf{y}_i^{l-1}\|} \right\rangle\right| = \left|\left\langle \mathbf{u}, \mathbf{Z}_i^{\mathrm{T}}\frac{\boldsymbol{\mathcal{P}}_i^{(l-2)\perp}\mathbf{y}_i^{l-1}}{\|\boldsymbol{\mathcal{P}}_i^{(l-2)\perp}\mathbf{y}_i^{l-1}\|} \right\rangle\right| \overset{a}{=} \left|\left\langle \mathbf{u}, \frac{\boldsymbol{\mathcal{P}}_i^{(l-2)\perp}\mathbf{y}_i^{l-1}}{\|\boldsymbol{\mathcal{P}}_i^{(l-2)\perp}\mathbf{y}_i^{l-1}\|} \right\rangle\right|, \tag{C.31}$$

where in $\overset{a}{=}$, we use $\boldsymbol{\mathcal{P}}_i^{(l-2)\perp}\mathbf{y}_i^{l-1}$ is in the span of $\boldsymbol{\mathcal{V}}_i^l \subseteq \boldsymbol{\mathcal{V}}_i^{t-1}$. This shows that if $\boldsymbol{\mathcal{G}}^{<t}$ and $\hat{i}^t = \hat{i}_0^t$ happen under $\mathbf{U}_{\hat{i}_0^t} = \mathbf{U}_0, \boldsymbol{\xi} = \boldsymbol{\xi}_0, \tilde{\mathbf{U}}_{\hat{i}_0^t} = \tilde{\mathbf{U}}_0$, then $\boldsymbol{\mathcal{G}}^{<t}$ and $\hat{i}^t = \hat{i}^t$ happen under $\mathbf{U}_{\hat{i}_0^t} = \mathbf{Z}_i\mathbf{U}_0, \boldsymbol{\xi} = \boldsymbol{\xi}_0, \tilde{\mathbf{U}}_{\hat{i}_0^t} = \tilde{\mathbf{U}}_0$. In the same way, we can prove the necessity. Thus for any $\mathbf{u} \in \mathbf{U}^t$, if $\|\boldsymbol{\mathcal{P}}_i^{(t-1)\perp}\mathbf{y}_i^t\| \neq 0$ (otherwise, $\left|\left\langle \mathbf{u}, \boldsymbol{\mathcal{P}}_i^{(t-1)\perp}\mathbf{y}_i^t \right\rangle\right| \leq a\|\boldsymbol{\mathcal{P}}_i^{(t-1)\perp}\mathbf{y}_i^t\|$ holds), we have

$$\begin{aligned}
&\mathbb{P}\left(\left\langle \mathbf{u}, \frac{\boldsymbol{\mathcal{P}}_i^{(t-1)\perp}\mathbf{y}_i^t}{\|\boldsymbol{\mathcal{P}}_i^{(t-1)\perp}\mathbf{y}_i^t\|} \right\rangle \geq a \mid \boldsymbol{\mathcal{G}}^{<t}, \hat{i}^t = \hat{i}_0^t, \boldsymbol{\xi} = \boldsymbol{\xi}_0, \tilde{\mathbf{U}}_{\hat{i}_0^t} = \tilde{\mathbf{U}}_0\right) \\
\overset{a}{\leq} \;& \mathbb{P}\left(\left\langle \frac{\boldsymbol{\mathcal{P}}_i^{(t-1)\perp}\mathbf{u}}{\|\boldsymbol{\mathcal{P}}_i^{(t-1)\perp}\mathbf{u}\|}, \frac{\boldsymbol{\mathcal{P}}_i^{(t-1)\perp}\mathbf{y}_i^t}{\|\boldsymbol{\mathcal{P}}_i^{(t-1)\perp}\mathbf{y}_i^t\|} \right\rangle \geq a \mid \boldsymbol{\mathcal{G}}^{<t}, \hat{i}^t = \hat{i}_0^t, \boldsymbol{\xi} = \boldsymbol{\xi}_0, \tilde{\mathbf{U}}_{\hat{i}_0^t} = \tilde{\mathbf{U}}_0\right) \\
\overset{b}{\leq} \;& 2e^{\frac{-a^2(d/n-2T)}{2}},
\end{aligned} \tag{C.32}$$

