Finding Second-Order Stationary Point for Nonconvex-Strongly-Concave Minimax Problem

Anonymous Author(s)
Affiliation
Address
email

Abstract

We study the smooth minimax optimization problem $\min_{\mathbf{x}} \max_{\mathbf{y}} f(\mathbf{x}, \mathbf{y})$, where f is ℓ-smooth, strongly-concave in $\mathbf{y}$ but possibly nonconvex in $\mathbf{x}$. Most of existing works focus on finding the first-order stationary point of the function $f(\mathbf{x}, \mathbf{y})$ or its primal function $P(\mathbf{x}) \triangleq \max_{\mathbf{y}} f(\mathbf{x}, \mathbf{y})$, but few of them focus on achieving the second-order stationary point. In this paper, we propose a novel approach for minimax optimization, called Minimax Cubic Newton (MCN), which could find an $\left(\varepsilon, \kappa^{1.5}\sqrt{\rho\varepsilon}\right)$-second-order stationary point of $P(\mathbf{x})$ with calling $\mathcal{O}(\kappa^{1.5}\sqrt{\rho}\varepsilon^{-1.5})$ times of second-order oracles and $\tilde{\mathcal{O}}\left(\kappa^2\sqrt{\rho}\varepsilon^{-1.5}\right)$ times of first-order oracles, where κ is the condition number and ρ is the Lipschitz continuous constant for the Hessian of $f(\mathbf{x}, \mathbf{y})$. In addition, we propose an inexact variant of MCN for high-dimensional problems to avoid calling the expensive second-order oracles. Instead, our method solves the cubic sub-problem inexactly via gradient descent and matrix Chebyshev expansion. This strategy still obtains the desired approximate second-order stationary point with high probability but only requires $\tilde{\mathcal{O}}\left(\kappa^{1.5}\ell\varepsilon^{-2}\right)$ Hessian-vector oracle calls and $\tilde{\mathcal{O}}\left(\kappa^2\sqrt{\rho}\varepsilon^{-1.5}\right)$ first-order oracle calls. To the best of our knowledge, this is the first work that considers the non-asymptotic convergence behavior of finding second-order stationary points for minimax problems without the convex-concave assumptions.

1 Introduction

We consider minimax optimization of the form

$$\min_{\mathbf{x}\in\mathbb{R}^{d_x}} \max_{\mathbf{y}\in\mathbb{R}^{d_y}} f(\mathbf{x}, \mathbf{y}), \tag{1}$$

where $f(\mathbf{x}, \mathbf{y})$ is ℓ-smooth, μ-strongly-concave in $\mathbf{y}$, but possibly nonconvex in $\mathbf{x}$. Problem (1) also can be written as

$$\min_{\mathbf{x}\in\mathbb{R}^{d_x}} \left\{ P(\mathbf{x}) \triangleq \max_{\mathbf{y}\in\mathbb{R}^{d_y}} f(\mathbf{x}, \mathbf{y}) \right\}. \tag{2}$$

This framework covers a wide range of applications in machine learning such as regularized GAN [30], reinforcement learning [29], domain adaptation [9] and adversarial training [33].

Most recent works focus on finding an ε-first-order stationary point (FSP) of $P(\mathbf{x})$. Lin et al. [20] showed that the vanilla gradient descent ascent (GDA) method could obtain an ε-FSP with $\mathcal{O}((\kappa^2\ell + \kappa\ell^2)\varepsilon^{-2})$ first-order oracle calls. This complexity can be reduced to $\tilde{\mathcal{O}}\left(\sqrt{\kappa}\ell\varepsilon^{-2}\right)$ by proximal iteration algorithms [21], which matches the gradient oracle lower bound for finding ε-FSP of $P(\mathbf{x})$ [13, 42]. The theory of first-order optimization for problem (1) has also been studied

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

in stochastic settings [12, 15, 20, 22, 39, 40]. However, the approximate FSP obtained by these algorithms cannot guarantee the local optimality since the primal function $P(\mathbf{x})$ could be nonconvex.

In this paper, we focus on finding a second-order stationary point (SSP) of $P(\mathbf{x})$ to capture the local optimal properties [8, 24]. Inspired by the success of second-order optimization in nonconvex minimization [1, 5, 6, 14, 18, 26, 36, 43], we propose a novel method, called Minimax Cubic Newton (MCN), which runs cubic Newton update on $\mathbf{x}$ and maximizes the objective on $\mathbf{y}$ alternatively. This iteration scheme avoids getting stuck at an unexpected FSP. Specifically, we show MCN will converge to an $\left(\varepsilon, \kappa^{1.5}\sqrt{\rho\varepsilon}\right)$-SSP of $P(\mathbf{x})$ with $\mathcal{O}\left(\kappa^{1.5}\sqrt{\rho}\varepsilon^{-1.5}\right)$ number of iterations, where κ is the condition number and ρ is the Lipschitz continuous constant of $\nabla^2 f(\mathbf{x}, \mathbf{y})$. For high-dimensional problems, we also propose an efficient algorithm, called Inexact Minimax Cubic Newton (IMCN), which avoids the expensive second-order oracle calls. IMCN approximates the second-order information by matrix Chebyshev polynomial and solves the cubic regularized sub-problem inexactly. It only requires $\tilde{\mathcal{O}}\left(\kappa^{1.5}\ell\varepsilon^{-2}\right)$ Hessian-vector oracle calls and $\tilde{\mathcal{O}}\left(\kappa^{2}\sqrt{\rho}\varepsilon^{-1.5}\right)$ first-order oracle calls to find an $\left(\varepsilon, \kappa^{1.5}\sqrt{\rho\varepsilon}\right)$-SSP. Under mild strict saddle condition [3, 10, 11, 34, 35], the approximate SSP of $P(\mathbf{x})$ implies an approximate local minimax point of $f(\mathbf{x}, \mathbf{y})$ defined by Jin et al. [17], which successfully characterizes the local optimality for problem (1). To the best of our knowledge, this is the first work that considers non-asymptotic convergence behavior of finding SSP for minimax problems without convex-concave assumptions. We also conduct experiments on both synthetic function and the real application to validate our theoretical analysis. The empirical results show that the proposed algorithms significantly outperform the GDA method.

In a concurrent work, Chen et al. [7] also studied Problem (1) and proposed Cubic-GDA which is similar to our MCN algorithm. MCN has advantage on complexity of first-order oracles by a factor of $\sqrt{\kappa}$ since Cubic-GDA adopts GD to update $\mathbf{y}$ while MCN uses AGD instead. Chen et al. [7] mentioned that the cubic sub-problem can be efficiently solved by gradient-based algorithms, but they did not provide theoretical analysis for this inexact variant, which is more practical in high dimensional case. As a comparison, we provide the complexity of both Hessian-vector oracles and first-order oracles of our inexact algorithm IMCN.

2 Preliminaries

This section first presents the notations and assumptions for our settings. Then we introduce the background of local optimality for minimax optimization and some basic algorithms.

2.1 Notations and Assumptions

For a twice differentiable function $f(\mathbf{x}, \mathbf{y})$, its partial gradients with respect to $\mathbf{x}$ and $\mathbf{y}$ are denoted as $\nabla_x f(\mathbf{x}, \mathbf{y})$ and $\nabla_y f(\mathbf{x}, \mathbf{y})$ respectively. Its Hessian matrix at $(\mathbf{x}, \mathbf{y})$ can be partitioned as $\nabla^2 f(\mathbf{x}, \mathbf{y}) = [\nabla^2_{xx} f(\mathbf{x}, \mathbf{y}), f(\mathbf{x}, \mathbf{y}); \nabla^2_{yx} f(\mathbf{x}, \mathbf{y}), \nabla^2_{yy} f(\mathbf{x}, \mathbf{y})]$, where $\nabla^2_{xx} f(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{d_x \times d_x}$, $\nabla^2_{xy} f(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{d_x \times d_y}$, $\nabla^2_{yx} f(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{d_y \times d_x}$ and $\nabla^2_{yy} f(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{d_y \times d_y}$. We also define $\mathbf{H}(\mathbf{x}, \mathbf{y}) = \nabla^2_{xx} f(\mathbf{x}, \mathbf{y}) - \nabla^2_{xy} f(\mathbf{x}, \mathbf{y})(\nabla^2_{yy} f(\mathbf{x}, \mathbf{y}))^{-1} \nabla^2_{yx} f(\mathbf{x}, \mathbf{y})$ if $\nabla^2_{yy} f(\mathbf{x}, \mathbf{y})$ is invertible.

Given a symmetric matrix $\mathbf{A}$, we denote $\lambda_{\min}(\mathbf{A})$ as the smallest eigenvalue of $\mathbf{A}$. We use $\|\cdot\|_2$ to denote the spectral norm of matrices and Euclidean norm of vectors. We also denote the closed Euclidean ball with radius r and center $\mathbf{x}^*$ as $\mathcal{B}(\mathbf{x}^*, r) = \{\mathbf{x} : \|\mathbf{x} - \mathbf{x}^*\|_2 \le r\}$. Additionally, we use notation $\tilde{\mathcal{O}}(\cdot)$ to hide logarithmic terms in the complexity.

