Near-Optimal Nonconvex-Strongly-Convex Bilevel Optimization with Fully First-Order Oracles

Lesi Chen* † chenlc23@mails.tsinghua.edu.cn
Yaohua Ma* † ma-yh21@mails.tsinghua.edu.cn
Jingzhao Zhang † jingzhaoz@mail.tsinghua.edu.cn

Abstract

Bilevel optimization has wide applications such as hyperparameter tuning, neural architecture search, and meta-learning. Designing efficient algorithms for bilevel optimization is challenging because the lower-level problem defines a feasibility set implicitly via another optimization problem. In this work, we consider one tractable case when the lower-level problem is strongly convex. Recent works show that with a Hessian-vector product oracle, one can provably find an ϵ-first-order stationary point within $\tilde{\mathcal{O}}(\epsilon^{-2})$ oracle calls. However, Hessian-vector product may be inaccessible or expensive in practice. Kwon et al. (ICML 2023) addressed this issue by proposing a first-order method that can achieve the same goal at a slower rate of $\tilde{\mathcal{O}}(\epsilon^{-3})$. In this work, we provide a tighter analysis demonstrating that this method can converge at the near-optimal $\tilde{\mathcal{O}}(\epsilon^{-2})$ rate as second-order methods. Our analysis further leads to simple first-order algorithms that achieve similar convergence rates for finding second-order stationary points and for distributed bilevel problems.

1 Introduction

In this paper, we consider the following bilevel optimization problem:

$$\min_{x\in\mathbb{R}^{d_x}} \varphi(x) := f(x, y^*(x)), \quad \text{where } y^*(x) = \arg\min_{y\in\mathbb{R}^{d_y}} g(x, y). \tag{1}$$

We call f the upper-level problem, g the lower-level problem, and φ the hyper-objective. This formulation covers many applications, including but not limited to hyperparameter tuning [8, 35, 38, 78, 87], neural architecture search [69, 107, 122, 127], meta-learning [6, 17, 30, 33, 51, 81, 88, 125], adversarial training [10, 12, 91, 103, 104, 123] and sample reweighting [37, 59, 90, 96, 119, 126].

This work focuses on the oracle complexity [82] of solving Problem 1. Since the hyper-objective $\varphi(x)$ is usually a nonconvex function, a common goal for non-asymptotic analysis is to find an approximate stationary point [7, 15, 16, 31, 32]. In general, $\varphi(x)$ can be non-differentiable or even discontinuous, so one cannot define a stationary point [18]. A common condition to ensure the differentiability of $\varphi(x)$ adopted by many existing works [19, 24, 44, 48, 49, 50, 53, 113] is to assume the lower-level strong convexity. When the lower-level problem is strongly convex in y, the so-called hyper-gradient $\nabla\varphi(x)$ [43, 87] can be expressed by:

$$\begin{aligned}\nabla\varphi(x) &= \nabla_x f(x, y^*(x)) + \nabla y^*(x)\nabla_y f(x, y^*(x)) \\ &= \nabla_x f(x, y^*(x)) - \nabla^2_{xy} g(x, y^*(x))[\nabla^2_{yy} g(x, y^*(x))]^{-1}\nabla_y f(x, y^*(x)).\end{aligned} \tag{2}$$

This equation enables one to implement second-order methods [42, 49] for nonconvex-strongly-convex bilevel optimization. These methods query second-order information of g to estimate $\nabla\varphi(x)$ via Equation 2, and then perform the so-called hyper-gradient descent on $\varphi(x)$. By using known convergence results of gradient

*Equal contributions.
†Institute for Interdisciplinary Information Sciences, Tsinghua University, China.

descent for smooth nonconvex objectives, these methods can find an ϵ-first-order stationary point of $\varphi(x)$ with $\tilde{\mathcal{O}}(\epsilon^{-2})$ first-order oracle calls from f and $\tilde{\mathcal{O}}(\epsilon^{-2})$ second-order oracle calls from g. In practice, these methods are usually implemented with the Jacobian/Hessian-vector product oracle[1]. However, in many applications [33, 84, 97, 99], calling the Jacobian/Hessian-vector product oracle may be very costly and become the computational bottleneck in bilevel optimization.

Very recently, Kwon et al. [57] proposed a novel first-order algorithm that can find an ϵ-first-order stationary point of $\varphi(x)$ without resorting to second-order information. The key idea is to exploit the following value-function reformulation [27, 38, 64, 85, 117] for Equation 1:

$$\min_{x\in\mathbb{R}^{d_x}, y\in\mathbb{R}^{d_y}} f(x,y), \quad \text{subject to} \;\; g(x,y) - g^*(x) \le 0. \tag{3}$$

Here $g^*(x) = g(x, y^*(x))$ is usually called the value-function [26, 115, 116]. The Lagrangian with a multiplier $\lambda > 0$ for this inequality-constrained problem takes the form of:

$$\mathcal{L}_\lambda(x,y) := f(x,y) + \lambda(g(x,y) - g^*(x)). \tag{4}$$

Kwon et al. [57] further define the following auxiliary function:

$$\mathcal{L}_\lambda^*(x) := \mathcal{L}_\lambda(x, y_\lambda^*(x)), \text{ where } y_\lambda^*(x) = \arg\min_{y\in\mathbb{R}^{d_y}} \mathcal{L}_\lambda(x,y), \tag{5}$$

and showed that $\mathcal{L}_\lambda^*(x)$ is a good proxy of $\varphi(x)$ with

$$\|\nabla\mathcal{L}_\lambda^*(x) - \nabla\varphi(x)\| = \mathcal{O}\left(\kappa^3/\lambda\right).$$

This indicates that when we set $\lambda \asymp \epsilon^{-1}$, an ϵ-first-order stationary point of $\mathcal{L}_\lambda^*(x)$ is also an $\mathcal{O}(\epsilon)$-first-order stationary point of $\varphi(x)$. Therefore, we can reduce the problem of finding an ϵ-first-order stationary point of $\varphi(x)$ to finding that of $\mathcal{L}_\lambda^*(x)$. The advantage of this reduction is that $\nabla\mathcal{L}_\lambda^*(x)$ can be evaluated with only first-order information of f and g:

$$\begin{aligned}\nabla\mathcal{L}_\lambda^*(x) &= \nabla_x\mathcal{L}_\lambda^*(x, y_\lambda^*(x)) + \nabla y_\lambda^*(x)\underbrace{\nabla_y\mathcal{L}_\lambda^*(x, y_\lambda^*(x))}_{=0} \\ &= \nabla_x f(x, y_\lambda^*(x)) + \lambda(\nabla_x g(x, y_\lambda^*(x)) - \nabla_x g(x, y^*(x))).\end{aligned} \tag{6}$$

Based on this property, Kwon et al. [57] proposed the Fully First-order Bilevel Approximation[2] (F^{2}BA), whose procedure is presented in Algorithm 1. Since $\mathcal{L}_\lambda(x,y)$ in Equation 4 is $\mathcal{O}(\lambda)$-gradient Lipschitz, direct analysis shows that it requires a complexity of $\tilde{\mathcal{O}}(\lambda\epsilon^{-2}) = \tilde{\mathcal{O}}(\epsilon^{-3})$ to find an ϵ-first-order stationary point, which is slower than the $\tilde{\mathcal{O}}(\epsilon^{-2})$ complexity of second-order methods. Then Kwon et al. [57] conjectured that a fundamental gap may exist between first-order and second-order methods.

However, this paper refutes this conjecture by showing that first-order methods can also achieve the near-optimal $\tilde{\mathcal{O}}(\epsilon^{-2})$ rate as second-order methods, even though less information is used. We prove that for a sufficiently large λ, the proxy $\mathcal{L}_\lambda^*(x)$ not only has the same approximate first-order stationary point of $\varphi(x)$, but also has almost the same gradient Lipschitz coefficient of $\varphi(x)$ given by:

$$\|\nabla^2\mathcal{L}_\lambda^*(x)\| \asymp \|\nabla^2\varphi(x)\| \asymp \kappa^3,$$

where κ is the condition number that will be formally defined later. This improved analysis indicates that although λ is large, the gradient Lipschitz coefficient of $\mathcal{L}_\lambda^*(x)$ would remain constant and not depend on ϵ. As a consequence, we can improve the complexity of F^{2}BA to $\tilde{\mathcal{O}}(\epsilon^{-2})$. Meanwhile, we prove this result under more relaxed assumptions than previous analyses for first-order methods, and we summarize the comparison in Table 1. As nonconvex-strongly-concave bilevel optimization problems subsume standard nonconvex optimization problems, the $\tilde{\mathcal{O}}(\epsilon^{-2})$ oracle call is also optimal up to logarithmic factors according to the lower bound provided by [15].

[1]For function $g(x,y)$ and vector v, the Jacobian-vector product oracle returns $\nabla^2_{xy}g(x,y)v$, and the Hessian-vector product oracle returns $\nabla^2_{yy}g(x,y)v$.

[2]Kwon et al. [57] called both their stochastic and deterministic method the same name Fully First-order Stochastic Approximation (F^{2}SA). Here we call the deterministic method Fully First-order Bilevel Approximation (F^{2}BA) for ease of distinction.

Table 1: We present the oracle complexities of different methods for finding an ϵ-first-order stationary point of the hyper-objective $\varphi(x) := f(x, y^*(x))$ under Assumption 3.1.

