Faster Stochastic Algorithms for Minimax Optimization under Polyak-Łojasiewicz Conditions

Lesi Chen  
School of Data Science  
Fudan University  
lschen19@fudan.edu.cn

Boyuan Yao  
School of Data Science  
Fudan University  
byyao19@fudan.edu.cn

Luo Luo∗  
School of Data Science  
Fudan University  
luoluo@fudan.edu.cn

Abstract

This paper considers stochastic first-order algorithms for minimax optimization under Polyak-Łojasiewicz (PL) conditions. We propose SPIDER-GDA for solving the finite-sum problem of the form

\[ \min_{x} \max_{y} f(x, y) = \frac{1}{n} \sum_{i=1}^{n} f_i(x, y), \]

where the objective function \( f(x, y) \) is \( \mu_x \)-PL in \( x \) and \( \mu_y \)-PL in \( y \); and each \( f_i(x, y) \) is \( L \)-smooth. We prove SPIDER-GDA could find an \( \epsilon \)-approximate solution within \( O \left( n + \sqrt{n} \kappa_x \kappa_y^2 \right) \log \left( \frac{1}{\epsilon} \right) \) stochastic first-order oracle (SFO) complexity, which is better than the state-of-the-art method whose SFO upper bound is \( O \left( n + n^{2/3} \kappa_x \kappa_y^2 \right) \log \left( \frac{1}{\epsilon} \right) \), where \( \kappa_x \triangleq L/\mu_x \) and \( \kappa_y \triangleq L/\mu_y \). For the ill-conditioned case, we provide an accelerated algorithm to reduce the computational cost further. It achieves \( \tilde{O} \left( n + \sqrt{n} \kappa_x \kappa_y \right) \log^2 \left( \frac{1}{\epsilon} \right) \) SFO upper bound when \( \kappa_y \gg \sqrt{n} \). Our ideas also can be applied to the more general setting that the objective function only satisfies PL condition for one variable. Numerical experiments validate the superiority of proposed methods.

1 Introduction

This paper focuses on smooth minimax optimization problem of the form

\[ \min_{x \in \mathbb{R}^d_x} \max_{y \in \mathbb{R}^d_y} f(x, y) = \frac{1}{n} \sum_{i=1}^{n} f_i(x, y), \]  

which covers a lot of important applications in machine learning such as reinforcement learning [10, 42], AUC maximization [14, 24, 48], imitation learning [5, 32], robust optimization [11], causal inference [28], game theory [6, 29] and so on.

We are interested in the minimax problems under PL conditions [9, 32, 45]. The PL condition [35] was originally proposed to relax the strong convexity in minimization problem that is sufficient for achieving the global linear convergence rate for first-order methods. In machine learning community, it has been successfully used to analyze the convergence behavior for overparameterized neural networks [23], robust phase retrieval [40] and a plenty of fundamental models [18]. There are many popular minimax formulations only satisfy PL condition, but lack strong convexity (or strong concavity). The examples include PL-game [32], robust least square [45], deep AUC maximization [24] and generative adversarial imitation learning of LQR [5, 32].

Yang et al. [45] showed that the alternating gradient descent ascent (AGDA) algorithm linearly converges to the saddle point when the objective function satisfies two-sided PL condition. They also proposed the SVRG-AGDA method for the finite-sum problem [1], which could find \( \epsilon \)-approximate
Table 1: We present the comparison of SFO complexities under two-sided PL condition. Note that Yang et al. [45] named their stochastic algorithm as variance-reduced-AGDA (VR-AGDA). Here we call it SVRG-AGDA to distinguish with other variance reduced algorithms.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Complexity</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>GDA/AGDA</td>
<td>$\mathcal{O}(n\kappa_x\kappa_y^2 \log(1/\epsilon))$</td>
<td>Theorem B.1 [45]</td>
</tr>
<tr>
<td>SVRG-AGDA</td>
<td>$\mathcal{O}((n + n^{2/3}\kappa_x\kappa_y^2) \log(1/\epsilon))$</td>
<td>[45]</td>
</tr>
<tr>
<td>SVRG-GDA</td>
<td>$\mathcal{O}((n + n^{2/3}\kappa_x\kappa_y^2) \log(1/\epsilon))$</td>
<td>Theorem C.1</td>
</tr>
<tr>
<td>SPIDER-GDA</td>
<td>$\mathcal{O}((n + \sqrt{n}\kappa_x\kappa_y^2) \log(1/\epsilon))$</td>
<td>Theorem 4.1</td>
</tr>
<tr>
<td>AccSPIDER-GDA</td>
<td>$\tilde{\mathcal{O}}\left(\sqrt{n}\kappa_x\kappa_y \log^2(1/\epsilon)\right)$, $\kappa_y \lesssim \sqrt{n}$; $\tilde{\mathcal{O}}(n\kappa_x \log^2(1/\epsilon))$, $\kappa_y \lesssim \kappa_x\kappa_y$; $\mathcal{O}\left((n + \sqrt{n}\kappa_x\kappa_y^2) \log(1/\epsilon)\right)$, $\kappa_x\kappa_y \lesssim \sqrt{n}$.</td>
<td>Theorem 5.1</td>
</tr>
</tbody>
</table>

solution within $\mathcal{O}((n + n^{2/3}\kappa_x\kappa_y^2) \log(1/\epsilon))$ stochastic first-order oracle (SFO) calls\(^2\) where $\kappa_x$ and $\kappa_y$ are the condition numbers with respect to PL condition for $x$ and $y$ respectively. The variance reduced technique in the SVRG-AGDA leads to a better convergence rate than full batch AGDA whose SFO complexity is $\mathcal{O}(n\kappa_x\kappa_y^2 \log(1/\epsilon))$. However, there are still some open questions left. Firstly, Yang et al. [45]’s theoretical analysis heavily relies on the alternating update rules. It remains interesting whether a simultaneous version of GDA (or its stochastic variants) also has similar convergence results. Secondly, it is unclear whether the SFO upper bound obtain by SVRG-AGDA can be improved by designing more efficient algorithms.