We suppose the objective function $f(\mathbf{x}, \mathbf{y})$ of Problem (1) satisfies the following assumptions.

Assumption 1. *The function $f(\mathbf{x}, \mathbf{y})$ has ℓ-Lipschitz gradients, i.e., there exists a constant $\ell > 0$ such that $\|\nabla f(\mathbf{x}, \mathbf{y}) - \nabla f(\mathbf{x}', \mathbf{y}')\|_2^2 \le \ell^2\left(\|\mathbf{x} - \mathbf{x}'\|_2^2 + \|\mathbf{y} - \mathbf{y}'\|_2^2\right)$ for any $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^{d_x}$ and $\mathbf{y}, \mathbf{y}' \in \mathbb{R}^{d_y}$.*

Assumption 2. *The function $f(\mathbf{x}, \mathbf{y})$ has ρ-Lipschitz Hessian, i.e., there exists a constant $\rho > 0$ such that $\left\|\nabla^2 f(\mathbf{x}, \mathbf{y}) - \nabla^2 f(\mathbf{x}', \mathbf{y}')\right\|_2^2 \le \rho^2\left(\|\mathbf{x} - \mathbf{x}'\|_2^2 + \|\mathbf{y} - \mathbf{y}'\|_2^2\right)$ for any $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^{d_x}$ and $\mathbf{y}, \mathbf{y}' \in \mathbb{R}^{d_y}$.*

Assumption 3. *The function $f(\mathbf{x}, \mathbf{y})$ is μ-strongly-concave in $\mathbf{y}$,i.e., there exists a constant $\mu > 0$ such that $f(\mathbf{x}, \mathbf{y}) \le f(\mathbf{x}, \mathbf{y}') + \nabla_y f(\mathbf{x}, \mathbf{y})^\top(\mathbf{y} - \mathbf{y}') - \frac{\mu}{2}\|\mathbf{y} - \mathbf{y}'\|_2^2$ for any $\mathbf{x} \in \mathbb{R}^{d_x}$ and $\mathbf{y}, \mathbf{y}' \in \mathbb{R}^{d_y}$.*

Assumption 4. *The function* $P(\mathbf{x}) \triangleq \max_{\mathbf{y}\in\mathbb{R}^{d_y}} f(\mathbf{x},\mathbf{y})$ *satisfies* $P^* \triangleq \inf_{\mathbf{x}\in\mathbb{R}^{d_x}} P(\mathbf{x}) > -\infty$.

Definition 1. *Under Assumption 1 and 3, we define the condition number of* $f(\mathbf{x},\mathbf{y})$ *as* $\kappa \triangleq \ell/\mu$.

The assumptions of Lipschitz continuous gradient and strongly-concavity on f indicate that the primal function $P(\mathbf{x}) \triangleq \max_{\mathbf{y}\in\mathbb{R}^{d_y}} f(\mathbf{x},\mathbf{y})$ is well-defined and has Lipschitz continuous gradients as shown in Lemma 1.

Lemma 1 ([20, Lemma 4.3])**.** *Suppose the objective function* f *satisfies Assumptions 1 and 3, then the primal function* $P(\mathbf{x}) \triangleq \max_{\mathbf{y}\in\mathbb{R}^{d_y}} f(\mathbf{x},\mathbf{y})$ *has* $(\kappa+1)\ell$*-Lipschitz continuous gradients. Additionally, the function* $\mathbf{y}^*(\mathbf{x}) = \arg\max_{\mathbf{y}\in\mathbb{R}^{d_y}} f(\mathbf{x},\mathbf{y})$ *is well-defined and* κ*-Lipschitz. We also have* $\nabla P(\mathbf{x}) = \nabla_{\mathbf{x}} f(\mathbf{x},\mathbf{y}^*(\mathbf{x}))$.

Now we give the definitions of ε-FSP and (ε,δ)-SSP as follows.

Definition 2. *Suppose the function* $f(\mathbf{x},\mathbf{y})$ *satisfies Assumption 1 and 3, then we call* $\mathbf{x}$ *an* ε*-FSP of* $P(\mathbf{x})$ *if* $\|\nabla P(\mathbf{x})\|_2 \le \varepsilon$.

Definition 3. *Suppose the function* $f(\mathbf{x},\mathbf{y})$ *satisfies Assumption 1, 2 and 3, then we call* $\mathbf{x}$ *is an* (ε,δ)*-SSP of* $P(\mathbf{x})$ *if* $\|\nabla P(\mathbf{x})\|_2 \le \varepsilon$ *and* $\nabla^2 P(\mathbf{x}) \succeq -\delta\mathbf{I}$.

The following two lemmas provide the closed form of $\nabla^2 P(\mathbf{x})$ and its Lipschitz continuity.

Lemma 2 ([32])**.** *Suppose the function* $f(\mathbf{x},\mathbf{y})$ *satisfies Assumption 1, 2 and 3. We use the definition of* $\mathbf{y}^*(\mathbf{x})$ *in Lemma 1, then it holds that* $\nabla^2 P(\mathbf{x}) = \mathbf{H}(\mathbf{x},\mathbf{y}^*(\mathbf{x}))$.

Lemma 3. *Under assumptions of Lemma 2, we have* $\left\|\nabla^2 P(\mathbf{x}) - \nabla^2 P(\mathbf{x}')\right\|_2 \le 4\sqrt{2}\kappa^3\rho\,\|\mathbf{x}-\mathbf{x}'\|_2$ *for any* $\mathbf{x}$ *and* $\mathbf{x}'$ *in* $\mathbb{R}^{d_x}$.

2.2 Local Optimality of Minimax Optimization

The Nash equilibrium is widely used in the study of convex-concave minimax optimization [38, 41], but it is intractable in general when the objective function $f(\mathbf{x},\mathbf{y})$ is nonconvex in $\mathbf{x}$ or nonconcave in $\mathbf{y}$. For the general minimax problem, we introduce local minimax point [17], which characterizes the optimality in two-player sequential games when players can only change their strategies locally.

Definition 4 ([17])**.** *Given a differentiable function* $f(\mathbf{x},\mathbf{y}) : \mathbb{R}^{d_x}\times\mathbb{R}^{d_y} \to \mathbb{R}$ *that is strongly-concave in* $\mathbf{y}$*, a point* $(\mathbf{x}^*,\mathbf{y}^*) \in \mathbb{R}^{d_x}\times\mathbb{R}^{d_y}$ *is called a local minimax point of* f*, if there exists* $\delta_0 > 0$ *and a function* h *satisfying* $h(\delta)\to 0$ *as* $\delta\to 0$*, such that for any* $\delta\in(0,\delta_0]$, $\mathbf{x}\in\mathcal{B}(\mathbf{x}^*,h(\delta))$ *and* $\mathbf{y}\in\mathbb{R}^{d_y}$*, we have*

$$f(\mathbf{x}^*,\mathbf{y}) \le f(\mathbf{x}^*,\mathbf{y}^*) \le \max_{\mathbf{y}'\in\mathbb{R}^{d_y}} f(\mathbf{x},\mathbf{y}'). \tag{3}$$

Remark 1. *The definition of local minimax point for general nonconvex-nonconcave [17] only requires (3) holds for any* $\mathbf{y}$ *and* $\mathbf{y}'$ *in a neighbour of* $\mathbf{y}^*$*, while the constraint on* $\mathbf{y}$ *and* $\mathbf{y}'$ *is unnecessary in our setting since we assume that* $f(\mathbf{x},\mathbf{y})$ *is strongly-concave in* $\mathbf{y}$.

The local minimax point enjoys the following property.

Lemma 4 ([17, Proposition 19])**.** *Suppose* $f(\mathbf{x},\mathbf{y})$ *is twice differentiable, then any point* $(\mathbf{x}^*,\mathbf{y}^*)$ *satisfying* $\nabla f(\mathbf{x}^*,\mathbf{y}^*)=\mathbf{0}$*,* $\nabla^2_{yy} f(\mathbf{x}^*,\mathbf{y}^*)\prec\mathbf{0}$ *and* $\mathbf{H}(\mathbf{x}^*,\mathbf{y}^*)\succ 0$ *is a local minimax point of* $f(\mathbf{x},\mathbf{y})$.

Based on Lemma 4, we introduce the strict-minimax condition on f which is an extension of strict-saddle condition for nonconvex minimization [3, 10, 11, 16, 34, 35].