Oracle	Method	Additional Assumption	Oracle Calls
1st+2nd	AID [42]	No	$\mathcal{O}(\epsilon^{-2.5})$
	AID [49]	No	$\mathcal{O}(\epsilon^{-2})$
	ITD [49]	No	$\mathcal{O}(\epsilon^{-2}\log(1/\epsilon))$
1st	PZOBO [99]	Asmp.(a)	$\mathcal{O}(d_x^2\epsilon^{-4}\log(1/\epsilon))$
	BOME [67]	Asmp.(b)	$\mathcal{O}(\epsilon^{-6}\log(1/\epsilon))$
	F^{2}BA [57]	Asmp.(c)	$\mathcal{O}(\epsilon^{-3}\log(1/\epsilon))$
	F^{2}BA (Our Analysis)	No	$\mathcal{O}(\epsilon^{-2}\log(1/\epsilon))$

Asmp.(a): Assume $\|\nabla^2 g(x,y) - \nabla^2 g(x',y')\|_F^2 \le \rho_g^2(\|x-x'\|^2 + \|y-y'\|^2)$, stronger than Assumption 3.1c.
Asmp.(b): Additionally assume both $|f(x,y)|$ and $|g(x,y)|$ are upper bounded for any $x \in \mathbb{R}^{d_x}$ and $y \in \mathbb{R}^{d_y}$.
Asmp.(c): Additionally assume $\nabla^2 f$ is Lipschitz and both $\|\nabla_x f(x,y)\|, \|\nabla_x g(x,y)\|$ are upper bounded.

Our analysis technique is general and it can simplify algorithms in multiple setups. If we additionally assume the Hessian Lipschitz continuity of f and the third-order derivative Lipschitz continuity of g, we can show that

$$\|\nabla^2 \mathcal{L}_\lambda^*(x) - \nabla^2 \varphi(x)\| = \mathcal{O}(\kappa^6/\lambda) \quad \text{and} \quad \|\nabla^3 \mathcal{L}_\lambda^*(x)\| \asymp \|\nabla^3 \varphi(x)\| \asymp \kappa^5.$$

Based on this observation, we propose the perturbed F^{2}BA (Algorithm 2), which can provably find an ϵ-second-order stationary point of $\varphi(x)$ within $\tilde{\mathcal{O}}(\epsilon^{-2})$ first-order oracle calls. Our result shows that first-order methods can also escape saddle points in bilevel optimization like second-order methods [45].

Our algorithms can also be easily applied to the distributed scenario, when both the upper and lower-level functions adopt the following finite-sum structure:

$$f(x,y) := \frac{1}{m}\sum_{i=1}^m f_i(x,y), \quad g(x,y) := \frac{1}{m}\sum_{i=1}^m g_i(x,y), \tag{7}$$

where f_i and g_i denote the local function on the i-th agent. One of the primal challenges for second-order methods in distributed bilevel optimization arises from the Hessian inverse in the expression of hyper-gradient (Equation 2) since directly aggregating the Hessian matrix requires an expensive $\mathcal{O}(d_y^2)$ communication complexity [75, 114]. To address this issue, existing works reformulate the Hessian inverse as a sub-problem and solve it with distributed algorithms that only require communicating Hessian-vector products [22, 100, 108]. In contrast, the extension of F^{2}BA to the distributed setting (Algorithm 3) naturally avoids this additional distributed sub-solver for matrix inverse. The proposed method achieves near-optimal computation and communication complexity simultaneously.

Notations. We use notations $\mathcal{O}(\cdot)$, $\tilde{\mathcal{O}}(\cdot)$, $\Omega(\cdot)$, $\asymp$ as follows: given two functions $p : \mathbb{R}^+ \to \mathbb{R}^+$ and $q : \mathbb{R}^+ \to \mathbb{R}^+$, $p(x) = \mathcal{O}(q(x))$ means $\limsup_{x\to+\infty} p(x)/q(x) < +\infty$; $p(x) = \tilde{\mathcal{O}}(p(x))$ means there exists some positive integer $k \ge 0$ such that $p(x) = \mathcal{O}(q(x)\log^k(q(x)))$, $p(x) = \Omega(q(x))$ means $\limsup_{x\to+\infty} p(x)/q(x) > 0$, and $p(x) \asymp q(x)$ means we both have $p(x) = \mathcal{O}(q(x))$ and $p(x) = \Omega(q(x))$. We use $I_d \in \mathbb{R}^{d\times d}$ to denote a d=dimensional identity matrix. For two symmetric matrices A and B, we use $A \succeq B$ to indicate that $A - B$ is positive semidefinite. We use $\mathbb{B}(r)$ to denote the Euclidean ball centered at the origin and radius r. For a function $h : \mathbb{R}^d \to \mathbb{R}$, we use $\nabla h \in \mathbb{R}^d, \nabla^2 h \in \mathbb{R}^{d\times d}, \nabla^3 h \in \mathbb{R}^{d\times d\times d}$ to denote its gradient, Hessian, and third-order derivative, and use h^* to denote the global minimum of $h(\cdot)$. We denote $\|\cdot\|$ to be the operator norm of a tensor and more details about the notations of tensors can be found in Appendix A.

2 Related Work

Second-Order Methods for Bilevel Optimization. Most existing second-order methods for nonconvex-strongly-convex bilevel optimization can be categorized into the approximate implicit differentiation (AID) methods and the iterative differentiation (ITD) methods. The AID approach [42, 49, 63, 73] constructs hyper-gradients explicitly according to Equation 2 that $\nabla\varphi(x) = \nabla_x f(x, y^*(x)) - \nabla^2_{xy} g(x, y^*(x))v^*$, where v^* is the solution to the linear system $\nabla^2_{yy} g(x, y^*(x))v^* = \nabla_y f(x, y^*(x))$. Then one can use iterative algorithms such as fix point iteration or conjugate gradient method to solve this linear system, avoiding the computation of the Hessian inverse [43, 44, 49]. The ITD approach [28, 34, 35, 79, 93] takes advantage of the fact that backpropagation can be efficiently implemented via modern automatic differentiation frameworks such as PyTorch [86]. These methods approximate hyper-gradients by $\partial f(x, y^K(x))/\partial x$, where $y^K(x)$ is the output from K-steps of gradient descent on $g(x, \cdot)$. Although the ITD approach does not query second-order information explicitly, the analytical form of $\partial f(x, y^K(x))/\partial x$ involves second-order derivatives [49], so they also belong to second-order methods. Both AID and ITD methods require $\tilde{\mathcal{O}}(\epsilon^{-2})$ first-order oracle calls from f and $\tilde{\mathcal{O}}(\epsilon^{-2})$ second-order oracle calls from g to find an ϵ-first-order stationary point of $\varphi(x)$. Recently, Huang et al. [45] proposed the perturbed AID to find an ϵ-second-order stationary point with $\tilde{\mathcal{O}}(\epsilon^{-2})$ complexity under additional smoothness conditions of both f and g.

First-Order Methods for Bilevel Optimization. The basic idea of first-order methods for bilevel optimization is to approximate the hyper-gradient in Equation 2 using gradient information. Sow et al. [99] proposed the Partial Zeroth-Order based Bilevel Optimizer (PZOBO) that applies a zeroth-order-like estimator to approximate the response Jacobian matrix $\nabla y^*(x)$ in Equation 2, and hence the complexity has a polynomial dependency on the dimension of the problem like the standard results in zeroth-order optimization [29, 41, 56]. Liu et al. [67] first observed Equation 6 that $\nabla\mathcal{L}^*_\lambda(x)$ only involves first-order information and proposed the method named Bilevel Optimization Made Easy (BOME), but Liu et al. [67] did not provide any convergence result of $\varphi(x)$. Remarkably, Kwon et al. [57] established the relationship between $\mathcal{L}^*_\lambda(x)$ and $\varphi(x)$, and proposed the Fully First-order Bilevel Approximation (F^{2}BA) that can provably find an ϵ-first-order stationary point of $\varphi(x)$ within $\tilde{\mathcal{O}}(\epsilon^{-3})$ oracle complexity. As a by-product of their analysis, one can also show that BOME [67] also converges within $\tilde{\mathcal{O}}(\epsilon^{-6})$ first-order oracle complexity. However, before our work, we did not know whether first-order methods could have comparable theoretical guarantees to second-order methods.

Distributed Bilevel Optimization. Distributed optimization uses several agents to train a large model in parallel [11]. Despite the popularity of distributed first-order methods [61, 65, 89], the design of distributed second-order methods [68, 94, 106] is more challenging since it is inefficient to communicate the Jacobian/Hessian matrix across the agents. This also becomes the main obstacle for second-order methods in distributed bilevel optimization [46, 74, 100, 114]. Note that directly aggregating the local hyper-gradients[3] in Equation 2 on all agents cannot give the correct form of $\nabla\varphi(x)$. Therefore, all the existing distributed second-order methods for bilevel optimization are more complicated than their single-machine counterparts. Tarzanagh et al. [100] and Chen et al. [22] both discussed how one can use an additional distributed sub-solver to overcome this challenge for AID methods. Xiao and Ji [108] proposed a well-designed aggregated iterative differentiation scheme (AggITD) to conduct the hyper-gradient estimation and the lower-level optimization simultaneously.

3 Preliminaries

In this section, we introduce the three different setups studied in this work. We focus on stating the assumptions and definitions used later, while delaying a more comprehensive description to future sections where we state and describe our main results.

[3]The local hyper-gradient on the i-th agent with its local variable (x_i, y_i) refers to the quantity $\nabla_x f_i(x_i, y_i) - \nabla^2_{xy} g_i(x_i, y_i)[\nabla^2_{yy} g_i(x_i, y_i)]^{-1}\nabla_y f_i(x_i, y_i)$.

First-Order Stationary Points We first discuss the assumptions for finding first-order stationary points of the hyper-objective $\varphi(x)$, detailed below.