where in $\overset{a}{\leq}$, we use $\|\boldsymbol{\mathcal{P}}_i^{(t-1)\perp}\mathbf{u}\| \leq 1$; and in $\overset{b}{\leq}$, we use $\frac{\boldsymbol{\mathcal{P}}_i^{(t-1)\perp}\mathbf{y}_i^t}{\|\boldsymbol{\mathcal{P}}_i^{(t-1)\perp}\mathbf{y}_i^t\|}$ is a known unit vector and $\frac{\boldsymbol{\mathcal{P}}_i^{(t-1)\perp}\mathbf{u}}{\|\boldsymbol{\mathcal{P}}_i^{(t-1)\perp}\mathbf{u}\|}$ has uniformed distribution on the unit space. Then by union bound, we have

	Algorithm		Online	Finite-Sum
First-order Stationary Point	GD / SGD	[26]	ϵ^{-4}	$n\epsilon^{-2}$
	SVRG / SCSG	[3, 24, 32]	$\epsilon^{-3.333}$	$n + n^{2/3}\epsilon^{-2}$
	SPIDER-SFO	(this work)	ϵ^{-3}	$n + n^{1/2}\epsilon^{-2}$ $^{\Delta}$
First-order Stationary Point (Hessian-Lipschitz Required)	Perturbed GD / SGD	[15, 20]	$poly(d)\epsilon^{-4}$	$n\epsilon^{-2}$
	NEON+GD / NEON+SGD	[4, 38]	ϵ^{-4}	$n\epsilon^{-2}$
	AGD	[21]	N/A	$n\epsilon^{-1.75}$
	NEON+SVRG / NEON+SCSG	[3, 24, 32]	$\epsilon^{-3.5}$ $(\epsilon^{-3.333})$	$n\epsilon^{-1.5} + n^{2/3}\epsilon^{-2}$
	NEON+FastCubic/CDHS	[1, 8, 35]	$\epsilon^{-3.5}$	$n\epsilon^{-1.5} + n^{3/4}\epsilon^{-1.75}$
	NEON+Natasha2	[2, 4, 38]	$\epsilon^{-3.5}$ $(\epsilon^{-3.25})$	$n\epsilon^{-1.5} + n^{2/3}\epsilon^{-2}$
	SPIDER-SFO$^+$	(this work)	ϵ^{-3}	$n^{1/2}\epsilon^{-2}$ $^{\Theta}$

Table 1: Comparable results on the gradient cost for nonconvex optimization algorithms that use only individual (or stochastic) gradients. Note that the gradient cost hides a poly-logarithmic factors of d, n, ϵ. For clarity and brevity purposes, we record for most algorithms the gradient cost for finding an $(\epsilon, \mathcal{O}(\epsilon^{0.5}))$-approximate second-order stationary point. For some algorithms we added in a bracket underneath the best gradient cost for finding an $(\epsilon, \mathcal{O}(\epsilon^{\alpha}))$-approximate second-order stationary point among $\alpha \in (0, 1]$, for the fairness of comparison.
$^{\Delta}$: we provide lower bound for this gradient cost entry.
$^{\Theta}$: this entry is for $n \geq \Omega(\epsilon^{-1})$ only, in which case SPIDER-SFO$^+$ outperforms NEON+FastCubic/CDHS.

$\mathbb{P}\left(\left(\boldsymbol{\mathcal{G}}^{\leq t}\right)^c \mid \boldsymbol{\mathcal{G}}^{<t}\right) \leq 2(n^2K)e^{\frac{-a^2(d/n-2T)}{2}}$. Thus

$$\begin{aligned}\mathbb{P}\left(\left(\boldsymbol{\mathcal{G}}^{\leq T}\right)^c\right) &\leq 2(T+1)n^2K\exp\left(\frac{-a^2(d/n-2T)}{2}\right)\\ &\overset{T=\frac{nK}{2}}{\leq} 2(nK)(n^2K)\exp\left(\frac{-a^2(d/n-2T)}{2}\right).\end{aligned} \tag{C.33}$$