Definition 5. *Under Assumption 1, 2 and 3, we say* $f(\mathbf{x},\mathbf{y})$ *is* (α,β,γ)*-strict-minimax for some* $\alpha>0$*,* $\beta>0$ *and* $\gamma>0$ *if every* $(\hat{\mathbf{x}},\hat{\mathbf{y}})\in\mathbb{R}^{d_x}\times\mathbb{R}^{d_y}$ *satisfies at least one of the following three conditions: (a)* $\|\nabla f(\hat{\mathbf{x}},\hat{\mathbf{y}})\|_2 > \alpha$*; (b)* $\lambda_{\min}(\mathbf{H}(\hat{\mathbf{x}},\hat{\mathbf{y}})) < -\beta$*; (c) There exists a local minimax point* $(\mathbf{x}^*,\mathbf{y}^*)$ *such that* $\|\hat{\mathbf{x}}-\mathbf{x}^*\|_2^2 + \|\hat{\mathbf{y}}-\mathbf{y}^*\|_2^2 \le \gamma^2$.

Note that if f is (α,β,γ)-strict-minimax and there exists a point $\hat{\mathbf{x}}$ which is an (ε,δ)-SSP of $P(\mathbf{x})$ with sufficient small ε and δ, then we can find $\hat{\mathbf{y}} \approx \arg\max_{\mathbf{y}} f(\hat{\mathbf{x}},\mathbf{y})$ via running a first-order algorithm to minimize $-f(\hat{\mathbf{x}},\cdot)$ and obtain $(\hat{\mathbf{x}},\hat{\mathbf{y}})$ which is in a neighborhood of $(\mathbf{x}^*,\mathbf{y}^*)$. In other words, under the strict-minimax condition, we can reduce the task of finding approximate local minimax point of $f(\mathbf{x},\mathbf{y})$ to finding an approximate SSP of $P(\mathbf{x})$. We will provide the formal statement for this fact in Section 3.2.

2.3 Accelerated Gradient Descent

Nesterov's accelerated gradient descent (AGD) is the optimal first-order algorithm for convex optimization [25, 27], which is widely used in minimax optimization algorithms [21, 37]. We describe the details of AGD for smooth and strongly-convex functions in Algorithm 1, which has the following convergence rate.

Algorithm 1 $\mathrm{AGD}(h, \mathbf{y}_0, K, \eta, \theta)$

$\tilde{\mathbf{y}}_0 = \mathbf{y}_0$
for $k = 0, \ldots, K$ **do**
  $\mathbf{y}_{k+1} = \tilde{\mathbf{y}}_k - \eta \nabla h(\tilde{\mathbf{y}}_k)$
  $\tilde{\mathbf{y}}_{k+1} = \mathbf{y}_{t+1} + \theta(\mathbf{y}_{k+1} - \mathbf{y}_k)$
end for
Output: $\mathbf{y}_K$

Lemma 5 ([37, Lemma 2]). *Running Algorithm 1 on a ℓ_h-smooth and μ_h-strongly-convex objective function $h(\cdot)$ with parameters $\eta = 1/\ell_h$ and $\theta = \frac{\sqrt{\kappa_h}-1}{\sqrt{\kappa_h}+1}$ produces the output satisfying $\|\mathbf{y}_K - \mathbf{y}^*\|_2^2 \le (\kappa_h + 1)\big(1 - \frac{1}{\sqrt{\kappa_h}}\big)^K \|\mathbf{y}_0 - \mathbf{y}^*\|_2^2$, where $\mathbf{y}^* = \arg\min_y h(\mathbf{y})$ and $\kappa_h = \ell_h/\mu_h$.*

2.4 Cubic Regularized Newton

Cubic regularized Newton (CRN) is a classic algorithm for nonconvex minimization [5, 6, 25, 26, 36]. It solves the nonconvex minimization problem $\min_{\mathbf{x}} g(\mathbf{x})$ via the following update rules

$$\mathbf{s}_t = \arg\min_{\mathbf{s}\in\mathbb{R}^d} \nabla g(\mathbf{x}_t)^\top \mathbf{s} + \frac{1}{2}\mathbf{s}^\top \nabla^2 g(\mathbf{x}_t)\mathbf{s} + \frac{\rho_g}{6}\|\mathbf{s}\|_2^3, \qquad \mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{s}_t.$$

The CRN method could find an $(\varepsilon, \sqrt{\rho_g \varepsilon})$-SSP of $g(\mathbf{x})$ with $\mathcal{O}(\sqrt{\rho_g}\varepsilon^{-1.5})$ number of iterations, where ρ_g is the Lipschitz continuous constant of the Hessian of $g(\mathbf{x})$.

3 Minimax Cubic Newton Algorithm

In this section, we propose our minimax cubic Newton algorithm and give its convergence results.

3.1 Minimax Cubic Newton Method

We present the details of Minimax Cubic Newton (MCN) method in Algorithm 2. In each round, the MCN algorithm performs following steps:

- Run AGD as presented in Algorithm 1 to estimate $\mathbf{y}_t \approx \mathbf{y}^*(\mathbf{x}_t) = \arg\max_{\mathbf{y}} f(\mathbf{x}_t, \mathbf{y})$.
- Construct inexact first-order and second-order information of P at $\mathbf{x}_t$ as $\nabla P(\mathbf{x}_t) \approx \mathbf{g}_t = \nabla_x f(\mathbf{x}_t, \mathbf{y}_t)$ and $\nabla^2 P(\mathbf{x}_t) \approx \mathbf{H}_t = \mathbf{H}(\mathbf{x}_t, \mathbf{y}_t)$.
- Solve the following cubic regularized problem

$$\mathbf{s}_t^* = \arg\min_{\mathbf{s}\in\mathbb{R}^{d_x}} \left(\mathbf{g}_t^\top \mathbf{s} + \frac{1}{2}\mathbf{s}^\top \mathbf{H}_t \mathbf{s} + \frac{M}{6}\|\mathbf{s}\|_2^3\right). \quad (4)$$

Algorithm 2 Minimax Cubic-Newton (MCN)

1: **Input:** $\mathbf{x}_0 \in \mathbb{R}^{d_x}$, $\mathbf{y}_{-1} = \mathbf{0}$, T, $\{K_t\}_{t=0}^T$, ε.
2: **for** $t = 0, \cdots T-1$ **do**
3: $\mathbf{y}_t = \mathrm{AGD}\left(-f(\mathbf{x}_t, \cdot), \mathbf{y}_{t-1}, K_t, \frac{1}{\ell}, \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)$
4: $\mathbf{g}_t = \nabla_x f(\mathbf{x}_t, \mathbf{y}_t)$, $\mathbf{H}_t = \mathbf{H}(\mathbf{x}_t, \mathbf{y}_t)$
5: $\mathbf{s}_t^* = \arg\min_{\mathbf{s}\in\mathbb{R}^{d_x}} \left(\mathbf{g}_t^\top \mathbf{s} + \frac{1}{2}\mathbf{s}^\top \mathbf{H}_t \mathbf{s} + \frac{M}{6}\|\mathbf{s}\|_2^3\right)$
6: **if** $\|\mathbf{s}_t^*\|_2 \le \frac{1}{2}\sqrt{\varepsilon/M}$ **then** break
7: $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{s}_t^*$
8: **end for**
9: **Output:** $\hat{\mathbf{x}} = \mathbf{x}_t + \mathbf{s}_t^*$

The expressions of $\mathbf{g}_t$ and $\mathbf{H}_t$ in the algorithm are inspired from Lemma 1 and 3. The smoothness of $\nabla P(\mathbf{x})$ and $\nabla^2 P(\mathbf{x})$ allow the total complexity of AGD iteration of the algorithm has a reasonable upper bound. Our theoretical analysis will show the termination condition in Line 7 of Algorithm 2 can be attained in no more than $\mathcal{O}(\kappa^{1.5}\sqrt{\rho}\varepsilon^{-1.5})$ number of iterations. We also show that a small $\|\mathbf{s}_t^*\|_2$ will lead to a desired approximate second-order stationary point of $P(\mathbf{x})$. Hence, using AGD to optimize $f(\mathbf{x}_t + \mathbf{s}_t^*, \mathbf{y})$ with respect to $\mathbf{y}$ generates an approximate local minimax point of $f(\mathbf{x}, \mathbf{y})$.

3.2 Complexity Analysis for MCN

In this section, we let $M = 4\sqrt{2}\kappa^3\rho$ for MCN. Our analysis for MCN algorithm contains three parts:

1. We follow Tripuraneni et al. [36]'s idea to show the algorithm with sufficient accurate gradient and Hessian estimator of $P(\mathbf{x})$ requires no more than $T = \mathcal{O}\left(\kappa^{1.5}\sqrt{\rho}\varepsilon^{-1.5}\right)$ rounds of iterations to achieve an $\left(\varepsilon, \kappa^{1.5}\sqrt{\rho\varepsilon}\right)$-SSP of $P(\mathbf{x})$.
2. We prove the AGD step in line 3 requires at most $\tilde{\mathcal{O}}\left(\kappa^2\sqrt{\rho}\varepsilon^{-1.5}\right)$ gradient calls in total.
3. The last part shows we can achieve an approximate local minimax point of $f(\mathbf{x}, \mathbf{y})$ from an $\left(\varepsilon, \kappa^{1.5}\sqrt{\rho\varepsilon}\right)$-second-order stationary point of $P(\mathbf{x})$ under strict-saddle condition.