Assumption 3.1. *Suppose that*

a. $g(x, y)$ *is* μ*-strongly convex in* y*;*

b. $g(x, y)$ *is* L_g*-gradient Lipschitz;*

c. $g(x, y)$ *is* ρ_g*-Hessian Lipschitz;*

d. $f(x, y)$ *is* C_f*-Lipschitz in* y*;*

e. $f(x, y)$ *is* L_f*-gradient Lipschitz;*

f. $f(x, y)$ *is two-times continuous differentiable;*

g. $\varphi(x)$ *is lower bounded,* i.e. $\inf_{x\in\mathbb{R}^{d_x}} \varphi(x) > -\infty$*;*

The above assumptions are common and necessary for nonasymptotic analyses. According to Example 5.1 in [18], the hyper-objective $\varphi(x)$ can be discontinuous in general. For this reason, existing non-asymptotic analyses for bilevel optimization commonly make the lower-level strong convexity assumption (Assumption 3.1a) to ensure the existence of the hyper-gradient $\nabla\varphi(x)$. Under this lower-level strong convexity assumption, $\nabla\varphi(x)$ can be expressed jointly by $\nabla_x f(x, y)$, $\nabla_y f(x, y)$, $\nabla^2_{xy} g(x, y)$ and $\nabla^2_{yy} g(x, y)$ as Equation 2. This expression indicates that we need the smoothness conditions for f and g (Assumption 3.1b - 3.1e) to guarantee the gradient Lipschitz continuity of $\varphi(x)$. Besides these, we adopt Assumption 3.1f to ensure that $\mathcal{L}_\lambda(x, y)$ (Equation 5) is two-times continuous differentiable, and Assumption 3.1g to ensure the bilevel optimization problem (Equation 1) is well-defined.

Definition 3.1. *Under Assumption 3.1, we define the largest smoothness constant* $\ell := \max\{C_f, L_f, L_g, \rho_g\}$ *and the condition number* $\kappa := \ell/\mu$*.*

We can derive from the above assumptions that $\nabla\varphi(x)$ is uniquely defined and Lipschitz continuous, formally stated as follows.

Proposition 3.1 (Lemma 2.2 by Ghadimi and Wang [42])**.** *Under Assumption 3.1, the hyper-gradient* $\nabla\varphi(x)$ *is uniquely defined by Equation 2, and the hyper-objective* $\varphi(x)$ *is* L_φ*-gradient Lipschitz, where* $L_\varphi = \mathcal{O}(\ell\kappa^3)$*.*

As the above proposition ensures that the hyper-objective $\varphi(x)$ is differentiable, we can define the ϵ-first-order stationary points as follows.

Definition 3.2. *Given a differentiable function* $\varphi(x) : \mathbb{R}^d \to \mathbb{R}$*, we call* $\hat{x}$ *an* ϵ*-first-order stationary point of* $\varphi(x)$ *if* $\|\nabla\varphi(\hat{x})\| \le \epsilon$*.*

Second-Order Stationary Points Algorithms that pursue first-order stationary points may get stuck in saddle points and have poor performances [25]. For single-level optimization problems, there have been many researchers studying how to escape saddle points [1, 4, 5, 14, 32, 39, 52, 58, 101, 111, 121, 124]. A common assumption in these works is to suppose that the objective is Hessian Lipschitz. When generalizing to bilevel optimization, we also expect $\varphi(x)$ to be Hessian Lipschitz, which can be proved if we further assume the following higher-order smoothness condition of f and g.

Assumption 3.2. *Suppose that*

a. $f(x, y)$ *is three-times continuous differentiable;*

b. $f(x, y)$ *is* ρ_f*-Hessian Lipschitz;*

c. $g(x, y)$ *is* ν_g*-third-order derivative Lipschitz.*

Definition 3.3. *Under Assumption 3.1 and 3.2, we define the largest smoothness constant* $\ell := \max\{C_f, L_f, L_g, \rho_g, \rho_f, \nu_g\}$ *and the condition number* $\kappa := \ell/\mu$*.*

Proposition 3.2 (Lemma 3.4 by Huang et al. [45]). *Under Assumption 3.1 and 3.2, the hyper-objective $\varphi(x)$ is two-times continuously differentiable and ρ_φ-Hessian Lipschitz, where $\rho_\varphi = \mathcal{O}(\ell\kappa^5)$.*

We can then formally define the approximate second-order stationary point as follows.

Definition 3.4 (Nesterov and Polyak [83]). *Given a two-times continuously differentiable function $\varphi(x) : \mathbb{R}^d \to \mathbb{R}$ with ρ-Lipschitz Hessian, we call $\hat{x}$ an ϵ-second-order stationary point of $\varphi(x)$ if*

$$\|\nabla\varphi(\hat{x})\| \le \epsilon, \quad \nabla^2\varphi(\hat{x}) \succeq -\sqrt{\rho\epsilon} I_d.$$

We will discuss finding second-order stationary points for bilevel problems later in Section 5.

Distributed Bilevel Problems As we have discussed, first-order methods have superiority over second-order methods when the Hessian-vector product oracle is expensive. One of these scenarios is distributed bilevel optimization, where both f and g are distributed on m agents as Equation 7. The direct extension of second-order methods is to aggregate the local hyper-gradients in Equation 2 on all the agents. Unfortunately, this simple algorithm may not converge and additional techniques are required to address this challenge [22, 100, 108]. In contrast, the first-order algorithm F^{2}BA does not have this issue since aggregating the local gradients in Equation 6 gives the exact form of $\nabla\mathcal{L}_\lambda^*(x)$.

Below, we introduce the notations for the distributed F^{2}BA. Under the distributed setting, each agent i has its own local variables within the optimization algorithm: $X(i) \in \mathbb{R}^{d_x}, Y(i) \in \mathbb{R}^{d_y}, Z(i) \in \mathbb{R}^{d_y}$ [4]. For the convenience of presenting the algorithm, we aggregate all the local variables (denoted by row vectors) in one matrix and denote

$$X = \begin{bmatrix} X(1) \\ \vdots \\ X(m) \end{bmatrix} \in \mathbb{R}^{m\times d_x}, \quad Y = \begin{bmatrix} Y(1) \\ \vdots \\ Y(m) \end{bmatrix} \in \mathbb{R}^{m\times d_y} \quad \text{and} \quad Z = \begin{bmatrix} Z(1) \\ \vdots \\ Z(m) \end{bmatrix} \in \mathbb{R}^{m\times d_y},$$

We denote $\mathbf{1} = [1, \cdots, 1]^\top \in \mathbb{R}^m$ and use the lowercase with the bar to represent the mean vector, such as $\bar{x} = \frac{1}{m}\mathbf{1}^\top X$. Let $d = d_x + d_y$ and we similarly define the aggregated gradients

$$\nabla\mathbf{f}(X,Y) = \begin{bmatrix} \nabla f_1(X(1),Y(1)) \\ \vdots \\ \nabla f_m(X(m),Y(m)) \end{bmatrix} \in \mathbb{R}^{m\times d}, \ \nabla\mathbf{g}(X,Y) = \begin{bmatrix} \nabla g_1(X(1),Y(1)) \\ \vdots \\ \nabla g_m(X(m),Y(m)) \end{bmatrix} \in \mathbb{R}^{m\times d}.$$

Then we can present the update of the distributed algorithm in matrix form. More details will be provided later in Section 6.

4 Finding First-Order Stationarity

In bilevel optimization, the hyper-objective $\varphi(x)$ is usually a nonconvex function. Since finding the global minimum of a nonconvex function in the worst case requires an exponential number of queries, a common compromise is to find a local minimum [40]. First-order stationary points (Definition 3.2) are the points that satisfy the first-order necessary condition of a local minimum, which turns out to be a valid optimality criterion for nonconvex optimization.

For the task of finding an ϵ-first-order stationary point of the hyper-objective $\varphi(x)$, Kwon et al. [57] proposed the algorithm F^{2}BA (see Algorithm 1) and proved a upper complexity bound of $\tilde{\mathcal{O}}(\epsilon^{-3})$. They noted that this upper bound is worse than the $\tilde{\mathcal{O}}(\epsilon^{-2})$ near-optimal rate achieved by second-order methods, and hoped future works to investigate the fundamental limits of first-order methods. However, we give a surprising result in this section: the F^{2}BA is already near-optimal if carefully analyzed.

The detailed procedure of F^{2}BA is presented in Algorithm 1. The algorithm introduces an auxiliary variable $z \in \mathbb{R}^{d_y}$, and performs gradient descent jointly in x, y, z to solve the following optimization problem:

$$\min_{x\in\mathbb{R}^{d_x}, y\in\mathbb{R}^{d_y}} \left\{ f(x,y) + \lambda\left(g(x,y) - \min_{z\in\mathbb{R}^{d_y}} g(x,z)\right)\right\} \overset{\text{Eq.5}}{=} \min_{x\in\mathbb{R}^{d_x}} \mathcal{L}_\lambda^*(x). \tag{8}$$

[4] z is an auxiliary variable in F^{2}BA that will be introduced in the subsequent section.

Algorithm 1 F^{2}BA $(x_0, y_0, \eta, \tau, \alpha, \lambda, T, K)$

$z_0 = y_0$
for $t = 0, 1, \cdots, T-1$
  $y_t^0 = y_t,\ z_t^0 = z_t$
  for $k = 0, 1, \cdots, K-1$
    $z_t^{k+1} = z_t^k - \alpha \nabla_y g(x_t, z_t^k)$
    $y_t^{k+1} = y_t^k - \tau \left(\nabla_y f(x_t, y_t^k) + \lambda \nabla_y g(x_t, y_t^k) \right)$
  end for
  $\hat{\nabla} \mathcal{L}_\lambda^*(x_t) = \nabla_x f(x_t, y_t^K) + \lambda (\nabla_x g(x_t, y_t^K) - \nabla_x g(x_t, z_t^K))$
  $x_{t+1} = x_t - \eta \hat{\nabla} \mathcal{L}_\lambda^*(x_t)$
end for

The intuition behind the algorithm is that optimizing $\mathcal{L}_\lambda^*(x)$ is almost equivalent to optimizing $\varphi(x)$ when λ is large. Therefore, to analyze the convergence of the algorithm, we first characterize the relationship between $\mathcal{L}_\lambda^*(x)$ and $\varphi(x)$ in the following lemmas.