For one-sided PL condition, we desire to find the stationary point of $g(x) \triangleq \max_{y \in \mathbb{R}^d} f(x, y)$, since the saddle point may not exist. Nouiehed et al. [32] proposed the multi-step GDA method that achieves the $\epsilon$-stationary point within $\mathcal{O}(\kappa_y^2 L \epsilon^{-2} \log(\kappa_y^2 / \epsilon))$ numbers of full gradient iterations. The similar complexity also can be obtained by AGDA [45]. Recently, Yang et al. [47] proposed the smoothed-AGDA that improves the upper bound into $\mathcal{O}(\kappa_y L \epsilon^{-2})$. Both multi-step GDA Nouiehed et al. [32] and smoothed-AGDA [46] can be extended to online setting [14], but the formulation [4] with finite-sum structure has not been explored.

In this paper, we introduce a variance reduced first-order method, called SPIDER-GDA, which constructs the gradient estimator by stochastic recursive gradient and the iterations are based on simultaneous gradient descent ascent. We prove that SPIDER-GDA could achieve $\epsilon$-approximate solution of the two-sided PL problem of the form (1) within $\mathcal{O}((n + \sqrt{n}\kappa_x\kappa_y^2) \log(1/\epsilon))$ SFO calls, which has better dependency on $n$ than SVRG-AGDA [45]. We also provide an acceleration framework to improve first-order methods for solving ill-conditioned minimax problems under PL conditions. The accelerated SPIDER-GDA (AccSPIDER-GDA) could achieve $\epsilon$-approximate solution within $\mathcal{O}((n + \sqrt{n}\kappa_x\kappa_y) \log^2(1/\epsilon))$ SFO calls\(^1\) when $\kappa_y \gtrsim \sqrt{n}$, which is the best known SFO upper bound for this problem. We summarize our main results and compare them with related work in Table 1. Without loss of generality, we always suppose $\kappa_x \gtrsim \kappa_y$. Furthermore, the proposed algorithms also work for minimax problem with one-sided PL condition. We present the results for this case in Table 2.

2 Related Work

The minimax optimization problem (1) can be viewed as the following minimization problem

$$\min_{x \in \mathbb{R}^d} \left\{ g(x) \triangleq \max_{y \in \mathbb{R}^d} f(x, y) \right\}.$$  

A natural way to solve such problem is the multi-step GDA algorithm [21][25][32][36] that contains double-loop iterations in which the outer loop can be regarded as running inexact gradient descent on

\(^1\)The original analysis [45] provided an SFO upper bound $\mathcal{O}((n + n^{2/3} \max\{\kappa_x^2, \kappa_y^2\}) \log(1/\epsilon))$, which can be refined to $\mathcal{O}((n + n^{2/3} \kappa_x \kappa_y^2) \log(1/\epsilon))$ by some little modification in the proof.

\(^2\)In this paper, we use the notation $\tilde{\mathcal{O}}(\cdot)$ to hide the logarithmic factors of $\kappa_x, \kappa_y$, but not $1/\epsilon$. 

2 Related Work

The minimax optimization problem (1) can be viewed as the following minimization problem

$$\min_{x \in \mathbb{R}^d} \left\{ g(x) \triangleq \max_{y \in \mathbb{R}^d} f(x, y) \right\}.$$  

A natural way to solve such problem is the multi-step GDA algorithm [21][25][32][36] that contains double-loop iterations in which the outer loop can be regarded as running inexact gradient descent on
We are interested in the finite-sum minimax optimization problem (1) under following assumptions. We suppose the differentiable function \( f \) is nonconvex \([18]\). Note that the PL condition does not require the strongly convexity and it can be satisfied even if the function is nonconvex \([18]\). Then we formally define the Polyak-Łojasiewicz (PL) condition \([35]\) as follows.

\[
\|\nabla f(x)\| \leq \kappa_f \langle f(x), y \rangle \quad \text{for any } x, y \in \mathbb{R}^d.
\]