Cubic Newton Iteration on $P(\mathbf{x})$ The procedure of our Algorithm 2 can be regarded as applying cubic Newton method to minimize nonconvex function $P(\mathbf{x})$, but depends on inexact first-order and second-order information. We consider the following conditions on the inexact gradient and Hessian, which will hold if we run AGD with enough number of iterations.

Assumption 5. *Suppose the estimators $\mathbf{g}_t \in \mathbb{R}^{d_x}$ and $\mathbf{H}_t \in \mathbb{R}^{d_x \times d_x}$ satisfy conditions $\|\nabla P(\mathbf{x}_t) - \mathbf{g}_t\|_2 \le C_g\varepsilon$ and $\left\|\nabla^2 P(\mathbf{x}_t) - \mathbf{H}_t\right\|_2 \le C_H\sqrt{M\varepsilon}$ for some $C_g > 0$ and $C_H > 0$.*

The following lemma implies the analysis of MCN algorithm only needs to focus on the case of each $\|\mathbf{s}_t^*\|_2$ is large, otherwise we have already find $\mathbf{x}_{t+1}$ which a desired approximate SSP of $P(\mathbf{x})$.

Lemma 6. *Under Assumption 5 with $C_g = 1/192$ and $C_H = 1/48$, if $\mathbf{x}_{t+1}$ from Algorithm 2 is not an $\left(\varepsilon, \sqrt{M\varepsilon}\right)$-SSP of $P(\mathbf{x})$, we have $\|\mathbf{s}_t^*\|_2 \ge \frac{1}{2}\sqrt{\varepsilon/M}$.*

By Lemma 6, we can show that MCN with sufficient accurate gradient and Hessian estimators can find an $\left(\varepsilon, \sqrt{M\varepsilon}\right)$-SSP of $P(\mathbf{x})$ with $T = \mathcal{O}\left(\kappa^{1.5}\sqrt{\rho}\varepsilon^{-1.5}\right)$ number of iterations as follows.

Theorem 1. *Under Assumption 1-4, we run Algorithm 2 with $T = \left\lceil 192(P(\mathbf{x}_0) - P^*)\sqrt{M}\varepsilon^{-1.5}\right\rceil + 1$; and further suppose K_t is sufficient large so that results $\mathbf{g}_t$ and $\mathbf{H}_t$ satisfy Assumption 5 with $C_g = 1/192$ and $C_H = 1/48$. Then the output $\hat{\mathbf{x}}$ is an $\left(\varepsilon, \sqrt{M\varepsilon}\right)$-SSP of $P(\mathbf{x})$.*

Total Complexity of AGD Note that MCN (Algorithm 2) applies AGD to maximize $f(\mathbf{x}_t, \cdot)$ by using $\mathbf{y}_{t-1}$ as initialization at the t-th round. With such initialization, the following theorem provides the upper bound of total number of gradient calls required by AGD and guarantees that $\mathbf{g}_t$ and $\mathbf{H}_t$ in the MCN algorithm satisfy Assumption 5.

Theorem 2. *Under Assumption 1-4, we run Algorithm 2 with $K_0 = \left\lceil 2\sqrt{\kappa}\log\left(\frac{\sqrt{\kappa+1}}{\tilde{\varepsilon}}\|\mathbf{y}^*(\mathbf{x}_0)\|_2\right)\right\rceil$ and $K_t = \left\lceil 2\sqrt{\kappa}\log\left(\frac{\sqrt{\kappa+1}}{\tilde{\varepsilon}}\left(\tilde{\varepsilon} + \kappa\|\mathbf{s}_{t-1}\|_2\right)\right)\right\rceil$ for $t \ge 1$, where $\tilde{\varepsilon} = \min\left\{C_g\varepsilon/\ell, C_H\sqrt{M\varepsilon}/\rho\right\}$ and $\mathbf{s}_0 = \mathbf{0}$. Then all $\mathbf{g}_t$ and $\mathbf{H}_t$ satisfy the condition of Assumption 5. We also have*

$$\sum_{t=0}^{T} K_t \le T+1+\frac{2\sqrt{\kappa}T}{3}\left[3\log\left(\frac{\sqrt{\kappa+1}}{\tilde{\varepsilon}}\|\mathbf{y}^*(\mathbf{x}_0)\|_2\right)+\log\left(8(\kappa+1)^{1.5}+\frac{8\kappa^3(\kappa+1)^{1.5}}{T\tilde{\varepsilon}^3}\sum_{t=1}^{T}\|\mathbf{s}_{t-1}\|_2^3\right)\right].$$

Combining Theorem 1 and Theorem 2, we can obtain the total number of gradient oracle calls, Hessian (inverse) oracle calls and exact cubic sub-problem solver calls as follows.

Corollary 1. *Under Assumption 1-4, running Algorithm 2 with $T = \left\lceil 192(P(\mathbf{x}_0) - P^*)\sqrt{M}\varepsilon'^{-1.5}\right\rceil + 1$, $K_0 = \left\lceil 2\sqrt{\kappa}\log\left(\frac{\sqrt{\kappa+1}}{\tilde{\varepsilon}'}\|\mathbf{y}^*(\mathbf{x}_0)\|_2\right)\right\rceil$ and $K_t = \left\lceil 2\sqrt{\kappa}\log\left(\frac{\sqrt{\kappa+1}}{\tilde{\varepsilon}'}\left(\tilde{\varepsilon}' + \kappa\|\mathbf{s}_{t-1}\|_2\right)\right)\right\rceil$ for $t \ge 1$, where $\tilde{\varepsilon}' = \min\left\{C_g\varepsilon'/\ell, C_H\sqrt{M\varepsilon'}/\rho\right\}$, $\mathbf{s}_0 = \mathbf{0}$ and $\varepsilon' = 2^{-2.5}\varepsilon$, then the output $\hat{\mathbf{x}}$ is an $\left(\varepsilon, \kappa^{1.5}\sqrt{\rho\varepsilon}\right)$-SSP of $P(\mathbf{x})$ and the number of gradient oracle calls is at most $\tilde{\mathcal{O}}\left(\kappa^2\sqrt{\rho}\varepsilon^{-1.5}\right)$. The total number of Hessian (inverse) oracle calls and exact cubic sub-problem solver calls is at most $\mathcal{O}\left(\sqrt{\rho}\kappa^{1.5}\varepsilon^{-1.5}\right)$.*

Note that MCN method needs to construct the Hessian estimator $\mathbf{H}_t = \mathbf{H}(\mathbf{x}_t, \mathbf{y}_t)$ and solve the cubic regularized sub-problem (4) in each round. Constructing $\mathbf{H}_t$ requires calling the second-order oracle at $(\mathbf{x}_t, \mathbf{y}_t)$ and taking $\mathcal{O}\left(d_y^3 + d_x d_y^2 + d_x^2 d_y\right)$ flops for matrix multiplication and inversion. Solving sub-problem (4) requires $\mathcal{O}\left(d_x^3\right)$ flops for matrix factorization or inversion [5, 6]. Hence, besides the gradient calls from AGD step, MCN requires $\mathcal{O}\left(d_x^3 + d_y^3\right)$ time complexity in each round and its space complexity is $\mathcal{O}\left(d_x^2 + d_y^2\right)$.

Algorithm 3 Inexact Minimax Cubic-Newton

1: **Input:** $\mathbf{x}_0 \in \mathbb{R}^{d_x}$, $\mathbf{y}_{-1} = \mathbf{0}$, T, $\{K_t\}_{t=0}^{T}$, ε
2: $c_k = \frac{2}{\sqrt{\ell\mu}} \left(\frac{\sqrt{\mu/\ell}-1}{\sqrt{\mu/\ell}+1} \right)^k$
3: **for** $t = 0, \cdots$ **do**
4: $\quad \mathbf{y}_t = \mathrm{AGD}\big(-f(\mathbf{x}_t, \cdot), \mathbf{y}_{t-1}, K_t, \frac{1}{\ell}, \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} \big)$
5: $\quad \mathbf{g}_t = \nabla_x f(\mathbf{x}_t, \mathbf{y}_t)$
6: $\quad \mathbf{Z}_t = -\frac{2}{\ell-\mu} \left(\nabla_{yy}^2 f(\mathbf{x}_t, \mathbf{y}_t) + \frac{\ell+\mu}{2}\mathbf{I} \right)$
7: $\quad$ Compute $\mathbf{H}_t$ as equation (5)
8: $\quad (\mathbf{s}_t, \Delta_t)$=Cubic-Solver$(\mathbf{g}_t, \mathbf{H}_t, \sigma, \mathcal{K}(\varepsilon, \delta'))$
9: $\quad \mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{s}_t$
10: $\quad$ **if** $\Delta_t > -\frac{1}{128}\sqrt{\varepsilon^3/M}$ **then**
11: $\quad\quad \hat{\mathbf{s}} =$ Final-Cubic-Solver$(\mathbf{g}_t, \mathbf{H}_t, \varepsilon)$
12: $\quad\quad \mathbf{x}_{t+1} = \mathbf{x}_t + \hat{\mathbf{s}}$
13: $\quad\quad$ **break**
14: $\quad$ **end if**
15: **end for**
16: **Output:** $\hat{\mathbf{x}} = \mathbf{x}_{t+1}$