Lemma 4.1. *Suppose Assumption 3.1 holds. Define ℓ, κ according to Definition 3.1, and $\mathcal{L}_\lambda^*(x)$ according to Equation 5. Set $\lambda \geq 2L_f/\mu$, then it holds that*

a. $\|\nabla \mathcal{L}_\lambda^*(x) - \nabla \varphi(x)\| = \mathcal{O}(\ell \kappa^3/\lambda),\ \forall x \in \mathbb{R}^{d_x}$ (Lemma B.4).

b. $|\mathcal{L}_\lambda^*(x) - \varphi(x)| = \mathcal{O}(\ell \kappa^2/\lambda),\ \forall x \in \mathbb{R}^{d_x}$ (Lemma B.3).

c. $\mathcal{L}_\lambda^*(x)$ *is* $\mathcal{O}(\ell \kappa^3)$*-gradient Lipschitz* (Lemma B.7).

All the formal versions of these lemmas and the corresponding proofs can be found in Appendix B. Lemma 4.1a is a restatement of Lemma 3.1 by Kwon et al. [57], which demonstrates that when $\lambda \asymp \epsilon^{-1}$, an ϵ-first-order stationary point of $\mathcal{L}_\lambda^*(x)$ is also an $\mathcal{O}(\epsilon)$-first-order stationary of $\varphi(x)$. Lemma 4.1c is a new result proved in this paper. It means although $\mathcal{L}_\lambda^*(x)$ depends on λ, when λ exceeds a certain threshold, the gradient Lipschitz coefficient of $\mathcal{L}_\lambda^*(x)$ only depends on that of $\varphi(x)$ and does not depend on λ.

Since the convergence rate of gradient descent depends on the gradient Lipschitz coefficient of the objective, Lemma 4.1c indicates that optimizing $\mathcal{L}_\lambda^*(x)$ is as easy as optimizing $\varphi(x)$. Note that $\nabla \mathcal{L}_\lambda^*(x)$ only involves first-order information (Equation 6), Lemma 4.1c then tells that first-order methods can have the same convergence rate as second-order methods, as stated in the following theorem.

Theorem 4.1. *Suppose Assumption 3.1 holds. Define $\Delta := \varphi(x_0) - \inf_{x \in \mathbb{R}^{d_x}} \varphi(x)$ and $R := \|y_0 - y^*(x_0)\|^2$. Let $\eta \asymp \ell^{-1} \kappa^{-3}$, $\lambda \asymp \max\left\{\kappa/R,\ \ell\kappa^2/\Delta,\ \ell\kappa^3/\epsilon\right\}$ and set other parameters in Algorithm 1 as*

$$\alpha = \frac{1}{L_g},\ \tau = \frac{1}{2\lambda L_g},\ K = \mathcal{O}\left(\frac{L_g}{\mu} \log\left(\frac{\lambda L_g}{\mu} \right) \right),$$

then it can find an ϵ-first-order stationary point of $\varphi(x)$ within $T = \mathcal{O}(\ell \kappa^4 \epsilon^{-2} \log(\ell\kappa/\epsilon))$ first-order oracle calls, where ℓ, κ are defined in Definition 3.1.

Remark 4.1. *When the upper-level function only depends on x, i.e. we have $f(x, y) \equiv h(x)$ for some function $h(\cdot)$, the bilevel problem reduces to a single-level problem, for which Carmon et al. [15] proved a lower complexity bound of $\Omega(\epsilon^{-2})$. Therefore, we can conclude that the first-order oracle complexity of F^{2}BA we proved is near-optimal.*

We defer the proof of Theorem 4.1 to Appendix D. The complexity of F^{2}BA in Theorem 4.1 achieves the near-optimal rate in the dependency on ϵ, and matches the state-of-the-art second-order methods AID and ITD [49] in the dependency of κ. Therefore, our result, for the first time, closes the gap between first-order and second-order methods for nonconvex-strongly-convex bilevel optimization.

Remark 4.2. *The complexity of F^{2}BA may be further improved to $\tilde{\mathcal{O}}(\ell \kappa^{3.5} \epsilon^{-2})$ if we use accelerated gradient descent [82] to replace gradient descent in the inner loop. We leave that in future works.*

Algorithm 2 Perturbed F^{2}BA$\big(x_0, y_0, \eta, \tau, \alpha, \lambda, T, \{K_t\}_{t=0}^{T-1}, \epsilon, r, \mathcal{T}\big)$

1: $z_0 = y_0$
2: **for** $t = 0, 1, \cdots, T-1$
3: $\quad y_t^0 = y_t,\ z_t^0 = z_t$
4: $\quad$ **for** $k = 0, 1, \cdots, K_t - 1$
5: $\quad\quad z_t^{k+1} = z_t^k - \alpha \nabla_y g(x_t, z_t^k)$
6: $\quad\quad y_t^{k+1} = y_t^k - \tau \left(\nabla_y f(x_t, y_t^k) + \lambda \nabla_y g(x_t, y_t^k)\right)$
7: $\quad$ **end for**
8: $\quad \hat{\nabla} \mathcal{L}_\lambda^*(x_t) = \nabla_x f(x_t, y_t^{K_t}) + \lambda(\nabla_x g(x_t, y_t^{K_t}) - \nabla_x g(x_t, z_t^{K_t}))$
9: $\quad$ **if** $\|\hat{\nabla} \mathcal{L}_\lambda^*(x_t)\| \le \frac{4}{5}\epsilon$ **and** no perturbation added in the last $\mathcal{T}$ steps
10: $\quad\quad x_t = x_t - \eta \xi_t$, where $\xi_t \sim \mathbb{B}(r)$
11: $\quad$ **end if**
12: $\quad x_{t+1} = x_t - \eta \hat{\nabla} \mathcal{L}_\lambda^*(x_t)$
13: **end for**

5 Finding Second-Order Stationarity

We have shown in the previous section that the F^{2}BA is near-optimal for finding first-order stationary points. However, a first-order stationary point may be a saddle point or a local maximizer, which needs to be escaped from for an effective optimizer. For this reason, many works aim to find a second-order stationary point (Definition 3.4). In this section, we propose a simple variant of F^{2}BA (Algorithm 2) that can achieve this higher goal. The only difference to Algorithm 1 is the additional Line 9-10 in Algorithm 2, which is motivated by the perturbed strategy for escaping saddle points [52].

To prove the desired conclusion, we need to extend the analysis in Lemma 4.1 to higher-order derivatives. Below, we show that once λ is sufficiently large, $\mathcal{L}_\lambda^*(x)$ and $\varphi(x)$ have not only very close gradients (Lemma 4.1a) but also very close Hessian matrices.

Lemma 5.1. *Suppose both Assumption 3.1 and 3.2 hold. Define ℓ, κ according to Definition 3.3, and $\mathcal{L}_\lambda^*(x)$ according to Equation 5. Set $\lambda \ge 2L_f/\mu$, then it holds that*

a. $\|\nabla^2 \mathcal{L}_\lambda^*(x) - \nabla^2 \varphi(x)\| = \mathcal{O}(\ell \kappa^6/\lambda),\ \forall x \in \mathbb{R}^{d_x}$. (Lemma C.3)

b. $\mathcal{L}_\lambda^*(x)$ *is* $\mathcal{O}(\ell\kappa^5)$*-Hessian Lipschitz.* (Lemma C.2)

All the formal versions of these lemmas and the corresponding proof can be found in Appendix C. Based on these lemmas, we can prove the convergence of perturbed F^{2}BA in the following theorem. Due to some technical reasons for controlling the approximation error of $\|y_t^0 - y_\lambda^*(x_t)\|^2 + \|z_t^0 - y^*(x_t)\|^2$, we use a dynamic sequence $\{K_t\}_{t=0}^{T-1}$ that depends on Equation 9, instead of the constant $K_t \equiv K$ used in Algorithm 1.

Theorem 5.1. *Suppose both Assumption 3.1 and 3.2 hold. Define $\Delta := \varphi(x_0) - \inf_{x \in \mathbb{R}^{d_x}} \varphi(x)$ and $R := \|y_0 - y^*(x_0)\|^2$. Let $\eta \asymp \ell^{-1}\kappa^{-3}$, $\lambda \asymp \max\{\kappa/R,\ \ell\kappa^2/\Delta,\ \ell\kappa^3/\epsilon,\ \ell\kappa^6/\sqrt{\epsilon}\}$ and set other parameters in Algorithm 1 as*

$$\alpha = \frac{1}{L_g},\ \tau = \frac{1}{2\lambda L_g},\ K_t = \tilde{\mathcal{O}}\left(\frac{L_g \log \delta_t}{\mu}\right),$$

where δ_t is defined via the recursion

$$\delta_{t+1} = \frac{1}{2}\delta_t + \frac{34 L_g^2}{\mu^2}\|x_{t+1} - x_t\|^2, \quad \delta_0 = \mathcal{O}(R), \tag{9}$$

then it can find an ϵ-second-order stationary point of $\varphi(x)$ within $T = \tilde{\mathcal{O}}\left(\ell\kappa^4\epsilon^{-2}\right)$ first-order oracle calls, where ℓ, κ are defined in Definition 3.3 and the notation $\tilde{\mathcal{O}}(\cdot)$ hides logarithmic factors of d_x, κ, ℓ, and ϵ.