The Catalyst acceleration \([20]\) is a useful approach to reduce the computational cost of ill-conditioned and strongly-convex problems, i.e., there exist constants \( L > 0 \) and \( \kappa_f > 0 \) such that \( \|\nabla f_i(x, y) - \nabla f_i(x', y')\|^2 \leq L^2 (\|x - x'\|^2 + \|y - y'\|^2) \) holds for any \( x, x' \in \mathbb{R}^d \) and \( y, y' \in \mathbb{R}^d \).\]

\[
\text{Assumption 3.1.} \quad \text{We suppose each component } f_i : \mathbb{R}^{d_x} \times \mathbb{R}^{d_y} \to \mathbb{R} \text{ is } L\text{-smooth, i.e., there exists a constant } L > 0 \text{ such that } \|\nabla f_i(x, y) - \nabla f_i(x', y')\|^2 \leq L^2 (\|x - x'\|^2 + \|y - y'\|^2) \text{ holds for any } x, x' \in \mathbb{R}^{d_x} \text{ and } y, y' \in \mathbb{R}^{d_y}.
\]

\[
\text{Assumption 3.2.} \quad \text{We suppose the differentiable function } f : \mathbb{R}^{d_x} \times \mathbb{R}^{d_y} \to \mathbb{R} \text{ satisfies two-sided PL condition, i.e., there exist constants } \mu_x > 0 \text{ and } \mu_y > 0 \text{ such that } f(x, y) \text{ is } \mu_x\text{-PL for any } y \in \mathbb{R}^{d_y} \text{ and } -f(x, \cdot) \text{ is } \mu_y\text{-PL for any } x \in \mathbb{R}^{d_x}.
\]

### 3 Notation and Preliminaries

First of all, we present the definition of saddle point.

**Definition 3.1.** We say \((x^*, y^*) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}\) is a saddle point of function \( f : \mathbb{R}^{d_x} \times \mathbb{R}^{d_y} \to \mathbb{R} \) if it holds that \( f(x^*, y) \leq f(x^*, y^*) \leq f(x, y^*) \) for any \( x \in \mathbb{R}^{d_x} \) and \( y \in \mathbb{R}^{d_y} \).

Then we formally define the Polyak-Łojasiewicz (PL) condition \([35]\) as follows.

**Definition 3.2.** We say a differentiable function \( h : \mathbb{R}^d \to \mathbb{R} \) satisfies \( \mu\text{-PL} \) for some \( \mu > 0 \) if \( \|\nabla h(z)\|^2 \geq 2\mu (h(z) - \min_{z' \in \mathbb{R}^d} h(z')) \) holds for any \( z \in \mathbb{R}^d \).

Note that the PL condition does not require the strongly convexity and it can be satisfied even if the function is nonconvex \([18]\). We are interested in the finite-sum minimax optimization problem (1) under following assumptions.

**Assumption 3.1.** We suppose each component \( f_i : \mathbb{R}^{d_x} \times \mathbb{R}^{d_y} \to \mathbb{R} \) is \( L\text{-smooth, i.e., there exists a constant } L > 0 \) such that \( \|\nabla f_i(x, y) - \nabla f_i(x', y')\|^2 \leq L^2 (\|x - x'\|^2 + \|y - y'\|^2) \) holds for any \( x, x' \in \mathbb{R}^{d_x} \) and \( y, y' \in \mathbb{R}^{d_y} \).

**Assumption 3.2.** We suppose the differentiable function \( f : \mathbb{R}^{d_x} \times \mathbb{R}^{d_y} \to \mathbb{R} \) satisfies two-sided PL condition, i.e., there exist constants \( \mu_x > 0 \) and \( \mu_y > 0 \) such that \( f(\cdot, y) \) is \( \mu_x\text{-PL} \) for any \( y \in \mathbb{R}^{d_y} \) and \(-f(x, \cdot) \) is \( \mu_y\text{-PL} \) for any \( x \in \mathbb{R}^{d_x} \).

---

### Table 2: We present the comparison of SFO complexities under one-sided PL condition.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Complexity</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-Step GDA</td>
<td>( O(n\kappa_y^2 L^{-2} \log (\kappa_y / \epsilon)) )</td>
<td>([32])</td>
</tr>
<tr>
<td>GDA/AGDA</td>
<td>( O \left( n\kappa_y^2 L^{-2} \right) )</td>
<td>Theorem 3.2, 45</td>
</tr>
<tr>
<td>Smooothed-AGDA</td>
<td>( O \left( n\kappa_y^2 L^{-2} \right) )</td>
<td>([47])</td>
</tr>
<tr>
<td>SVRG-GDA</td>
<td>( O \left( n + n^{2/3}\kappa_y^2 L^{-2} \right) )</td>
<td>Theorem F.1</td>
</tr>
<tr>
<td>SPIDER-GDA</td>
<td>( O \left( n + \sqrt{n\kappa_y^2 L^{-2}} \right) )</td>
<td>Theorem 6.1</td>
</tr>
</tbody>
</table>
| AccSPIDER-GDA      | \( \begin{cases} O \left( \sqrt{n\kappa_y L^{-2} \log (\kappa_y / \epsilon)} \right), & \sqrt{n} \lesssim \kappa_y; \\
|                     | O \left( nL^{-2} \log (\kappa_y / \epsilon) \right), & \kappa_y \lesssim \sqrt{n} \lesssim \kappa_y^2; \\
|                     | O \left( n + \sqrt{n\kappa_y^2 L^{-2}} \right), & \kappa_y^2 \lesssim \sqrt{n}. \end{cases} \) | Theorem 6.2 |

---

\( g(x) \) and the inner loop finds the approximate solution to \( \max_{y \in \mathbb{R}^{d_y}} f(x, y) \) for a given \( x \). Another class of methods is the two-timescale (alternating) GDA algorithm \([9, 21, 44, 45]\) that only has single-loop iterations which update two variables with different stepsizes. The two-timescale GDA method can be implemented more easily and typically performs better than multi-step GDA empirically. Its convergence rate also can be established by analyzing function \( g(x) \) but the analysis is more challenging than the multi-step GDA.
We first consider the two-sided PL conditioned minimax problem of the finite-sum form (1) under Assumption 3.1, 3.2 and 3.3. We propose a novel stochastic algorithm, which we refer to as SPIDER-GDA. The detailed procedure of our method is presented in Algorithm 1. SPIDER-GDA constructs solutions or saddle point that is defined as follows.