Then by setting

$$\begin{aligned}d/n &\geq 2\max(9n^2K^2, 12nKR^2)\log(\frac{2n^3K^2}{p}) + nK\\ &\geq 2\max(9(T+1)^2, 2(2\sqrt{3T})^2R^2)\log(\frac{2n^3K^2}{p}) + 2T\\ &\geq 2\max(9(T+1)^2, 2(1+\sqrt{3T})^2R^2)\log(\frac{2n^3K^2}{p}) + 2T\\ &\geq \frac{2}{a^2}\log(\frac{2n^3K^2}{p}) + 2T,\end{aligned} \tag{C.34}$$

we have $\mathbb{P}\left(\left(\boldsymbol{\mathcal{G}}^{\leq T}\right)^c\right) \leq p$. This ends proof.

□

D Comparison with Concurrent Works

We detail our main result for applying SPIDER to first-order methods in the list below:

(i) For the problem of finding an ϵ-approximate first-order stationary point, under Assumption 1 our results indicate a gradient cost of $\mathcal{O}(\min(\epsilon^{-3}, n^{1/2}\epsilon^{-2}))$ which supersedes the best-known convergence rate results for stochastic optimization problem (1.2) [Theorems 1 and 2]. Before this work, the best-known result is $\mathcal{O}\left(\min(\epsilon^{-3.333}, n^{2/3}\epsilon^{-2})\right)$, achieved by Allen-Zhu & Hazan

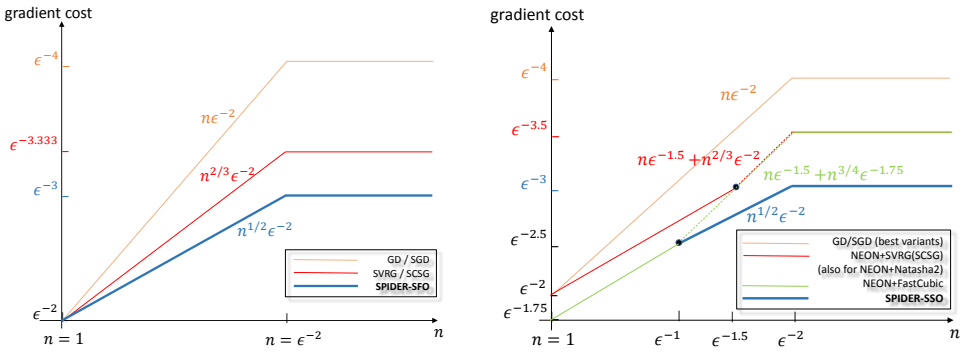

Figure 1: Left panel: gradient cost comparison for finding an ϵ-approximate first-order stationary point. Right panel: gradient cost comparison for finding an $(\epsilon, \mathcal{O}(\epsilon^{0.5}))$-approximate second-order stationary points (note we assume Hessian Lipschitz condition). Both axes are on the logarithmic scale of ϵ^{-1}.

[3], Reddi et al. [32] in the finite-sum case and Lei et al. [24] in the online case, separately. Moreover, such a gradient cost achieves the algorithmic lower bound for the finite-sum setting [Theorem 3].

(ii) For the problem of finding (ϵ, δ)-approximate second-order stationary point x, the gradient cost is $\tilde{\mathcal{O}}(\epsilon^{-3}+\epsilon^{-2}\delta^{-2}+\delta^{-5})$ in the online case and $\tilde{\mathcal{O}}(n^{1/2}\epsilon^{-2}+n^{1/2}\epsilon^{-1}\delta^{-2}+\epsilon^{-1}\delta^{-3}+\delta^{-5}+n)$. In the classical definition of second-order stationary point where $\delta = \mathcal{O}(\epsilon^{0.5})$, such gradient cost is simply $\mathcal{O}(\epsilon^{-3})$ in the online case. In comparison, to the best of our knowledge the best-known results only achieve a gradient cost of $\mathcal{O}(\epsilon^{-3.5})$ under similar assumptions [2, 4, 31, 35, 39].