Algorithm 3 Distributed F^{2}BA $(\bar{x}_0, \bar{y}_0, \eta, \tau, \alpha, \lambda, T, K)$

$X_0 = \mathbf{1}\bar{x}_0,\ Y_0 = \mathbf{1}\bar{y}_0,\ Z_0 = \mathbf{1}\bar{y}_0$
for $t = 0, 1, \cdots, T-1$
  $Y_t^0 = Y_t,\ Z_t^0(i) = Z_t$
  for $k = 0, 1, \cdots, K-1$
    $V_t^k = \nabla_y \mathbf{g}(X_t, Z_t^k), \quad U_t^k = \nabla_y \mathbf{f}(X_t, Y_t^k) + \lambda \nabla_y \mathbf{g}(X_t, Y_t^k)$
    Aggregate and broadcast $\bar{v}_t^k = \frac{1}{m}\mathbf{1}\mathbf{1}^\top V_t^k,\ \bar{u}_t^k = \frac{1}{m}\mathbf{1}\mathbf{1}^\top U_t^k$
    $Z_t^{k+1} = Z_t^k - \alpha \mathbf{1}\bar{v}_t^k\ , \quad Y_t^{k+1} = Y_t^k - \tau \mathbf{1}\bar{u}_t^k$
  end for
  $H_t = \nabla_x \mathbf{f}(X_t, Y_t^K) + \lambda(\nabla_x \mathbf{g}(X_t, Y_t^K) - \nabla_x \mathbf{g}(X_t, Z_t^K))$
  Aggregate and broadcast $\bar{h}_t = \frac{1}{m}\mathbf{1}\mathbf{1}^\top H_t$
  $X_{t+1} = X_t - \eta \mathbf{1}\bar{h}_t$
end for

We defer the proof to Appendix E. The above complexity for finding ϵ-second-order stationary points matches that for finding ϵ-first-order stationary points (Theorem 4.1), up to logarithmic factors. Therefore, we conclude that F^{2}BA can escape saddle points almost for free by simply adding some small perturbation in each step.

Remark 5.1. *Although the $\tilde{\mathcal{O}}(\epsilon^{-2})$ complexity of F^{2}BA is near-optimal for finding ϵ-first-order stationary points under Assumption 3.1, the $\tilde{\mathcal{O}}(\epsilon^{-2})$ complexity of perturbed F^{2}BA may be further improved to $\tilde{\mathcal{O}}(\epsilon^{-1.75})$ by applying the restarted accelerated gradient descent [60, 112] in the outer loop. The acceleration is possible because the additional Assumption 3.2 implies the Hessian smoothness of $\varphi(x)$. And we also leave that in future works.*

6 Extension to the Distributed Scenario

Some applications of bilevel optimization like neural architecture search suffer from a heavy computational burden, so the experiments are usually conducted in a distributed way [21]. In this section, we extend F^{2}BA to the distributed setting, where both the upper and lower-level functions are distributed on m agents as Equation 7. The detailed procedure is presented in Algorithm 3.

The classic paradigm for distributed algorithms is to compute the local gradients on each agent in parallel, and then aggregate them on the server. However, challenges arise when extending existing second-order methods for bilevel optimization from the single-machine setting to the distributed setting. Given a variable $\bar{x}$, one may want to calculate its hyper-gradient (Equation 2) according to this paradigm by:

$$\hat{\nabla}\varphi(\bar{x}) = \mathbf{\Pi}\left(\nabla_x \mathbf{f}(\mathbf{1}x, \mathbf{1}y^*(\bar{x})) + \nabla_{xy}^2 \mathbf{f}(\mathbf{1}x, \mathbf{1}y^*(\bar{x}))[\nabla_{yy}^2 \mathbf{g}(\mathbf{1}\bar{x}, \mathbf{1}y^*(\bar{x}))]^{-1} \nabla_y \mathbf{f}(\mathbf{1}\bar{x}, \mathbf{1}y^*(\bar{x}))\right),$$

where $y^*(x) := \arg\min_{y \in \mathbb{R}^{d_y}} g(x, y)$ and $\mathbf{\Pi} := \frac{1}{m}\mathbf{1}\mathbf{1}^\top$ denotes the aggregation operator on the server. But the nested structure of the hyper-gradient indicates that $\hat{\nabla}\varphi(\bar{x}) \neq \nabla\varphi(\bar{x})$. As a consequence, researchers need to pay extra effort to make second-order methods work in distributed bilevel problems [22, 100]. It is a non-trivial issue to address, especially when the operator $\mathbf{\Pi}\nabla_{yy}^2 \mathbf{g}(\mathbf{1}\bar{x}, \mathbf{1}y^*(\bar{x}))$ is forbidden to avoid an unacceptable $\mathcal{O}(d_y^2)$ communication complexity.

In contrast, the F^{2}BA naturally avoid this challenge since

$$\nabla\mathcal{L}_\lambda^*(\bar{x}) = \mathbf{\Pi}\left(\nabla_x \mathbf{f}(\mathbf{1}x, \mathbf{1}y_\lambda^*(\bar{x})) + \lambda(\nabla_x \mathbf{g}(\mathbf{1}\bar{x}, \mathbf{1}y_\lambda^*(\bar{x})) - \nabla_x \mathbf{g}(\mathbf{1}\bar{x}, \mathbf{1}y^*(\bar{x})))\right),$$

where $y_\lambda^*(x) := \arg\min_{y \in \mathbb{R}^{d_y}} f(x, y) + \lambda g(x, y)$. It means that the previously mentioned classic aggregation paradigm for distributed optimization directly works for F^{2}BA without any additional modification in the algorithm, as stated below.

Proposition 6.1. *Running Algorithm 3 is equivalent to running Algorithm 1 on the mean variables*

$$\bar{x} = \frac{1}{m}\mathbf{1}^\top X,\ \bar{y} = \frac{1}{m}\mathbf{1}^\top Y \text{ and } \bar{z} = \frac{1}{m}\mathbf{1}^\top Z.$$

Then the convergence of Algorithm 3 directly follows Theorem 4.1.

Corollary 6.1. *Suppose Assumption 3.1 holds. Algorithm 3 with the same parameters in Theorem 4.1 can find X_t satisfying $\|\nabla\varphi(\bar{x}_t)\| \le \epsilon$ within $\mathcal{O}(\ell\kappa^4\epsilon^{-2}\log(\ell\kappa/\epsilon))$ first-order oracle calls and $\mathcal{O}(\ell\kappa^4\epsilon^{-2}\log(\ell\kappa/\epsilon))$ communication rounds.*

Algorithm 3 is a near-optimal distributed algorithm since both the computation and communication complexity match the $\Omega(\epsilon^{-2})$ lower bound (Theorem 1 by Lu and De Sa [76]), up to logarithmic factors. Compared with the second-order methods, the distributed F^{2}BA is more practical since it neither requires a $\mathcal{O}(d_y^2)$ communication complexity per iteration [114] nor an additional distributed sub-solver for matrix inverse [22, 100].

Remark 6.1. *Similarly, we can also implement perturbed F^{2}BA (Algorithm 2) under the distributed setting. This would lead to a distributed first-order algorithm that can provably find an ϵ-second-order stationary point with $\tilde{\mathcal{O}}(\epsilon^{-2})$ complexity.*

7 Numerical Experiments

We conduct experiments to compare the proposed first-order method F^{2}BA with existing second-order methods under both the single-machine and distributed settings. We implement all the algorithms with PyTorch [86] and conduct the experiments on a single NVIDIA A40 GPU.

7.1 Single-Machine Data Hyper-Cleaning

The task of data hyper-cleaning [35, 93, 126] considers finding the optimal weights of samples on a corrupted training set $\mathcal{D}^{\text{tr}}$, such that models trained on the weighted training set have good performances on the validation set $\mathcal{D}^{\text{val}}$. Let (a_i, b_i) be the i-th sample in the dataset, where a_i denotes the feature and b_i denotes its corresponding label. And let $\sigma(\cdot)$ be the function that clips a scalar to the interval $[0, 1]$. The problem can be formulated as the following bilevel problem, with upper and lower-level functions given by:

$$f(x,y) := \frac{1}{|\mathcal{D}^{\text{val}}|}\sum_{(a_i,b_i)\in\mathcal{D}^{\text{val}}} \ell(\langle a_i, y\rangle, b_i),$$
$$g(x,y) := \frac{1}{|\mathcal{D}^{\text{tr}}|}\sum_{(a_i,b_i)\in\mathcal{D}^{\text{tr}}} \sigma(x_i)\,\ell(\langle a_i, y\rangle, b_i) + c\|y\|^2,$$

where $\ell(\cdot,\cdot)$ is the cross-entropy loss. We set $c = 10^{-3}$ by following [49, 67, 93]. We corrupt 50% of the training set by setting their labels as random classes by following [67]. We compare F^{2}BA (Algorithm 1) with ITD [49] and AID with conjugate gradient [79] on datasets MNIST and Fashion-MNIST. For all the algorithms, we set the number of inner loops as 10 and tune the learning rates in $\{10^3, 10^2, 10^1, 10^0, 10^{-1}, 10^{-2}, 10^{-3}\}$. For F^{2}BA, we additionally tune the multiplier λ in $\{10^1, 10^2, 10^3\}$. Figure 1 shows the comparison on the computational time (seconds) against the suboptimality gap in the logarithmic scale, where the suboptimality gap is defined as the difference between the testing loss and the lowest achievable loss by running all the algorithms for 50,000 epochs.

7.2 Distributed Learnable Regularization

We also use bilevel optimization to find the optimal regularization coefficient on the 20news dataset as [49, 67]. In this formulation, the upper and lower-level functions take the form of:

$$f(x,y) := \frac{1}{|\mathcal{D}^{\text{val}}|}\sum_{(a_i,b_i)\in\mathcal{D}^{\text{val}}} \ell(\langle a_i, y\rangle, b_i), \quad g(x,y) := \frac{1}{|\mathcal{D}^{\text{tr}}|}\sum_{(a_i,b_i)\in\mathcal{D}^{\text{tr}}} \ell(\langle a_i, y\rangle, b_i) + y^\top\Sigma(x)\,y,$$

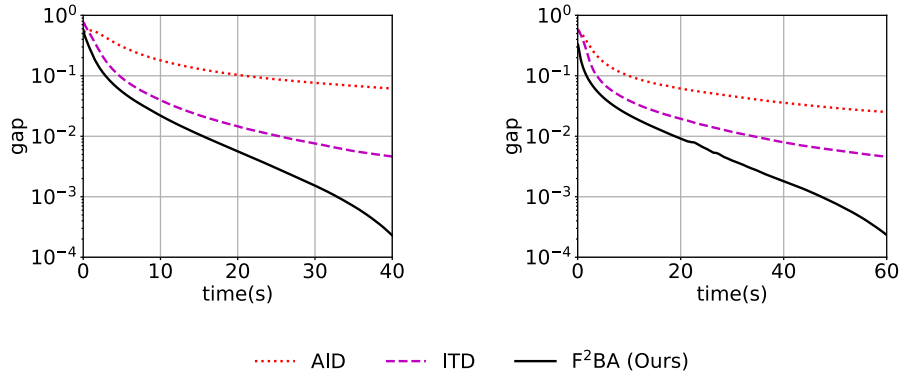

Figure 1: Experiments of data hyper-cleaning on MNIST (left) and Fashion-MNIST (right).