Assumption 3.3 (Yang et al. [45]). We suppose the function \( f : \mathbb{R}^d_x \times \mathbb{R}^d_y \rightarrow \mathbb{R} \) has at least one saddle point \((x^*, y^*)\). We also suppose that for any fixed \( y \in \mathbb{R}^d_y \), the problem \( \min_{x \in \mathbb{R}^d_x} f(x, y) \) has a nonempty solution set and a finite optimal value; and for any fixed \( x \in \mathbb{R}^d_x \), the problem \( \max_{y \in \mathbb{R}^d_y} f(x, y) \) has a nonempty solution set and a finite optimal value.

The goal of solving minimax problem under two-sided PL condition is finding an \( \epsilon \)-approximate solution or \( \epsilon \)-saddle point that is defined as follows.

Definition 3.3. We say \( x \) is an \( \epsilon \)-approximate solution of problem (1) if it holds that \( g(x) - g(x^*) \leq \epsilon \), where \( g(x) = \max_{y \in \mathbb{R}^d_y} f(x, y) \).

Definition 3.4. Under Assumption 3.3 we say \((x, y)\) is an \( \epsilon \)-saddle point of problem (1) if it holds that \( \|x - x^*\|^2 + \|y - y^*\|^2 \leq \epsilon \) for some saddle point \((x^*, y^*)\).

We allow the saddle point does not exist for the problem with one-sided PL condition. In such case, it is guaranteed that \( g(x) \triangleq \max_{y \in \mathbb{R}^d_y} f(x, y) \) is differentiable [32, Lemma A.5] and we target to find an \( \epsilon \)-stationary point of \( g(x) \).

Definition 3.5. If the function \( g : \mathbb{R}^d_x \rightarrow \mathbb{R} \) is differentiable, we say \( x \) is an \( \epsilon \)-stationary point of \( g \) if it holds that \( \|\nabla g(x)\| \leq \epsilon \).

4 A Faster Algorithm for the Two-Sided PL Condition

We first consider the two-sided PL conditioned minimax problem of the finite-sum form (1) under Assumption 3.1, 3.2 and 3.3. We propose a novel stochastic algorithm, which we refer to as SPIDER-GDA. The detailed procedure of our method is presented in Algorithm 1. SPIDER-GDA constructs stochastic recursive gradient estimators [12, 31] as follows:

\[
G_x(x_{t,k}, y_{t,k}) = \frac{1}{B} \sum_{i \in S_x} \left( \nabla_x f_i(x_{t,k}, y_{t,k}) - \nabla_x f_i(x_{t,k-1}, y_{t,k-1}) + G_x(x_{t,k-1}, y_{t,k-1}) \right),
\]

\[
G_y(x_{t,k}, y_{t,k}) = \frac{1}{B} \sum_{i \in S_y} \left( \nabla_y f_i(x_{t,k}, y_{t,k}) - \nabla_y f_i(x_{t,k-1}, y_{t,k-1}) + G_y(x_{t,k-1}, y_{t,k-1}) \right).
\]

It simultaneously updates two variables \( x \) and \( y \) by estimators \( G_x \) and \( G_y \) with different stepsizes \( \tau_x = \Theta(1/(\kappa_x^2 L)) \) and \( \tau_y = \Theta(1/L) \) respectively. Huang et al. [16], Luo et al. [25], Xian et al. [44] have studied the SPIDER-type algorithm for nonconvex-strongly-concave problem and showed it converges to the stationary point of \( g(x) \triangleq \max_{y \in \mathbb{R}^d_y} f(x, y) \) sublinearly. However, solving the problem minimax problems with two-sided PL condition desires stronger linear convergence rate, which leads to our theoretical analysis be different from previous work.

We measure the convergence of SPIDER-GDA by the following Lyapunov function

\[
\mathcal{V}_{t,k} \triangleq g(x_{t,k}) - g(x^*) + \frac{\lambda\tau_x}{\tau_y}(g(x_{t,k}) - f(x_{t,k}, y_{t,k}))\text{,}
\]

where \( x^* \) arg \( \min_{x \in \mathbb{R}^d_x} g(x) \) and \( \lambda = \Theta(\kappa_y^2) \). We can establish recursion for \( \mathcal{V}_{t,k} \) as follows

\[
E[\mathcal{V}_{t,k}|\mathcal{F}_{t-1}] \leq E[\mathcal{V}_{t,0}|\mathcal{F}_{t-1}] - \frac{\tau_x}{16} \left( 2 - \frac{M}{B} \right) \sum_{k=0}^{K-1} \|G_x(x_{t,k}, y_{t,k})\|^2 - \frac{\lambda\tau_x}{16} \left( 2 - \frac{M}{B} \right) \sum_{k=0}^{K-1} \|G_y(x_{t,k}, y_{t,k})\|^2.
\]

Using the above inequality by setting \( M = B = \sqrt{n} \) leads to the estimators \( G_x(x_{t}, y_{t}) \) and \( G_y(x_{t}, y_{t}) \) are sufficiently close to the exact gradient and converge to zero linearly, which indicates \( g(x_{t}) \) also converges to \( g(x^*) \) linearly. We formally provide the convergence result for SPIDER-GDA in the following theorem and its detailed proof is shown in appendix.