Algorithm 4 Cubic-Solver

1: **Input:** $\mathbf{g}$, $\mathbf{H}$, σ, $\mathcal{K}(\varepsilon, \delta')$
2: **if** $\|\mathbf{g}\|_2 \geq L^2/M$ **then**
3: $\quad R_C = -\frac{\mathbf{g}^\top \mathbf{H}\mathbf{g}}{M\|\mathbf{g}\|_2^2} + \sqrt{\left(\frac{\mathbf{g}^\top \mathbf{H}\mathbf{g}}{M\|\mathbf{g}\|_2^2}\right)^2 + \frac{2\|\mathbf{g}\|_2}{M}}$
4: $\quad \hat{\mathbf{s}} = -R_C \mathbf{g}/\|\mathbf{g}\|_2$
5: **else**
6: $\quad \mathbf{s}_0 = \mathbf{0}$, $\eta = 1/20L$
7: $\quad \tilde{\mathbf{g}} = \mathbf{g} + \sigma\boldsymbol{\zeta}$ for $\boldsymbol{\zeta} \sim \mathrm{Uniform}(\mathbb{S}^{d-1})$
8: $\quad$ **for** $k = 0, \cdots, \mathcal{K}(\varepsilon, \delta')$ **do**
9: $\quad\quad \mathbf{s}_{k+1} = \mathbf{s}_k - \eta\left(\tilde{\mathbf{g}} + \mathbf{H}\mathbf{s}_k + \frac{M}{2}\|\mathbf{s}_k\|_2 \mathbf{s}_k\right)$
10: $\quad$ **end for**
11: $\quad \hat{\mathbf{s}} = \mathbf{s}_{\mathcal{K}(\varepsilon,\delta')}$
12: **end if**
13: **Output:** $\hat{\mathbf{s}}$ and $\Delta = \mathbf{g}^\top\hat{\mathbf{s}} + \frac{1}{2}\hat{\mathbf{s}}^\top\mathbf{H}\hat{\mathbf{s}} + \frac{M}{6}\|\hat{\mathbf{s}}\|_2^3$

Approximate Local Minimax Point Under the strict-minimax condition, we can find an approximate local minimax point by performing an additional AGD procedure on the output of MCN.

Corollary 2. *Suppose $f(\mathbf{x}, \mathbf{y})$ is (α, β, γ)-strict-minimax and satisfies Assumption 1-4. Let $\hat{\mathbf{x}}$ be the output of running Algorithm 2 with the setting of Corollary 1 and $\varepsilon = \min\left\{\alpha/3,\ \beta^2/(8\kappa^3\rho)\right\}$. We let $\hat{\mathbf{y}} = \mathrm{AGD}\big(-f(\hat{\mathbf{x}}, \cdot), \mathbf{y}_t, \hat{K}, \frac{1}{\ell}, \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\big)$ where t corresponds to the last iteration of Algorithm 2 such that $\hat{\mathbf{x}} = \mathbf{x}_t + \mathbf{s}_t^*$ and $\hat{K} = \sqrt{\kappa}\log\left(\min\left\{\frac{\alpha}{2\ell}, \frac{\beta}{8\kappa^2\rho}\right\} / \left(\sqrt{\kappa+1}\left(\tilde{\varepsilon} + \frac{\kappa}{2^{2.25}}\sqrt{\frac{\varepsilon}{M}}\right)\right)\right)$. Then there exists an local minimax point $(\mathbf{x}^*, \mathbf{y}^*)$ of $f(\mathbf{x}, \mathbf{y})$ such that $\|\mathbf{x}^* - \hat{\mathbf{x}}\|_2^2 + \|\mathbf{y}^* - \hat{\mathbf{y}}\|_2^2 \leq \gamma^2$.*

4 Inexact Minimax Cubic Newton Algorithm

In this section, we proposed an efficient algorithm called inexact minimax cubic Newton (IMCN), which avoids any operation related to the second-order oracle and only requires $\tilde{\mathcal{O}}\left(\kappa^{1.5}\ell\varepsilon^{-2}\right)$ Hessian-vector product calls and $\tilde{\mathcal{O}}\left(\kappa^2\sqrt{\rho}\varepsilon^{-1.5}\right)$ gradient calls in total to find $\left(\varepsilon, \kappa^{1.5}\sqrt{\rho\varepsilon}\right)$-SSP of $P(\mathbf{x})$. Since the Hessian-vector products can be computed as fast as gradient [28, 31], IMCN is much more efficient than MCN in high-dimensional problems.

Algorithm 5 Final-Cubic-Solver

1: **Input:** $\mathbf{g}$, $\mathbf{H}$, ε
2: $\mathbf{s}_0 = \mathbf{0}$, $\mathbf{g}_0 = \mathbf{g}$, $\eta = 1/(20L)$
3: **for** $t = 0, \ldots$ **do**
4: $\quad$ **if** $\|\mathbf{g}_t\|_2 \leq \varepsilon/2$ **then break**
5: $\quad \mathbf{s}_{t+1} = \mathbf{s}_t - \eta\mathbf{g}_t$
6: $\quad \mathbf{g}_{t+1} = \mathbf{g} + \mathbf{H}\mathbf{s}_{t+1} + \frac{M}{2}\|\mathbf{s}_{t+1}\|_2 \mathbf{s}_{t+1}$
7: **end for**
8: **Output:** $\mathbf{s}_t$

We present the details of IMCN in Algorithm 3. Unlike MCN solves problem (4) exactly, IMCN uses gradient descent-based cubic sub-problem solver (Algorithm 4) [4] to obtain $\mathbf{s}_t$ such that $\mathbf{s}_t \approx \arg\min_{\mathbf{s}\in\mathbb{R}^{d_x}} m_t(\mathbf{s}) \triangleq \mathbf{g}_t^\top\mathbf{s} + \frac{1}{2}\mathbf{s}^\top\mathbf{H}_t\mathbf{s} + \frac{M}{6}\|\mathbf{s}\|_2^3$. If the condition in line 10 of Algorithm 3 holds, the point $\mathbf{x}_t + \mathbf{s}_t^*$ must be a desired approximate second-order stationary point. Due to $\mathbf{s}_t^*$ is unavailable, we introduce additional gradient descent steps (Algorithm 5) to approximate it by $\hat{\mathbf{s}}$ and use $\mathbf{x}_t + \mathbf{s}_t^*$ as the final output.

The design and the convergence analysis of IMCN is more challenging than existing inexact cubic Newton algorithms for minimization problems [18, 36] since the Hessian estimator in IMCN has a more complicated structure. To address this issue, we approximate the Hessian $\nabla^2 P(x_t)$ as follows

$$\mathbf{H}_t = \nabla_{xx}^2 f(\mathbf{x}_t, \mathbf{y}_t) + \nabla_{xy}^2 f(\mathbf{x}_t, \mathbf{y}_t)\mathbf{C}_t\nabla_{yx}^2 f(\mathbf{x}_t, \mathbf{y}_t), \tag{5}$$

where $\mathbf{C}_t = \frac{c_0}{4\ell}\mathbf{I} + \frac{1}{2\ell}\sum_{k=1}^{K'} c_k\mathbf{T}_k(\mathbf{Z}_t)$ and $\mathbf{Z}_t = \frac{4\ell}{\ell-\mu}\big(-\frac{1}{2\ell}\nabla_{yy}^2 f(\mathbf{x}_t, \mathbf{y}_t) - \frac{\ell+\mu}{4\ell}\mathbf{I}\big)$.

Here $\mathbf{T}_k(\cdot)$ is the matrix Chebyshev polynomials (see the exact form in Section 4.1) leading to $\mathbf{C}_t \approx -\left(\nabla^2_{yy}(\mathbf{x}_t, \mathbf{y}_t)\right)^{-1}$. Note that we never construct matrix $\mathbf{H}_t$ explicitly in implementation because all operations related to $\mathbf{H}_t$ can be reduced to compute Hessian-vector products, which avoid any second-order oracle calls or matrix factorization/inversion.

Then we provide the convergence analysis for the IMCN algorithm. Throughout this section, we let $L = 2\kappa\ell$ and $M = 4\sqrt{2}\kappa^3\rho$ be the Lipschitz continuous constants of $\nabla P(x)$ and $\nabla^2 P(x)$. We suppose $\varepsilon \leq L^2/M$, otherwise, the second-order condition $\nabla^2 P(\mathbf{x}) \succeq -\sqrt{M\varepsilon}\mathbf{I}$ always holds and we only need to use gradient methods [20, 21] to find first-order stationary point.