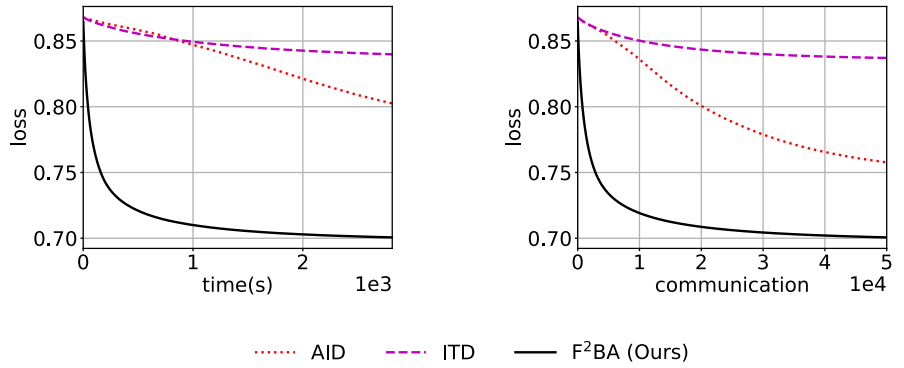

Figure 2: Experiments of distributed learnable regularization on the 20news dataset.

where $\ell(\cdot, \cdot)$ is the cross-entropy loss and $\Sigma(x) := \mathrm{diag}(\exp(x))$ determines the weights of features. We evenly distribute the datasets on four agents and use the Gloo backend to implement parallel computing. We compare the distributed F^{2}BA (Algorithm 3) with AggITD [108] and AID with HIGD oracle [22]. The hyperparameter selection strategy follows the same way as Section 7.1. We run all the algorithms for 5,000 epochs and plot the testing loss in Figure 2. It can be seen that the F^{2}BA is more efficient than second-order methods in both computation and communication.

8 Conclusions and Future Directions

This paper shows how to achieve the near-optimal complexity for first-order methods in nonconvex-strongly-convex bilevel optimization. Our results close the gap between first-order and second-order methods for bilevel optimization under various setups. We conclude this paper with several potential directions for future research.

Acceleration. As discussed in our remarks, it is possible to apply acceleration techniques to improve the complexity dependency on κ for finding both first-order and second-order stationary points, and the dependency on ϵ for finding second-order stationary points.

Logarithmic factors. The complexity of F^{2}BA has an additional $\log(1/\epsilon)$ factor compared with the lower bound for finding first-order stationary points [15]. Although this factor can be shaved for second-order

methods using tighter analysis [49], it remains open whether it is avoidable or unavoidable for first-order methods.

Single-Loop Methods. Our method adopts a double-loop structure. Designing single-loop methods could be more efficient in certain scenarios, but the analysis would also be more challenging [19, 20, 23, 44, 53, 57, 66].

Stochastic Bilevel Problems. This paper only considers the deterministic case. It is of great significance to study first-order methods using stochastic oracles [47, 49, 53, 57, 113] in future works.

Distributed Bilevel Problems. We believe that the distributed first-order methods would have broader application prospects than second-order methods and anticipate our algorithm could be applied to more distributed scenarios [2, 3, 9, 36, 62, 80, 92, 105, 118, 120].

More Lower-Level Problems. Assumption 3.1 is somewhat restricted. It would be important to study the convergence of first-order algorithms for bilevel problems under more relaxed assumptions on the lower-level problems, such as nonsmooth lower-level problems [77], constrained lower-level problems [54, 72, 102, 110], and bilevel problems without lower-level strong convexity [18, 70, 71, 72, 95, 98, 109].

References

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

[2] Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li. Byzantine stochastic gradient descent. In *NeurIPS*, 2018.

[3] Dan Alistarh, Torsten Hoefler, Mikael Johansson, Nikola Konstantinov, Sarit Khirirat, and Cédric Renggli. The convergence of sparsified gradient methods. In *NeurIPS*, 2018.

[4] Zeyuan Allen-Zhu. Natasha 2: Faster non-convex optimization than SGD. In *NeurIPS*, 2018.

[5] Zeyuan Allen-Zhu and Yuanzhi Li. Neon2: Finding local minima via first-order oracles. In *NeurIPS*, 2018.

[6] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *NeurIPS*, 2016.

[7] Yossi Arjevani, Yair Carmon, John C. Duchi, Dylan J. Foster, Nathan Srebro, and Blake Woodworth. Lower bounds for non-convex stochastic optimization. *Mathematical Programming*, 199(1-2):165–214, 2023.

[8] Juhan Bae and Roger B. Grosse. Delta-STN: Efficient bilevel optimization for neural networks using structured response jacobians. In *NeurIPS*, 2020.

[9] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed optimisation for non-convex problems. In *ICML*, 2018.

[10] Nicholas Bishop, Long Tran-Thanh, and Enrico Gerding. Optimal learning from verified training data. In *NeurIPS*, 2020.

[11] Stephen Boyd. Convex optimization: from embedded real-time to large-scale distributed. In *KDD*, 2011.

[12] Michael Brückner and Tobias Scheffer. Stackelberg games for adversarial prediction problems. In *SIGKDD*, 2011.

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

[14] Yair Carmon, John C. Duchi, Oliver Hinder, and Aaron Sidford. "Convex Until Proven Guilty": Dimension-free acceleration of gradient descent on non-convex functions. In *ICML*, 2017.

[15] Yair Carmon, John C. Duchi, Oliver Hinder, and Aaron Sidford. Lower bounds for finding stationary points I. *Mathematical Programming*, 184(1-2):71–120, 2020.

[16] Yair Carmon, John C. Duchi, Oliver Hinder, and Aaron Sidford. Lower bounds for finding stationary points II: first-order methods. *Mathematical Programming*, 185(1):315–355, 2021.

[17] Kartik Chandra, Audrey Xie, Jonathan Ragan-Kelley, and Erik Meijer. Gradient descent: The ultimate optimizer. In *NeurIPS*, 2022.

[18] Lesi Chen, Jing Xu, and Jingzhao Zhang. Bilevel optimization without lower-level strong convexity from the hyper-objective perspective. *arXiv preprint arXiv:2301.00712*, 2023.

[19] Tianyi Chen, Yuejiao Sun, and Wotao Yin. Closing the gap: Tighter analysis of alternating stochastic gradient methods for bilevel problems. In *NeurIPS*, 2021.

[20] Tianyi Chen, Yuejiao Sun, and Wotao Yin. A single-timescale stochastic bilevel optimization method. In *AISTATS*, 2022.

[21] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *ICCV*, 2019.

[22] Xuxing Chen, Minhui Huang, Shiqian Ma, and Krishnakumar Balasubramanian. Decentralized stochastic bilevel optimization with improved per-iteration complexity. In *ICML*, 2023.

[23] Xuxing Chen, Tesi Xiao, and Krishnakumar Balasubramanian. Optimal algorithms for stochastic bilevel optimization under relaxed smoothness conditions. *arXiv preprint arXiv:2306.12067*, 2023.

[24] Mathieu Dagréou, Pierre Ablin, Samuel Vaiter, and Thomas Moreau. A framework for bilevel optimization that enables stochastic and global variance reduction algorithms. In *NeurIPS*, 2022.

[25] Yann N. Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *NIPS*, 2014.

[26] Stephan Dempe. *Foundations of bilevel programming*. Springer Science & Business Media, 2002.

[27] Stephan Dempe and Alain Zemkoho. Bilevel optimization. *Springer optimization and its applications*, 161, 2020.

[28] Justin Domke. Generic methods for optimization-based modeling. In *AISTATS*, 2012.

[29] John C. Duchi, Michael I. Jordan, Martin J. Wainwright, and Andre Wibisono. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61(5):2788–2806, 2015.

[30] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In *AISTATS*, 2020.

[31] Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *NeurIPS*, 2018.

[32] Cong Fang, Zhouchen Lin, and Tong Zhang. Sharp analysis for nonconvex sgd escaping from saddle points. In *COLT*, 2019.

[33] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

[34] Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. Forward and reverse gradient-based hyperparameter optimization. In *ICML*, 2017.

[35] Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *ICML*, 2018.

[36] Venkata Gandikota, Daniel Kane, Raj Kumar Maity, and Arya Mazumdar. vqSGD: Vector quantized stochastic gradient descent. In *AISTATS*, 2021.

[37] Jiahui Gao, Renjie Pi, LIN Yong, Hang Xu, Jiacheng Ye, Zhiyong Wu, WeiZhong Zhang, Xiaodan Liang, Zhenguo Li, and Lingpeng Kong. Self-guided noise-free data generation for efficient zero-shot learning. In *ICLR*, 2022.

[38] Lucy Gao, Jane J. Ye, Haian Yin, Shangzhi Zeng, and Jin Zhang. Value function based difference-of-convex algorithm for bilevel hyperparameter selection problems. In *ICML*, 2022.

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

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

[41] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

[42] Saeed Ghadimi and Mengdi Wang. Approximation methods for bilevel programming. *arXiv preprint arXiv:1802.02246*, 2018.

[43] Riccardo Grazzi, Luca Franceschi, Massimiliano Pontil, and Saverio Salzo. On the iteration complexity of hypergradient computation. In *ICML*, 2020.