Theorem 4.1. Under Assumption 3.1, 3.2 and 3.3 we run Algorithm 1 with \( M = B = \sqrt{n} \), \( \tau_x = 1/(5\kappa_x^2 \mu_x^2) \), \( \tau_y = g_y/\left(24\kappa_x \mu_y\right) \), \( \kappa_x = 4224/(\mu_x \tau_x) \), \( T = \lceil \log(1/\epsilon) \rceil \), and \( \epsilon = 24\epsilon \). Then the output \((\hat{x}_T, \hat{y}_T)\) satisfies \( g(\hat{x}_T) - g(x^*) \leq \epsilon \) and \( g(\hat{x}_T) - f(\hat{x}_T, \hat{y}_T) \leq 24\epsilon \) in expectation; and it takes no more than \( O(n + \sqrt{n}\kappa_x \kappa_y^2 \log(1/\epsilon)) \) SFO calls.
Algorithm 1 SPIDER-GDA $(f, (x_0, y_0), T, K, M, B, \tau_x, \tau_y)$

1: $\tilde{x}_0 = x_0, \tilde{y}_0 = y_0$
2: for $t = 0, 1, \ldots, T - 1$ do
3: \hspace{1em} $x_{t,0} = \tilde{x}_t, y_{t,0} = \tilde{y}_t$
4: \hspace{1em} for $k = 0, 1, \ldots, K - 1$ do
5: \hspace{2em} if $\text{mod}(k, M) = 0$ then
6: \hspace{3em} $G_x(x_{t,k}, y_{t,k}) = \nabla_x f(x_{t,k}, y_{t,k})$
7: \hspace{3em} $G_y(x_{t,k}, y_{t,k}) = \nabla_y f(x_{t,k}, y_{t,k})$
8: \hspace{2em} else
9: \hspace{3em} draw mini-batches $S_x$ and $S_y$ independently with both sizes of $B$.
10: \hspace{3em} $G_x(x_{t,k}, y_{t,k}) = \frac{1}{B} \sum_{i \in S_x} [\nabla_x f_i(x_{t,k}, y_{t,k}) - \nabla_x f_i(x_{t,k-1}, y_{t,k-1}) + G_x(x_{t,k-1}, y_{t,k-1})]$ \hspace{0.5em} 
11: \hspace{3em} $G_y(x_{t,k}, y_{t,k}) = \frac{1}{B} \sum_{i \in S_y} [\nabla_y f_i(x_{t,k}, y_{t,k}) - \nabla_y f_i(x_{t,k-1}, y_{t,k-1}) + G_y(x_{t,k-1}, y_{t,k-1})]$ \hspace{0.5em} 
12: \hspace{2em} end if
13: \hspace{1em} $x_{t,k+1} = x_{t,k} - \tau_x G_x(x_{t,k}, y_{t,k})$
14: \hspace{1em} $y_{t,k+1} = x_{t,k} + \tau_y G_y(x_{t,k}, y_{t,k})$
15: \hspace{1em} end for
16: \hspace{1em} choose $(\tilde{x}_{t+1}, \tilde{y}_{t+1})$ from $\{(x_{t,k}, y_{t,k})\}_{k=0}^{K-1}$ uniformly at random.
17: \hspace{1em} end for
18: return $(\tilde{x}_T, \tilde{y}_T)$

Algorithm 2 AccSPIDER-GDA

1: $u_0 = x_0$
2: for $k = 0, 1, \ldots, K - 1$ do
3: \hspace{1em} $(x_{k+1}, y_{k+1}) = \text{SPIDER-GDA}(f(x, y) + \frac{\alpha}{2}\|x - u_k\|^2, (x_k, y_k), T_k, K, M, B, \tau_x, \tau_y)$
4: \hspace{1em} $u_{k+1} = x_{k+1} + \gamma(x_{k+1} - x_k)$
5: \hspace{1em} end for
6: option I (two-sided PL): return $(x_K, y_K)$
7: option II (one-sided PL): return $(\tilde{x}, \tilde{y})$ chosen uniformly at random from $\{(x_k, y_k)\}_{k=0}^{K-1}$

Our results provide an SFO upper bound of $O(\sqrt{n} + \sqrt{n} \|x_0\|^2 \log(1/\epsilon))$ for finding an $\epsilon$-approximate solution that is better than the complexity $O((n + n^{2/3} \|x_0\|^2) \log(1/\epsilon))$ derived from SVRG-AGDA \[45\]. It is possible to use SVRG-type \[17\][49] estimators to replace the stochastic recursive estimators in Algorithm \[1\] which results the algorithm SVRG-GDA. We can prove that SVRG-GDA also has $O((n + n^{2/3} \|x_0\|^2) \log(1/\epsilon))$ SFO upper bound that matches the theoretical result of SVRG-AGDA. We provide the details in Appendix \[C\].

5 Further Acceleration with Catalyst

Both the proposed SPIDER-GDA (Algorithm \[1\]) and existing SVRG-AGDA \[45\] have the complexities more heavily depend on the condition number of $y$ than the condition number of $x$. It is natural to ask can we make the dependence of two condition numbers balanced like the results in the strongly-convex-strongly-concave case \[22\][45][41]. In this section, we show it is possible by introducing the Catalyst acceleration.