4.1 Approximating Hessian by Matrix Chebyshev Polynomials

We first show the error bound of matrix inverse approximation via matrix Chebyshev polynomials.

Lemma 7. *Suppose symmetric matrix $\mathbf{X} \in \mathbb{R}^{d\times d}$ satisfieses $\mu'\mathbf{I} \preceq \mathbf{X} \preceq \ell'\mathbf{I}$ with $0 < \mu' \leq \ell' < 1$, then we have $\left\|\mathbf{X}^{-1} - \left(\frac{c_0}{2}\mathbf{I} + \sum_{k=1}^{K'} c_k \mathbf{T}_k(\mathbf{Z}')\right)\right\|_2 \leq \frac{\sqrt{\ell'/\mu'}-1}{\sqrt{\ell'\mu'}}\left(1 - 2/(\sqrt{\ell'/\mu'}+1)\right)^{K'}$ where $\mathbf{Z}' = \frac{2}{\ell'-\mu'}\left(\mathbf{X} - \frac{\ell'+\mu'}{2}\mathbf{I}\right)$, $c_k = \frac{2}{\sqrt{\ell'\mu'}}\left(\frac{\sqrt{\mu'/\ell'}-1}{\sqrt{\mu'/\ell'}+1}\right)^k$, and $\mathbf{T}_k(\cdot)$ are matrix Chebyshev polynomials with $T_0(\mathbf{Z}') = \mathbf{I}$, $T_1(\mathbf{Z}') = \mathbf{Z}'$ and $\mathbf{T}_k(\mathbf{Z}') = 2\mathbf{Z}'\mathbf{T}_{k-1}(\mathbf{Z}') - \mathbf{T}_{k-2}(\mathbf{Z}')$ for $k \geq 2$.*

Based on Lemma 7, we can bound the approximation error of the Hessian estimator $\mathbf{H}_t$ as follows.

Lemma 8. *Using the notation of Algorithm 3, we have*

$$\left\|\nabla^2_{xx}P(\mathbf{x}_t) - \mathbf{H}_t\right\|_2 \leq 3\rho\kappa^2\sqrt{\kappa+1}\left(1 - \frac{1}{\sqrt{\kappa}}\right)^{K_t/2}\|\mathbf{y}_{t-1} - \mathbf{y}^*(\mathbf{x}_t)\|_2 + \kappa\ell\left(1 - \frac{2}{\sqrt{\kappa}+1}\right)^{K'}.$$

Lemma 8 means that taking the number of iterations for AGD with $K_t = \mathcal{O}\left(\sqrt{\kappa}\log(\kappa\rho/\varepsilon_H)\right) = \tilde{\mathcal{O}}\left(\sqrt{\kappa}\right)$[1] and the number of terms for matrix Chebyshev polynomials with $K' = \mathcal{O}\left(\sqrt{\kappa}\log(\kappa\ell/\varepsilon_H)\right) = \tilde{\mathcal{O}}\left(\sqrt{\kappa}\right)$ could achieve $\mathbf{H}_t$ with $\left\|\nabla^2 P(\mathbf{x}_t) - \mathbf{H}_t\right\|_2 \leq \varepsilon_H$ for any $\varepsilon_H > 0$.

In the implementation of IMCN, all operations related to $\mathbf{H}_t$ can be viewed as computing Hessian-vector products. Actually, we can obtain $\mathbf{H}_t\mathbf{u}'$ with $\mathcal{O}(K') = \tilde{\mathcal{O}}(\sqrt{\kappa})$ Hessian-vector calls for any $\mathbf{u}' \in \mathbb{R}^{d_x}$, which avoids $\mathcal{O}\left(d_x^2 + d_y^2\right)$ space to keep Hessian matrices. The detailed implementation is presented in Appendix E.

4.2 Complexity Analysis for IMCN

The IMCN method calls a gradient-based sub-problem solver (Algorithm 4) to optimize the following cubic regularized problem [4, 36] in each iteration

$$\min_{\mathbf{s}\in\mathbb{R}^{d_x}} \tilde{m}_t(\mathbf{s}) \triangleq \mathbf{g}_t^\top\mathbf{s} + \frac{1}{2}\mathbf{s}^\top\mathbf{H}_t\mathbf{s} + \frac{M}{6}\|\mathbf{s}\|_2^3. \tag{6}$$

It requires at most $\mathcal{K}(\varepsilon, \delta') = \mathcal{O}\left(\frac{L}{\sqrt{M\varepsilon}}\left(\log\left(\frac{\sqrt{d_x}}{\delta'}\right) + \log\left(\frac{L+\sqrt{\rho\varepsilon}}{\sqrt{M\varepsilon}}\right)\right)\right)$ number of iterations to achieve an approximate solution with enough accuracy. The detailed analysis for the complexity of Algorithm 4 is deferred to appendix D.

Then we bound the total number of iterations for Algorithm 3.

Theorem 3. *Running Algorithm 3 with $C_\sigma = 1/4$, $\delta' = \delta/T$, $T = \lceil 626(P(\mathbf{x}_0) - P^*)\sqrt{M}\varepsilon^{-1.5}\rceil$ and sufficient large K_t and K' such that Assumption 5 holds with $C_g = 1/240$ and $C_H = 1/200$. Then the condition $\Delta_t \geq -\frac{1}{128}\sqrt{\varepsilon^3/M}$ in line 11 must hold in no more than $T = \mathcal{O}\left(\kappa^{1.5}\sqrt{\rho}\varepsilon^{-1.5}\right)$ iterations; and the output $\hat{\mathbf{x}}$ is an $(\varepsilon, 2\kappa^{1.5}\sqrt{\rho\varepsilon})$-SSP with probability $1-\delta$.*

We also bound the number of gradient calls from AGD procedure in Algorithm 3 as follows.

[1] Rigorously speaking, the term $\log(\|\mathbf{y}_{t-1} - \mathbf{y}^*(\mathbf{x}_t)\|_2)$ also should be considered into the total complexity, which will be discussed in later sections.

Theorem 4. *Under the setting of Theorem 3, if we run Algorithm 3 with*

$$K' = \left\lceil \frac{\sqrt{\kappa}+1}{2} \log\left(\frac{\kappa \ell}{2\min\left\{C_H\sqrt{M\varepsilon}, \varepsilon_H L\right\}}\right)\right\rceil \text{ and } K_t = \begin{cases} \left\lceil 2\sqrt{\kappa}\log\left(\frac{\sqrt{\kappa+1}}{\tilde{\varepsilon}}\left\|\mathbf{y}^*(\mathbf{x}_0)\right\|_2\right)\right\rceil, & t=0 \\ \left\lceil 2\sqrt{\kappa}\log\left(\frac{\sqrt{\kappa+1}}{\tilde{\varepsilon}}\left(\tilde{\varepsilon}+\kappa\left\|\mathbf{s}_{t-1}\right\|_2\right)\right)\right\rceil, & t\geq 1 \end{cases}$$

where $\tilde{\varepsilon} = \min\left\{C_g\varepsilon/\ell, \min\left\{C_H\sqrt{M\varepsilon}, \varepsilon_H L\right\}/(6\rho\kappa^2)\right\}$ *and* $\mathbf{s}_0 = \mathbf{0}$, *then it holds that* $\left\|\nabla P(\mathbf{x}_t) - \mathbf{g}_t\right\|_2 \leq C_g\varepsilon$, $\left\|\nabla^2 P(\mathbf{x}_t) - \mathbf{H}_t\right\|_2 \leq \min\left\{C_H\sqrt{M\varepsilon}, \varepsilon_H L\right\}$ *and*

$$\sum_{t=0}^{T} K_t \leq T+1+\frac{2\sqrt{\kappa}T}{3}\left[\frac{3}{T}\log\left(\frac{\sqrt{\kappa+1}}{\tilde{\varepsilon}}\left\|\mathbf{y}^*(\mathbf{x}_0)\right\|_2\right)+\log\left(8(\kappa+1)^{1.5}+\frac{8\kappa^3(\kappa+1)^{1.5}}{T\tilde{\varepsilon}^3}\sum_{t=1}^{T}\left\|\mathbf{s}_{t-1}\right\|_2^3\right)\right].$$

Note that the value of $K' = \tilde{\mathcal{O}}(\sqrt{\kappa})$ corresponds to the number of Hessian vector calls for each iteration of cubic sub-problem solver (Algorithm 4 and 5). Combining Theorem 3, Theorem 4 and the value of $\mathcal{K}(\varepsilon, \delta')$, we obtain the main result for Algorithm 3 as follows.