[44] Mingyi Hong, Hoi-To Wai, Zhaoran Wang, and Zhuoran Yang. A two-timescale stochastic algorithm framework for bilevel optimization: Complexity analysis and application to actor-critic. *SIAM Journal on Optimization*, 33(1):147–180, 2023.

[45] Minhui Huang, Kaiyi Ji, Shiqian Ma, and Lifeng Lai. Efficiently escaping saddle points in bilevel optimization. *arXiv preprint arXiv:2202.03684*, 2022.

[46] Minhui Huang, Dewei Zhang, and Kaiyi Ji. Achieving linear speedup in non-iid federated bilevel learning. In *ICML*, 2023.

[47] Haimei Huo, Risheng Liu, and Zhixun Su. A new simple stochastic gradient descent type algorithm with lower computational complexity for bilevel optimization. *arXiv preprint arXiv:2306.11211*, 2023.

[48] Kaiyi Ji and Yingbin Liang. Lower bounds and accelerated algorithms for bilevel optimization. *JMLR*, 2021.

[49] Kaiyi Ji, Junjie Yang, and Yingbin Liang. Bilevel optimization: Convergence analysis and enhanced design. In *ICML*, 2021.

[50] Kaiyi Ji, Mingrui Liu, Yingbin Liang, and Lei Ying. Will bilevel optimizers benefit from loops. In *NeurIPS*, 2022.

[51] Kaiyi Ji, Junjie Yang, and Yingbin Liang. Theoretical convergence of multi-step model-agnostic meta-learning. *JMLR*, 2022.

[52] Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M. Kakade, and Michael I. Jordan. How to escape saddle points efficiently. In *ICML*, 2017.

[53] Prashant Khanduri, Siliang Zeng, Mingyi Hong, Hoi-To Wai, Zhaoran Wang, and Zhuoran Yang. A near-optimal algorithm for stochastic bilevel optimization via double-momentum. In *NeurIPS*, 2021.

[54] Prashant Khanduri, Ioannis Tsaknakis, Yihua Zhang, Jia Liu, Sijia Liu, Jiawei Zhang, and Mingyi Hong. Linearly constrained bilevel optimization: A smoothed implicit gradient approach. In *ICML*, 2023.

[55] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3): 455–500, 2009.

[56] Guy Kornowski and Ohad Shamir. An algorithm with optimal dimension-dependence for zero-order nonsmooth nonconvex stochastic optimization. *arXiv preprint arXiv:2307.04504*, 2023.

[57] Jeongyeol Kwon, Dohyun Kwon, Stephen Wright, and Robert Nowak. A fully first-order method for stochastic bilevel optimization. In *ICML*, 2023.

[58] Jason D. Lee, Max Simchowitz, Michael I. Jordan, and Benjamin Recht. Gradient descent only converges to minimizers. In *COLT*, 2016.

[59] Haoyang Li, Xin Wang, Ziwei Zhang, and Wenwu Zhu. OOD-GNN: Out-of-distribution generalized graph neural network. *TKDE*, 2022.

[60] Huan Li and Zhouchen Lin. Restarted nonconvex accelerated gradient descent: No more polylogarithmic factor in the $\mathcal{O}(\varepsilon^{-7/4})$ complexity. In *ICML*, 2022.

[61] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. In *ICLR*, 2020.

[62] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *NeurIPS*, 2017.

[63] Renjie Liao, Yuwen Xiong, Ethan Fetaya, Lisa Zhang, KiJung Yoon, Xaq Pitkow, Raquel Urtasun, and Richard Zemel. Reviving and improving recurrent back-propagation. In *ICML*, 2018.

[64] Gui-Hua Lin, Mengwei Xu, and Jane J. Ye. On solving simple bilevel programs with a nonconvex lower level program. *Mathematical Programming*, 144(1-2):277–305, 2014.

[65] Tao Lin, Sebastian Urban Stich, Kumar Kshitij Patel, and Martin Jaggi. Don't use large mini-batches, use local SGD. In *ICLR*, 2019.

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

[67] Bo Liu, Mao Ye, Stephen Wright, Peter Stone, and Qiang Liu. BOME! bilevel optimization made easy: A simple first-order approach. In *NeurIPS*, 2022.

[68] Chengchang Liu, Lesi Chen, Luo Luo, and John Lui. Communication efficient distributed newton method with fast convergence rates. In *KDD*, 2023.

[69] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *ICLR*, 2019.

[70] Risheng Liu, Pan Mu, Xiaoming Yuan, Shangzhi Zeng, and Jin Zhang. A generic first-order algorithmic framework for bi-level programming beyond lower-level singleton. In *ICML*, 2020.

[71] Risheng Liu, Xuan Liu, Xiaoming Yuan, Shangzhi Zeng, and Jin Zhang. A value-function-based interior-point method for non-convex bi-level optimization. In *ICML*, 2021.

[72] Risheng Liu, Yaohua Liu, Shangzhi Zeng, and Jin Zhang. Towards gradient-based bilevel optimization with non-convex followers and beyond. *NeurIPS*, 2021.

[73] Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *AISTATS*, 2020.

[74] Songtao Lu, Xiaodong Cui, Mark S. Squillante, Brian Kingsbury, and Lior Horesh. Decentralized bilevel optimization for personalized client learning. In *ICASSP*, 2022.

[75] Songtao Lu, Siliang Zeng, Xiaodong Cui, Mark Squillante, Lior Horesh, Brian Kingsbury, Jia Liu, and Mingyi Hong. A stochastic linearized augmented lagrangian method for decentralized bilevel optimization. In *NeurIPS*, 2022.

[76] Yucheng Lu and Christopher De Sa. Optimal complexity in decentralized training. In *ICML*, 2021.

[77] Zhaosong Lu and Sanyou Mei. First-order penalty methods for bilevel optimization. *arXiv preprint arXiv:2301.01716*, 2023.

[78] Matthew Mackay, Paul Vicol, Jonathan Lorraine, David Duvenaud, and Roger Grosse. Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions. In *ICML*, 2018.

[79] Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *ICML*, 2015.

[80] Konstantin Mishchenko, Grigory Malinovsky, Sebastian Stich, and Peter Richtárik. Proxskip: Yes! local gradient steps provably lead to communication acceleration! finally! In *ICML*, 2022.

[81] Konstantin Mishchenko, Slavomír Hanzely, and Peter Richtárik. Convergence of first-order algorithms for meta-learning with moreau envelopes. *arXiv preprint arXiv:2301.06806*, 2023.

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

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

[84] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[85] Jiří V. Outrata. On the numerical solution of a class of stackelberg problems. *Zeitschrift für Operations Research*, 34:255–277, 1990.

[86] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[87] Fabian Pedregosa. Hyperparameter optimization with approximate gradient. In *ICML*, 2016.

[88] Aravind Rajeswaran, Chelsea Finn, Sham M. Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *NeurIPS*, 2019.

[89] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. HOGWILD!: A lock-free approach to parallelizing stochastic gradient descent. In *NeurIPS*, 2011.

[90] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *ICML*, 2018.

[91] Alexander Robey, Fabian Latorre, George J. Pappas, Hamed Hassani, and Volkan Cevher. Adversarial training should be cast as a non-zero-sum game. *arXiv preprint arXiv:2306.11035*, 2023.

[92] Kevin Scaman, Francis Bach, Sébastien Bubeck, Yin Tat Lee, and Laurent Massoulié. Optimal algorithms for smooth and strongly convex distributed optimization in networks. In *ICML*, 2017.

[93] Amirreza Shaban, Ching-An Cheng, Nathan Hatch, and Byron Boots. Truncated back-propagation for bilevel optimization. In *AISTATS*, 2019.

[94] Ohad Shamir, Nati Srebro, and Tong Zhang. Communication-efficient distributed optimization using an approximate newton-type method. In *ICML*, 2014.

[95] Han Shen and Tianyi Chen. On penalty-based bilevel gradient descent method. In *ICML*, 2023.

[96] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. In *NeurIPS*, 2019.

[97] Xingyou Song, Wenbo Gao, Yuxiang Yang, Krzysztof Choromanski, Aldo Pacchiano, and Yunhao Tang. ES-MAML: Simple hessian-free meta learning. In *ICLR*, 2019.

[98] Daouda Sow, Kaiyi Ji, Ziwei Guan, and Yingbin Liang. A constrained optimization approach to bilevel optimization with multiple inner minima. *arXiv preprint arXiv:2203.01123*, 2022.

[99] Daouda Sow, Kaiyi Ji, and Yingbin Liang. On the convergence theory for hessian-free bilevel algorithms. In *NeurIPS*, 2022.

[100] Davoud Ataee Tarzanagh, Mingchen Li, Christos Thrampoulidis, and Samet Oymak. Fednest: Federated bilevel, minimax, and compositional optimization. In *ICML*, 2022.

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

[102] Ioannis Tsaknakis, Prashant Khanduri, and Mingyi Hong. An implicit gradient-type method for linearly constrained bilevel problems. In *ICASSP*, 2022.

[103] Jiali Wang, He Chen, Rujun Jiang, Xudong Li, and Zihao Li. Fast algorithms for stackelberg prediction game with least squares loss. In *ICML*, 2021.

[104] Jiali Wang, Wen Huang, Rujun Jiang, Xudong Li, and Alex L. Wang. Solving stackelberg prediction game with least squares loss via spherically constrained least squares reformulation. In *ICML*, 2022.

[105] Jue Wang, Binhang Yuan, Luka Rimanic, Yongjun He, Tri Dao, Beidi Chen, Christopher Ré, and Ce Zhang. Fine-tuning language models over slow networks using activation quantization with guarantees. In *NeurIPS*, 2022.

[106] Shusen Wang, Fred Roosta, Peng Xu, and Michael W. Mahoney. GIANT: Globally improved approximate newton method for distributed optimization. In *NeurIPS*, 2018.

[107] Xiaoxing Wang, Wenxuan Guo, Jianlin Su, Xiaokang Yang, and Junchi Yan. ZARTS: On zero-order optimization for neural architecture search. In *NeurIPS*, 2022.