To make acceleration possible, we need to assume the uniqueness of the optimal set for inner problem.

Assumption 5.1. We assume the inner problem $\max_{y \in \mathbb{R}^d} f(x, y)$ has an unique solution.
We proposed the accelerated SPIDER-GDA (AccSPIDER-GDA) in Algorithm 2 for reducing the computational cost further. Each iteration of the algorithm solve the following sub-problem

\[
\min_{x \in \mathbb{R}^d} \max_{y \in \mathbb{R}^d} F_k(x, y) \triangleq \min_{x \in \mathbb{R}^d} \left\{ g(x) + \frac{\beta}{2} \|x - u_k\|^2 \right\}.
\]

by SPIDER-GDA (Algorithm 1). AccSPIDER-GDA has the following convergence result if the sub-problem attain the required accuracy.

**Lemma 5.1.** Under Assumption 3.1, 3.2 and 3.3, if we use Algorithm 2 with \( \beta = 2L, \gamma = 0 \) and the appropriate setting for the sub-problem solver such that \( \mathbb{E}[\|x_k - \bar{x}_k\|^2 + \|y_k - y_k\|^2] \leq \delta \), where \((\bar{x}_k, \bar{y}_k)\) is a saddle point of \( F_{k-1} \) \((k \geq 1)\) and we set the precision

\[
\delta = \frac{\mu_x \epsilon}{11(\mu_x + 4L) L}
\]

Then it holds that

\[
\mathbb{E}[g(x_k) - g(x^*)] \leq \left( 1 - \frac{\mu_x}{2 \beta + \mu_x} \right) \left( g(x_0) - g(x^*) \right) + \frac{\epsilon}{2}.
\]

The setting \( \beta = \Theta(L) \) in Lemma 5.1 guarantees the sub-problem \( (2) \) has condition number of the order \( O(1) \) for \( x \). It is more well-conditioned on \( x \), we prefer to address the following equivalent problem

\[
\max_{y \in \mathbb{R}^d} \min_{x \in \mathbb{R}^d} F_k(x, y) = -\min_{y \in \mathbb{R}^d} \max_{x \in \mathbb{R}^d} \left\{ -F_k(x, y) \right\}.
\]

Since \( (1) \) is a minimax problem satisfying two-sided PL condition, we can apply SPIDER-GDA to solve it. And we can show that under Assumption 5.1, the saddle point \((\bar{x}_k, \bar{y}_k)\) of each \( F_{k-1} \) \((k \geq 1)\) is unique (see Lemma 2 in appendix) and we are able to obtain a good approximation to it.

**Lemma 5.2.** Under Assumption 3.1, 3.2 and 3.3, if we use Algorithm 2 to solve each sub-problem \[
\max_{y \in \mathbb{R}^d} \min_{x \in \mathbb{R}^d} F_k(x, y)
\]
with \( \beta = 2L, M = B = \sqrt{n}, \tau_x = 1/(15L), \lambda = 288, \tau_y = \tau_x/(24\lambda), K = [4224/((\mu_y \tau_y)), T_k = [\log(1/\delta_k)], \) then it holds that

\[
\mathbb{E}[\|x_{k+1} - \bar{x}_{k+1}\|^2 + \|y_{k+1} - \bar{y}_{k+1}\|^2] \leq 7236\kappa_y^2 \delta_k \mathbb{E}[\|x_k - \bar{x}_k\|^2 + \|y_k - \bar{y}_k\|^2],
\]

where \((\bar{x}_k, \bar{y}_k)\) is the unique saddle point of \( F_{k-1} \) \((k \geq 1)\).

For a short summary, Lemma 5.1 means Algorithm 2 requires \( O(\kappa_x \log(1/\epsilon)) \) numbers of inexact proximal point iterations to find an \( \epsilon \)-approximate solution of the problem. And Lemma 5.2 tells us that each sub-problem can be solved within a SFO complexity of \( O(n + \sqrt{n} \kappa_y \log(1/\delta_k)) \). Thus, the total complexity for AccSPIDER-GDA becomes \( O((n \kappa_x + \sqrt{n} \kappa_y) \log(1/\epsilon) \log(1/\delta_k)) \). Our next step is to specify \( \delta_k \) which would lead to the total SFO complexity of the algorithm.

**Theorem 5.1.** Under Assumption 3.1, 3.2, 3.3 and 3.3, if we let \( \gamma = 0, \beta = 2L \) and use Algorithm 2 to solve each sub-problem \max_{y \in \mathbb{R}^d} \min_{x \in \mathbb{R}^d} F_k(x, y)
\]
with \( M, \tau_x, \tau_y, K \) defined as Lemma 5.2 and \( T_k = [\log(1/\delta_k)] \), we have

\[
\delta_k = \begin{cases} 
\frac{1}{2288 \kappa_y} \min \left\{ \frac{1}{4} \cdot \left( \frac{(\beta - L)(1 + 8\epsilon)}{\mu_y \mu_x} \right), \frac{\delta_{\mu_y}}{144 \kappa_x^2 g(x_0) - g(x^*)}, \frac{\delta_{\mu_x} }{144 \kappa_x^2 g(x_0) - g(x^*)} \right\} , & k \geq 1; \\
\frac{1}{144 \kappa_x^2 g(x_0) - g(x^*)}, & k = 0,
\end{cases}
\]

and \( \delta \) is followed by the definition in (3). Then Algorithm 2 can return \( x_k \) such that \( g(x_k) - g(x^*) \leq \epsilon \) in expectation with no more than \( O((n \kappa_x + \sqrt{n} \kappa_y) \log(1/\epsilon) \log(1/\delta_k)) \) SFO calls.