We summarize the comparison with concurrent works that solve (1.2) under similar assumptions in Table 1. In addition, we provide Figure 1 which draws the gradient cost against the magnitude of n for both an approximate stationary point.[7] For simplicity, we leave out the complexities of the algorithms that has Hessian-vector product access and only record algorithms that use stochastic gradients only.[8] Specifically, the yellow-boxed complexities in Table 1 achieved by NEON+FastCubic/CDHS [4] and nonconvex AGD [21] for finding approximate second-order stationary points in the finite-sum case using momentum technique, are the only result that has *not* been outperformed by our SPIDER-SFO$^+$ algorithm in certain parameter regimes ($n \leq \mathcal{O}(\epsilon^{-1})$ in this case).

E Experiments

We use SPIDER-SFO to optimize a synthetic function and a neural network. The synthetic function is similar to [35], with the stochasticity coming from a random coordinate shift instead of the noise on the gradient oracle. In the experiment of neural networks, we train a fully connected network on MNIST dataset.

E.1 Synthetic function

We optimize the following function:

$$\mathbb{E}_{a,b}[w(x_1 - a) + 10(x_2 - b)^2],$$

where a and b are independently drawn from $\mathcal{N}(0, 0.1)$. $w(\cdot)$ is a W-shaped scalar function with a local maximum at the origin and two local minima on either side. We defer the exact form of $w(\cdot)$

[7] One of the results not included in this table is Carmon et al. [9], which finds an ϵ-approximate first-order stationary point in $\mathcal{O}(n\epsilon^{-1.75})$ gradient evaluations. However, their result relies on a more stringent Hessian-Lipschitz condition, in which case a second-order stationary point can be found in similar gradient cost [21].

[8] Due to the NEON method [4, 38], nearly all existing Hessian-vector product based algorithms in stochastic optimization can be converted to ones that use stochastic gradients only.

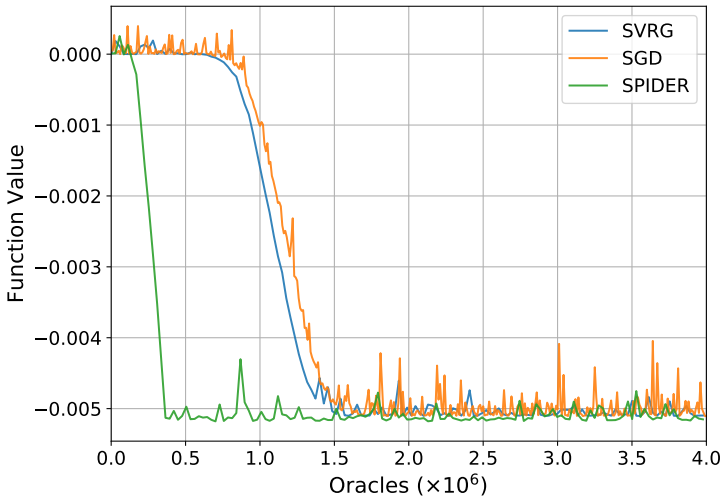

Figure 2: Results on synthetic non-convex optimization problem.

to appendix. We initialize with $x_1 = x_2 = 0$, and use SPIDER-SFO to optimize the function in an online manner. We compare SPIDER with SGD and SVRG. For each algorithm, we tune the learning rate with a grid search and plot the optimal choice. The hyper-parameters and grid search settings can be found later.

We plot the relationship between function value and number of gradient oracles called in Figure 2. It can be seen that SPIDER escapes saddle point and converges to the global minimum faster than both SGD and SVRG.