Corollary 3. *Under Assumption 1-4, if we run Algorithm 3 with* $C_\sigma = 1/4$, $\delta' = \delta/T$,

$$K' = \left\lceil \frac{\sqrt{\kappa}+1}{2} \log\left(\frac{\kappa \ell}{2\min\left\{C_H\sqrt{M\varepsilon}, \frac{3L}{25}\right\}}\right)\right\rceil \text{ and } K_t = \begin{cases} \left\lceil 2\sqrt{\kappa}\log\left(\frac{\sqrt{\kappa+1}}{\tilde{\varepsilon}}\left\|\mathbf{y}^*(\mathbf{x}_0)\right\|_2\right)\right\rceil, & t=0 \\ \left\lceil 2\sqrt{\kappa}\log\left(\frac{\sqrt{\kappa+1}}{\tilde{\varepsilon}}\left(\tilde{\varepsilon}+\kappa\left\|\mathbf{s}_{t-1}\right\|_2\right)\right)\right\rceil, & t\geq 1 \end{cases}$$

where $T = \left\lceil 626(P(\mathbf{x}_0) - P^*)\sqrt{M}\varepsilon'^{-1.5}\right\rceil$, $C_g = 1/240$, $C_H = 1/200$, $\mathbf{s}_0 = \mathbf{0}$, $\tilde{\varepsilon}' = \varepsilon/4$ *and* $\tilde{\varepsilon} = \min\left\{C_g\varepsilon/(4\ell), \min\left\{C_H\sqrt{M\varepsilon}, \frac{3L}{25}\right\}/(12\rho\kappa^2)\right\}$, *then the output* $\hat{\mathbf{x}}$ *is an* $\left(\varepsilon, \kappa^{1.5}\sqrt{\rho\varepsilon}\right)$*-second-order stationary point of* $P(\mathbf{x})$ *with probability* $1-\delta$ *and the number of gradient calls is at most* $\tilde{\mathcal{O}}\left(\kappa^2\sqrt{\rho}\varepsilon^{-1.5}\right)$. *The number of Hessian-vector product calls is at most* $\mathcal{O}\left(\kappa^{1.5}\ell\varepsilon^{-2}\right)$.

Following the analysis of Corollary 2, we can also achieve an approximate local minimax point by running AGD based on the $\hat{\mathbf{x}}$ obtained from Algorithm 3.

5 Experiments

In this section, we conduct empirical studies for our methods against the classical GDA algorithm [20] on both synthetic problem and real-world application.

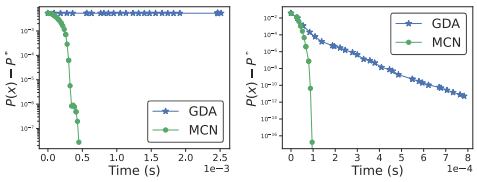
5.1 Synthetic Minimax Problem

We construct the following nonconvex-strongly-concave minimax problem:

$$\min_{\mathbf{x}\in\mathbb{R}^3}\max_{\mathbf{y}\in\mathbb{R}^2} f(\mathbf{x},\mathbf{y}) = w(x_3) - \frac{y_1^2}{40} + x_1y_1 - \frac{5y_2^2}{2} + x_2y_2 \tag{7}$$

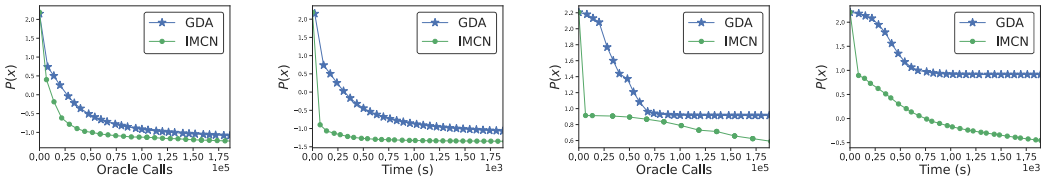
where $\mathbf{x} = [x_1, x_2, x_3]^\top$, $\mathbf{y} = [y_1, y_2]^\top$ and $w(\cdot)$ is the W-shaped scalar function [36] whose exact form is shown in Appendix G.1. It is easy to verify that the problem has an unexpected FSP at $(\mathbf{x}_0, \mathbf{y}_0) = ([0,0,0]^\top, [0,0]^\top)$.

We conduct experiments on problem (7) with two different initial points $(\mathbf{x}_1, \mathbf{y}_1) = \left([10^{-3}, 10^{-3}, 10^{-3}]^\top, [0,0]^\top\right)$ and $(\mathbf{x}_2, \mathbf{y}_2) = \left([0,0,1]^\top, [0,0]^\top\right)$. Note that the initial point $(\mathbf{x}_1, \mathbf{y}_1)$ is close to $(\mathbf{x}_0, \mathbf{y}_0)$, while $(\mathbf{x}_2, \mathbf{y}_2)$ is far from $(\mathbf{x}_0, \mathbf{y}_0)$. We compare the proposed algorithm MCN with GDA. The learning rate of GDA and AGD step in MCN is selected from $\left\{c\cdot 10^{-i} : c\in\{1,5\}, i\in\{1,2,3\}\right\}$. For MCN method, we choose $M = 10$. We compare the running time against $P(\mathbf{x}) - P^*$ for two algorithms and plot the results in Figure 1. We observe that both MCN and GDA converge to the minimum when the initial point is far from the bad stationary point, but MCN converges much faster than GDA. When the initial point is close to the bad stationary point, GDA looks hard to reach the minimum while MCN still performs well.



(a) Initial point $(\mathbf{x}_1, \mathbf{y}_1)$ (b) Initial point $(\mathbf{x}_2, \mathbf{y}_2)$

Figure 1: Results for the synthetic problem.



(a) MNIST to MNIST-m (b) MNIST to MNIST-m (c) MNIST-m to MNIST (d) MNIST-m to MNIST

Figure 2: We present results on DANN model. Figure (a) and (b) show the results of domain adaptation from MNIST to MNIST-m. Figure (c) and (d) show the results of domain adaptation from MNIST-m to MNIST.

5.2 Domain Adaptation

The Domain-Adversarial Neural Network (DANN) [9] is a classic method to domain adaptation. Suppose the source domain dataset is $\mathcal{S} = \{(\mathbf{a}_i^S, b_i^S)\}_{i=1}^{N_\mathcal{S}}$ where $\mathbf{a}_i^\mathcal{S}$ is the feature vector of the i-th sample and $b_i^\mathcal{S}$ is the corresponding label. The target domain dataset $\mathcal{T} = \{\mathbf{a}_i^\mathcal{T}\}_{i=1}^{N_\mathcal{T}}$ only contains features. Then DANN aims to solve the following nonconvex-strongly-concave minimax problem

$$\min_{[\mathbf{x}_1;\mathbf{x}_2]\in\mathbb{R}^{d_x}} \max_{\mathbf{y}\in\mathbb{R}^{d_y}} L_1(\mathbf{x}_1,\mathbf{x}_2) - \alpha \cdot L_2(\mathbf{x}_1,\mathbf{y})$$

where $L_1(\mathbf{x}_1,\mathbf{x}_2) = \frac{1}{N_\mathcal{S}}\sum_{i=1}^{N_\mathcal{S}} l(\mathbf{x}_2;\Phi(\mathbf{x}_1;\mathbf{a}_i^\mathcal{S}), b_i^\mathcal{S})$ is the loss of supervised learning and

$$L_2(\mathbf{x}_1,\mathbf{y}) = \frac{1}{N_\mathcal{S}}\sum_{i=1}^{N_\mathcal{S}} D_\mathcal{S}(h(\mathbf{y};\Phi(\mathbf{x}_1;\mathbf{a}_i^\mathcal{S}))) - \frac{1}{N_\mathcal{T}}\sum_{i=1}^{N_\mathcal{T}} D_\mathcal{T}(h(\mathbf{y};\Phi(\mathbf{x}_1;\mathbf{a}_i^\mathcal{T}))) + \lambda\|\mathbf{y}\|^2$$

is the domain classification loss. Here Φ is a single-layer neural network of size $(28\times 28)\times 200$ with parameter $\mathbf{x}_1$ and l is a two-layer neural network of size $200\times 20\times 10$ with parameter $\mathbf{x}_2$, followed by a cross entropy loss.We choose the sigmoid function as the activation function for them. With the commonly used logistic loss for L_2, we let $h(\mathbf{y};\mathbf{z}) = 1/(1+\exp(-\mathbf{y}^\top\mathbf{z}))$, $D_S(z) = 1-\log(z)$ and $D_T(z) = \log(1-z)$.

Since the dimension of the minimax problem is quite large, we implement the IMCN algorithm instead of the MCN algorithm for efficiency. We compare IMCN with GDA on the domain adaptation problem between two different datasets: MNIST [19] and MNIST-m [9]. Since we do not know the close form of $P(\mathbf{x})$, we estimate the value of $P(\mathbf{x}) = \max_\mathbf{y} f(\mathbf{x},\mathbf{y})$ by AGD procedure. More details about our experimental setup can be found in Appendix G.2.

We compare IMCN and GDA algorithms via running time and oracle calls and show the results in Figure 2. The oracle calls of GDA only contains gradient while the oracle calls of IMCN contains both gradient and Hessian-vector product. Based on Figure 2, we observe that IMCN significantly outperforms GDA in both time and oracle comparison. Notice that IMCN requires to call much more gradient/Hessian-vector oracles on y than the gradient oracles on $\mathbf{x}$. These results verify our convergence analysis and show the advantage of proposed algorithm.