Lemma 5.1 does not rely on the choice of sub-problem solver, we can apply the acceleration framework in Algorithm 2 by replacing SPIDER-GDA with other algorithms. We summarize the SFO complexities for the acceleration of different algorithms in Table 5.

### 6 Extension to One-Sided PL Condition

In this section, we show the idea that SPIDER-GDA and its Catalyst acceleration also work for one-sided PL condition. We relax Assumption 5.2 and 5.3 to the following one.
Algorithm 1 to solve each sub-problem

We first show that the SFO complexity of SPIDER-GDA outperforms SVRG-GDA

we obtain the following results.

**Assumption 6.1.** We suppose that

Theorem 6.2. Under Assumption 3.1, 6.1 and 5.1, if we run Algorithm 2 by

Theorem 6.1.

The AccSPIDER-GDA also performs better than SPIDER-GDA in one-sided PL condition for ill

β

Let

SFO calls.

PL condition is shown in Table 2. Besides, the algorithms of GDA and SVRG-GDA also can be

GDA reduces to SPIDER-GDA. The summary and comparison for the complexities for the one-sided

conditioned case. In the following lemma, we show that AccSPIDER-GDA could find an approximate

for some saddle point

(\hat{x}, \hat{y}) = \inf_{x \in \mathbb{R}^d} f(x, y) \hat{=} \max_{y \in \mathbb{R}^d} f(x, y) is lower bounded, i.e., we have

We first show that the SFO complexity of SPIDER-GDA outperforms SVRG-GDA by a factor of

\( O(n/\delta^2) \) in Theorem 6.1.

Theorem 6.1. Under Assumption 3.1 and 6.1, Let \( T = 1 \) and \( M, B, \tau_x, \tau_y, \lambda \) as defined in Theorem 4.1 and \( K = [64/(\tau_x \epsilon^2)] \), then Algorithm 2 can guarantee the output \( \hat{x} \) to satisfy \( \|\nabla g(\hat{x})\| \leq \epsilon \)

in expectation with no more than \( O(n + \sqrt{n \kappa_y} L \epsilon^{-2}) \) SFO calls.

The AccSPIDER-GDA also performs better than SPIDER-GDA in one-sided PL condition for ill

conditioned case. In the following lemma, we show that AccSPIDER-GDA could find an approximate

stationary point if we solve the sub-problem sufficiently accurate.

Lemma 6.1. Under Assumption 3.1 and 6.1 if it holds true that

\[ \mathbb{E}[\|x_k - \hat{x}_k\|^2 + \|y_k - \hat{y}_k\|^2] \leq \delta \]

for some saddle point \((\hat{x}_k, \hat{y}_k)\) of \( F_k - 1 (k \geq 1) \), where

\[ \delta = \frac{\epsilon^2}{8L \kappa_y (22\mu_y + 1)}. \]  

(6)

Let \( \beta = 2L \), then for the output \((\hat{x}, \hat{y})\) of Algorithm 2 it holds true that

\[ \mathbb{E}[\|\nabla g(\hat{x})\|^2 \leq \frac{8\beta (g(x_0) - g^*)}{K} + \frac{\epsilon^2}{2}. \]

Compared with two-sided PL condition, the analysis of AccSPIDER-GDA is more complicated since

the precision \( \delta_k \) at each round are different. By choosing the parameters of the algorithm carefully, we obtain the following results.

Theorem 6.2. Under Assumption 3.1, 6.1 and 5.1 if we run Algorithm 2 by \( \gamma = 0, \beta = 2L \) and use Algorithm 3 to solve each sub-problem \( \max_{y \in \mathbb{R}^d} \min_{x \in \mathbb{R}^d} F_k(x, y) \) with \( M, B, \tau_x, \tau_y, \lambda, K \) and \( T_k \) (dependent on \( \delta \)) as in Theorem 5.1 and \( \delta \) is followed by the definition in Lemma 6.1, then Algorithm 2 can find \( \hat{x} \) such that \( \|\nabla g(\hat{x})\| \leq \epsilon \) in expectation within \( O((n + \sqrt{n \kappa_y}) L \epsilon^{-2} \log(\kappa_y/\epsilon)) \) SFO calls.

We can directly set \( \beta = 0 \) for Algorithm 2 in the case of very large \( n \) and in this case AccSPIDER-GDA reduces to SPIDER-GDA. The summary and comparison for the complexities for the one-sided PL condition is shown in Table 2. Besides, the algorithms of GDA and SVRG-GDA also can be accelerated with Catalyst framework and we present the corresponding results in Table 4.

---

4The complexity for finding an \( \epsilon \)-stationary point of SVRG-GDA is presented in Appendix F.
Table 4: Acceleration for different methods under one-sided PL condition.