Exact form of w: The function $w(\cdot)$ in the experiments of synthetic experiment is defined as [35]

$$w(x) = \begin{cases} \sqrt{\epsilon}(x + (L+1)\sqrt{\epsilon})^2 - \frac{1}{3}(x + (L+1)\sqrt{\epsilon})^3 - \frac{1}{3}(3L+1)\epsilon^{3/2}, & x \le -L\sqrt{\epsilon}; \\ \epsilon x + \frac{\epsilon^{3/2}}{3} & -L\sqrt{\epsilon} < x \le -\sqrt{\epsilon}; \\ -\sqrt{\epsilon}x^2 - \frac{x^3}{3}, & -\sqrt{\epsilon} < x \le 0; \\ -\sqrt{\epsilon}x^2 + \frac{x^3}{3}, & 0 < x \le \sqrt{\epsilon}; \\ -\epsilon x + \frac{\epsilon^{3/2}}{3} & \sqrt{\epsilon} < x \le L\sqrt{\epsilon}; \\ \sqrt{\epsilon}(x - (L+1)\sqrt{\epsilon})^2 - \frac{1}{3}(x - (L+1)\sqrt{\epsilon})^3 - \frac{1}{3}(3L+1)\epsilon^{3/2}, & L\sqrt{\epsilon} \le x. \end{cases} \tag{E.1}$$

We set $\epsilon = 0.01$ and $L = 5$ in the experiments.

Hyper-parameters: For SGD, we use minibatch of size 100. For SPIDER-SFO, we set $s1 = 1000$, $s2 = 100$ and $q = 10$. We set the $\tilde{\epsilon}$ in the algorithm as 10^{-10}. For SVRG, we set the larger batch size with 1000 and the smaller batch size as 100, and update the large batch gradient every 10 iterations.

Learning rate: For each algorithm, we grid search the learning rate from set

$$\{0.1, 0.3, 0.01, 0.03, 0.001, 0.003, 0.0001, 0.0003\}$$

and choose the best one.

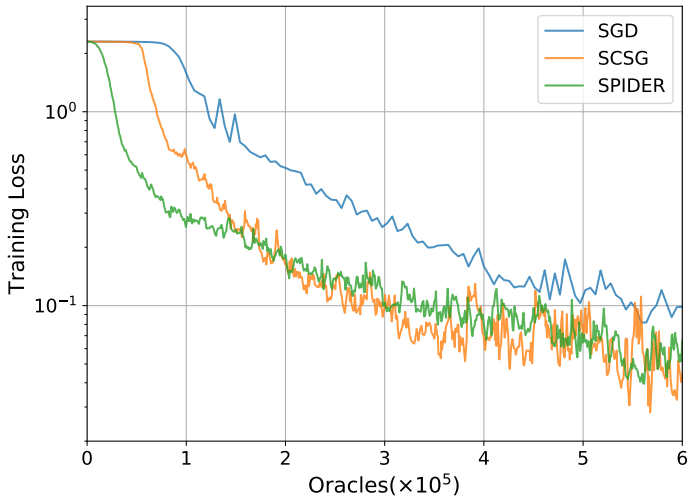

Figure 3: Results on MNIST classification.

E.2 Neural network

In addition to the synthetic function, we also train a two hidden-layer fully connected neural network for classification on the MNIST dataset. Results are presented in Figure 3. In addition to SPIDER and SGD, we also compare with SCSG [24], an online version of SVRG. For both SPIDER and SGD, we set the larger batch size as 512 and the smaller batch size as 32.

In the real experiment on MNIST, we find that SPIDER only achieves a slight improvement. We leave designing a practical version of SPIDER in the near future.

All the experiments with neural networks are implemented with PyTorch. Implementation details are as follows:

Dataset: We use MNIST dataset with 50000 training data. Each data point is a picture with 28×28 pixels with one of 10 labels. We train a network to minimize the cross-entropy loss.

Network architecture: We use a 2 hidden-layer fully connected neural network, where each hidden layer contains 512 neurons. The weights of the network are initialized randomly with normal distribution $\mathcal{N}(0, 0.01)$.

Hyper-parameters: For SGD, we use minibatch of size 512. For SPIDER-SFO, we set $s1 = 512$, $s2 = 32$ and $q = 16$. For SCSG, we set the larger batch size with 512 and the smaller batch size as 32.

Learning rate: For each algorithm, we grid search the learning rate from set

$$\{0.1, 0.3, 0.01, 0.03\}$$

and choose the best one.