6 Conclusion and Future Work

In this paper, we study second-order optimization methods for nonconvex-strongly-concave minimax problems. We have proposed a novel algorithm so-called minimax cubic Newton (MCN) which could find an $(\varepsilon, \kappa^{1.5}\sqrt{\rho\varepsilon})$-SSP of the primal function with $\mathcal{O}(\kappa^{1.5}\sqrt{\rho}\varepsilon^{-1.5})$ second-order oracle calls and $\tilde{\mathcal{O}}(\kappa^2\sqrt{\rho}\varepsilon^{-1.5})$ gradient oracle calls. We also provide an efficient algorithm for high dimensional problem, which avoids accessing second-order oracle and contains $\tilde{\mathcal{O}}(\kappa^{1.5}\ell\varepsilon^{-2})$ Hessian-vector oracle calls and $\tilde{\mathcal{O}}(\kappa^2\sqrt{\rho}\varepsilon^{-1.5})$ gradient oracle calls. To best of our knowledge, this paper first achieves non-asymptotic convergence result for finding SSP of minimax problem without convex-concave assumption. It is interesting to investigate how to find SSPs by pure first-order methods, which are more commonly used in deep learning than second-order methods.

References

[1] Naman Agarwal, Zeyuan Allen-Zhu, Brian Bullins, Elad Hazan, and Tengyu Ma. Finding approximate local minima faster than gradient descent. In *STOC*, 2017.

[2] Owe Axelsson. *Iterative Solution Methods*. Cambridge University Press, 1994.

[3] Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. Global optimality of local search for low rank matrix recovery. In *Advances in Neural Information Processing Systems*, pages 3873–3881, 2016.

[4] Yair Carmon and John Duchi. Gradient descent finds the cubic-regularized nonconvex newton step. *SIAM Journal on Optimization*, 29(3):2146–2178, 2019.

[5] Coralia Cartis, Nicholas I.M. Gould, and Philippe L. Toint. Adaptive cubic regularisation methods for unconstrained optimization. part I: motivation, convergence and numerical results. *Mathematical Programming*, 127(2):245–295, 2011.

[6] Coralia Cartis, Nicholas I.M. Gould, and Philippe L. Toint. Adaptive cubic regularisation methods for unconstrained optimization. part II: worst-case function-and derivative-evaluation complexity. *Mathematical programming*, 130(2):295–319, 2011.

[7] Ziyi Chen, Qunwei Li, and Yi Zhou. Escaping saddle points in nonconvex minimax optimization via cubic-regularized gradient descent-ascent. *arXiv preprint arXiv:2110.07098*, 2021.

[8] Tanner Fiez, Lillian Ratliff, Eric Mazumdar, Evan Faulkner, and Adhyyan Narang. Global convergence to local minmax equilibrium in classes of nonconvex zero-sum games. In *NeurIPS*, 2021.

[9] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[10] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points-online stochastic gradient for tensor decomposition. In *COLT*, 2015.

[11] Rong Ge, Jason D. Lee, and Tengyu Ma. Matrix completion has no spurious local minimum. In *NIPS*, 2016.

[12] Zhishuai Guo, Zhuoning Yuan, Yan Yan, and Tianbao Yang. Fast objective & duality gap convergence for nonconvex-strongly-concave min-max problems. *arXiv preprint arXiv:2006.06889*, 2020.

[13] Yuze Han, Guangzeng Xie, and Zhihua Zhang. Lower complexity bounds of finite-sum optimization problems: The results and construction. *arXiv preprint arXiv:2103.08280*, 2021.

[14] Filip Hanzely, Nikita Doikov, Yurii Nesterov, and Peter Richtarik. Stochastic subspace cubic newton method. In *ICML*, 2020.

[15] Feihu Huang, Shangqian Gao, Jian Pei, and Heng Huang. Accelerated zeroth-order momentum methods from mini to minimax optimization. *arXiv preprint arXiv:2008.08170*, 2020.

[16] Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan. How to escape saddle points efficiently. In *International Conference on Machine Learning*, pages 1724–1732. PMLR, 2017.

[17] Chi Jin, Praneeth Netrapalli, and Michael I. Jordan. What is local optimality in nonconvex-nonconcave minimax optimization? In *ICML*, 2020.

[18] Jonas Moritz Kohler and Aurelien Lucchi. Sub-sampled cubic regularization for non-convex optimization. In *ICML*, 2017.

[19] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[20] Tianyi Lin, Chi Jin, and Michael I. Jordan. On gradient descent ascent for nonconvex-concave minimax problems. In *ICML*, 2020.

[21] Tianyi Lin, Chi Jin, and Michael I. Jordan. Near-optimal algorithms for minimax optimization. In *COLT*, 2020.

[22] Luo Luo, Haishan Ye, Zhichao Huang, and Tong Zhang. Stochastic recursive gradient descent ascent for stochastic nonconvex-strongly-concave minimax problems. In *NeurIPS*, 2020.

[23] Luo Luo, Guangzeng Xie, Tong Zhang, and Zhihua Zhang. Near optimal stochastic algorithms for finite-sum unbalanced convex-concave minimax optimization. *arXiv preprint arXiv:2106.01761*, 2021.

[24] Eric V Mazumdar, Michael I Jordan, and S Shankar Sastry. On finding local nash equilibria (and only local nash equilibria) in zero-sum games. *arXiv preprint arXiv:1901.00838*, 2019.

[25] Yurii Nesterov. *Lectures on convex optimization*, volume 137. Springer, 2018.

[26] Yurii Nesterov and Boris T. Polyak. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

[27] Yurii E Nesterov. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. In *Dokl. akad. nauk Sssr*, volume 269, pages 543–547, 1983.

[28] Barak A. Pearlmutter. Fast exact multiplication by the Hessian. *Neural computation*, 6(1): 147–160, 1994.

[29] Shuang Qiu, Zhuoran Yang, Xiaohan Wei, Jieping Ye, and Zhaoran Wang. Single-timescale stochastic nonconvex-concave optimization for smooth nonlinear TD learning. *arXiv preprint arXiv:2008.10103*, 2020.

[30] Maziar Sanjabi, Jimmy Ba, Meisam Razaviyayn, and Jason D. Lee. On the convergence and robustness of training GANs with regularized optimal transport. *arXiv preprint arXiv:1802.08249*, 2018.

[31] Nicol N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7):1723, 2002.

[32] Alexander Shapiro. Second-order derivatives of extremal-value functions and optimality conditions for semi-infinite programs. *Mathematics of Operations Research*, 10(2):207–219, 1985.

[33] Aman Sinha, Hongseok Namkoong, and John Duchi. Certifying some distributional robustness with principled adversarial training. In *ICLR*, 2018.

[34] Ju Sun, Qing Qu, and John Wright. Complete dictionary recovery over the sphere I: Overview and the geometric picture. *IEEE Transactions on Information Theory*, 63(2):853–884, 2016.

[35] Ju Sun, Qing Qu, and John Wright. A geometric analysis of phase retrieval. *Foundations of Computational Mathematics*, 18(5):1131–1198, 2018.

[36] Nilesh Tripuraneni, Mitchell Stern, Chi Jin, Jeffrey Regier, and Michael I. Jordan. Stochastic cubic regularization for fast nonconvex optimization. In *NeurIPS*, 2018.

[37] Yuanhao Wang and Jian Li. Improved algorithms for convex-concave minimax optimization. In *NeurIPS*, 2020.

[38] Yuanhao Wang, Guodong Zhang, and Jimmy Ba. On solving minimax optimization locally: A follow-the-ridge approach. In *ICLR*, 2019.

[39] Wenhan Xian, Feihu Huang, Yanfu Zhang, and Heng Huang. A faster decentralized algorithm for nonconvex minimax problems. *NeurIPS*, 2021.

[40] Tengyu Xu, Zhe Wang, Yingbin Liang, and H Vincent Poor. Gradient free minimax optimization: Variance reduction and faster convergence. *arXiv preprint arXiv:2006.09361*, 2020.

[41] Guojun Zhang, Kaiwen Wu, Pascal Poupart, and Yaoliang Yu. Newton-type methods for minimax optimization. *arXiv preprint arXiv:2006.14592*, 2020.

[42] Siqi Zhang, Junchi Yang, Cristóbal Guzmán, Negar Kiyavash, and Niao He. The complexity of nonconvex-strongly-concave minimax optimization. *arXiv preprint arXiv:2103.15888*, 2021.

[43] Dongruo Zhou, Pan Xu, and Quanquan Gu. Stochastic variance-reduced cubic regularization methods. *Journal of Machine Learning Research*, 20(134):1–47, 2019.

Checklist

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]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (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]
   (b) Did you include complete proofs of all theoretical results? [Yes]
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]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (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)? [Yes]
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? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
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]