<table>
<thead>
<tr>
<th>Method</th>
<th>Before Acceleration</th>
<th>After Acceleration</th>
</tr>
</thead>
<tbody>
<tr>
<td>GDA</td>
<td>$O\left(\frac{\kappa_y^2 L e^{-2}}{n}\right)$</td>
<td>$O\left(\frac{\kappa_y^2 L e^{-2} \log^2(\kappa_y/\epsilon)}{n}\right)$</td>
</tr>
<tr>
<td>SVRG-GDA</td>
<td>$O\left(n + \frac{2}{3} \kappa_y^2 L e^{-2}\right)$</td>
<td>$\begin{cases} O\left(\frac{n^{2/3} \kappa_y L e^{-2} \log(\kappa_y/\epsilon)}{n}\right), &amp; n^{1/3} \lesssim \kappa_y; \ O\left(\frac{n L e^{-2} \log(\kappa_y/\epsilon)}{n}\right), &amp; \kappa_y \lesssim n^{1/3} \lesssim \kappa_y^2; \ \text{no acceleration}, &amp; \kappa_y^2 \lesssim n^{1/3}. \end{cases}$</td>
</tr>
<tr>
<td>SPIDER-GDA</td>
<td>$O\left(n + \sqrt{n} \kappa_y^2 L e^{-2}\right)$</td>
<td>$\begin{cases} O\left(\sqrt{n} \kappa_y L e^{-2} \log(\kappa_y/\epsilon)}{\sqrt{n}}\right), &amp; \sqrt{n} \lesssim \kappa_y; \ O\left(n L e^{-2} \log(\kappa_y/\epsilon)}{\sqrt{n}}\right), &amp; \kappa_y \lesssim \sqrt{n} \lesssim \kappa_y^2; \ \text{no acceleration}, &amp; \kappa_y^2 \lesssim \sqrt{n}. \end{cases}$</td>
</tr>
</tbody>
</table>

7 Experiments

In this section, we conduct the numerical experiments to show the advantage of proposed algorithms and the source code is available.

We consider the following two player Polyak-Łojasiewicz game:

$$\min_{x \in \mathbb{R}^d} \max_{y \in \mathbb{R}^d} \ f(x, y) \triangleq \frac{1}{2} x^\top P x - \frac{1}{2} y^\top Q y + x^\top R y,$$

where

$$P = \frac{1}{n} \sum_{i=1}^n p_i p_i^\top, \quad Q = \frac{1}{n} \sum_{i=1}^n q_i q_i^\top \quad \text{and} \quad R = \frac{1}{n} \sum_{i=1}^n r_i r_i^\top.$$ 

We independently sample $p_i, q_i$ and $r_i$ from $\mathcal{N}(0, \Sigma_P), \mathcal{N}(0, \Sigma_Q)$ and $\mathcal{N}(0, \Sigma_R)$ respectively. We set the covariance matrix $\Sigma_P$ as the form of $UDU^\top$ such that $U \in \mathbb{R}^{d \times r}$ is column orthogonal matrix and $D \in \mathbb{R}^{r \times r}$ is diagonal with $r < d$. The diagonal elements of $D$ are distributed uniformly in the interval $[\mu, L]$ with $0 < \mu < L$. The matrix $\Sigma_Q$ is set by the similar way to $\Sigma_P$. We also let $\Sigma_R = 0.1 V V^\top$, where each element of $V \in \mathbb{R}^{d \times d}$ is sampled from $\mathcal{N}(0, 1)$ independently. Since the covariance matrices $\Sigma_P$ and $\Sigma_Q$ are rank-deficient, it is guaranteed that both $P$ and $Q$ are singular. Hence, the objective function is not strongly-convex and not strongly-concave, but it satisfies the two-sided PL-condition [13]. We set $n = 6000$, $d = 10$, $r = 5$, $L = 1$ for all experiments; and let $\mu$ be $10^{-5}$ and $10^{-3}$ for two different settings.

We compare the proposed SPIDER-GDA (Algorithm[1]) and AccSPIDER-GDA (Algorithm[2]) with the baseline algorithm SVRG-AGDA [45]. We let $B = 1$ and $M = n$ for all of these algorithms and both of the stepsizes for $x$ and $y$ are tuned from $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. For AccSPIDER, we set $\beta = L/(20n)$ and $\gamma = 0.999$. We present the results of the number of SFO calls against the norm of gradient and the distance to the saddle point in Figure[1] and Figure[2]. It is clear that our algorithms outperform than baselines.

8 Conclusion and Future Work

In this paper, we have investigated stochastic optimization for PL conditioned minimax problem with the finite-sum objective. We have proposed the SPIDER-GDA algorithm, which reduces the dependency of the sample numbers in SFO complexity. Moreover, we have introduced a Catalyst scheme to accelerate our algorithm for solving the ill-conditioned problems. We improve the SFO upper bound of the state-of-the-art algorithms for both two-sided and one-sided PL conditions. However, the optimality of SFO algorithms for the PL conditioned minimax problem is still unclear. It is interesting to construct the lower bound for verifying the tightness of our results. It is also possible to extend our algorithm to online setting.

[https://github.com/TrueNobility303/SPIDER-GDA]
Acknowledgements

This work is supported by National Natural Science Foundation of China (No. 62206058) and Shanghai Sailing Program (22YF1402900).

References


1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 8.
   (c) Did you discuss any potential negative societal impacts of your work? [No] The purpose of this work is for to provide a better understanding of GDA on a class of nonconvex-nonconcave minimax optimization.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix for details.

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We include the codes in the supplemental materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 7.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We wants to compare the training dynamic, and different trials may cause different numerical results, which can not be observed clearly in one graph.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] The experiments is certainly simple and easy to run under CPUs.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? [No] These datasets are common.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] These datasets are common.

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]