[108] Peiyao Xiao and Kaiyi Ji. Communication-efficient federated hypergradient computation via aggregated iterative differentiation. In *ICML*, 2023.

[109] Quan Xiao, Songtao Lu, and Tianyi Chen. A generalized alternating method for bilevel optimization under the Polyak-Łojasiewicz condition. *arXiv preprint arXiv:2306.02422*, 2023.

[110] Quan Xiao, Han Shen, Wotao Yin, and Tianyi Chen. Alternating projected SGD for equality-constrained bilevel optimization. In *AISTATS*, 2023.

[111] Yi Xu, Rong Jin, and Tianbao Yang. First-order stochastic algorithms for escaping from saddle points in almost linear time. In *NeurIPS*, 2018.

[112] Haikuo Yang, Luo Luo, Chris Junchi Li, and Michael I. Jordan. Accelerating inexact hypergradient descent for bilevel optimization. *arXiv preprint arXiv:2307.00126*, 2023.

[113] Junjie Yang, Kaiyi Ji, and Yingbin Liang. Provably faster algorithms for bilevel optimization. *NeurIPS*, 2021.

[114] Shuoguang Yang, Xuezhou Zhang, and Mengdi Wang. Decentralized gossip-based stochastic bilevel optimization over communication networks. In *NeurIPS*, 2022.

[115] Jane J. Ye and Daoli Zhu. Optimality conditions for bilevel programming problems. *Optimization*, 33 (1):9–27, 1995.

[116] Jane J. Ye and Daoli Zhu. New necessary optimality conditions for bilevel programs by combining the mpec and value function approaches. *SIAM Journal on Optimization*, 20(4):1885–1905, 2010.

[117] Jane J. Ye, Xiaoming Yuan, Shangzhi Zeng, and Jin Zhang. Difference of convex algorithms for bilevel programs with applications in hyperparameter selection. *Mathematical Programming*, 198(2): 1583–1616, 2023.

[118] Min Ye and Emmanuel Abbe. Communication-computation efficient gradient coding. In *ICML*, 2018.

[119] Lin Yong, Renjie Pi, Weizhong Zhang, Xiaobo Xia, Jiahui Gao, Xiao Zhou, Tongliang Liu, and Bo Han. A holistic view of label noise transition matrix in deep learning and beyond. In *ICLR*, 2022.

[120] Binhang Yuan, Yongjun He, Jared Davis, Tianyi Zhang, Tri Dao, Beidi Chen, Percy S. Liang, Christopher Re, and Ce Zhang. Decentralized training of foundation models in heterogeneous environments. In *NeurIPS*, 2022.

[121] Chenyi Zhang and Tongyang Li. Escape saddle points by a simple gradient-descent based algorithm. In *NeurIPS*, 2021.

[122] Miao Zhang, Steven W Su, Shirui Pan, Xiaojun Chang, Ehsan M Abbasnejad, and Reza Haffari. iDARTS: Differentiable architecture search with stochastic implicit gradients. In *ICML*, 2021.

[123] Yihua Zhang, Guanhua Zhang, Prashant Khanduri, Mingyi Hong, Shiyu Chang, and Sijia Liu. Revisiting and advancing fast adversarial training through the lens of bi-level optimization. In *ICML*, 2022.

[124] Dongruo Zhou, Pan Xu, and Quanquan Gu. Stochastic nested variance reduction for nonconvex optimization. *JMLR*, 2020.

[125] Pan Zhou, Xiaotong Yuan, Huan Xu, Shuicheng Yan, and Jiashi Feng. Efficient meta learning via minibatch proximal update. In *NeurIPS*, 2019.

[126] Xiao Zhou, Yong Lin, Renjie Pi, Weizhong Zhang, Renzhe Xu, Peng Cui, and Tong Zhang. Model agnostic sample reweighting for out-of-distribution learning. In *ICML*, 2022.

[127] Barret Zoph and Quoc Le. Neural architecture search with reinforcement learning. In *ICLR*, 2016.

A Notations for Tensors and Derivatives

We follow the notations of tensors used in [55]. For a three-way tensor $\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$, we use $[\mathcal{X}]_{i_1,i_2,i_3}$ to represent its (i_1, i_2, i_3)-th element. The inner product of two three-way tensors $\mathcal{X}, \mathcal{Y}$ is defined by $\langle \mathcal{X}, \mathcal{Y} \rangle := \sum_{i_1,i_2,i_3} [\mathcal{X}]_{i_1,i_2,i_3} \cdot [\mathcal{Y}]_{i_1,i_2,i_3}$. The operator norm of three-way tensor $\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ is defined by $\|\mathcal{X}\| := \sup_{\|x_i\|=1} \langle \mathcal{X}, x_1 \circ x_2 \circ x_3 \rangle$, where the elements of $x_1 \circ x_2 \circ x_3 \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ is $[x_1 \circ x_2 \circ x_3]_{i_1,i_2,i_3} := [x_1]_{i_1} \cdot [x_2]_{i_2} \cdot [x_3]_{i_3}$. It can be verified that such a definition generalizes the Euclidean norm of a vector and the spectral norm of a matrix to tensors. For a three-way tensor $\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ and a vector $v \in \mathbb{R}^{d_1}$, their mode-1 product, denoted by $\mathcal{X} \bar{\times}_1 v$, gives a matrix in $\mathbb{R}^{d_2 \times d_3}$ with elements $[\mathcal{X} \bar{\times}_1 v]_{i_2,i_3} := \sum_{i_1} [\mathcal{X}]_{i_1,i_2,i_3} \cdot [v]_{i_1}$. We define $\bar{\times}_2$ and $\bar{\times}_3$ in a similar way, and it can be verified that $\|\mathcal{X} \bar{\times}_i v\| \leq \|\mathcal{X}\| \|v\|$. For a three-way tensor $\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ and a matrix $A \in \mathbb{R}^{d_1' \times d_1}$, their mode-1 product, denoted by $\mathcal{X} \times_1 A$, gives a tensor in $\mathbb{R}^{d_1' \times d_2 \times d_3}$ with elements $[\mathcal{X} \times_1 A]_{i_1',i_2,i_3} := \sum_{i_1} [\mathcal{X}]_{i_1,i_2,i_3} \cdot [A]_{i_1',i_1}$. We define $\times_2$ and $\times_3$ in a similar way, and it can also be verified that $\|\mathcal{X} \times_i A\| \leq \|\mathcal{X}\| \|A\|$.

For a function $h(x, y) : \mathbb{R}^{d_x} \times \mathbb{R}^{d_y} \to \mathbb{R}$, we denote $\nabla_x h(x, y) \in \mathbb{R}^{d_x}$ to be the partial gradient with respect to x, with elements given by $[\nabla_x h(x, y)]_i := \partial f(x, y)/\partial [x]_i$. And we define $\nabla_y h(x, y) \in \mathbb{R}^{d_y}$ in a similar way. We denote $\nabla^2_{xx} h(x, y) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_x}$ to be the partial Hessian with respect to x, with elements given by $[\nabla^2_{xx} h(x, y)]_{i_1,i_2} := \partial^2 h(x, y)/(\partial [x]_{i_1} \partial [x]_{i_2})$. And we define $\nabla^2_{xy} h(x, y) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$, $\nabla^2_{yx} h(x, y) \in \mathbb{R}^{d_y} \times \mathbb{R}^{d_x}$ and $\nabla^2_{yy} h(x, y) \in \mathbb{R}^{d_y} \times \mathbb{R}^{d_y}$ in a similar way. We denote $\nabla^3_{xxx} h(x, y) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_x} \times \mathbb{R}^{d_x}$ to be the partial third-order derivative with respect to x, with elements given by $[\nabla^3_{xxx} h(x, y)]_{i_1,i_2,i_3} := \partial^3 h(x, y)/(\partial [x]_{i_1} \partial [x]_{i_2} \partial [x]_{i_3})$. And we define $\nabla^3_{xxy} h(x, y) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$, $\nabla^3_{xyx} h(x, y) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_y} \times \mathbb{R}^{d_x}$, $\nabla^3_{xyy} h(x, y) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_y} \times \mathbb{R}^{d_y}$, $\nabla^3_{yyy} h(x, y) \in \mathbb{R}^{d_y} \times \mathbb{R}^{d_y} \times \mathbb{R}^{d_y}$, $\nabla^3_{yyx} h(x, y) \in \mathbb{R}^{d_y} \times \mathbb{R}^{d_y} \times \mathbb{R}^{d_x}$, $\nabla^3_{yxy} h(x, y) \in \mathbb{R}^{d_y} \times \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$ and$\nabla^3_{yxx} h(x, y) \in \mathbb{R}^{d_y} \times \mathbb{R}^{d_x} \times \mathbb{R}^{d_x}$ in a similar way. We denote $\nabla y^*(x) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$ to be the matrix with elements given by $[\nabla y^*(x)]_{i_1,i_2} := \partial [y^*(x)]_{i_2} / \partial [x]_{i_1}$. We denote $\nabla^2 y^*(x) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$ to be the three-way tensor with elements given by $[\nabla^2 y^*(x)]_{i_1,i_2,i_3} := \partial^2 [y^*(x)]_{i_3} / (\partial [x]_{i_1} \partial [x]_{i_2})$. And we define $\nabla y^*_\lambda(x)$ and $\nabla^2 y^*_\lambda(x)$ in a similar way.

B Lemmas for Finding First-Order Stationarity

Lemma B.1 (From Kwon et al. [57])**.** *Under Assumption 3.1, for $\lambda \geq 2L_f/\mu$, $\mathcal{L}_\lambda(x, \cdot)$ is $(\lambda\mu/2)$-strongly convex.*

Lemma B.2. *Under Assumption 3.1, for $\lambda \geq 2L_f/\mu$, it holds that*

$$\|y^*_\lambda(x) - y^*(x)\| \leq \frac{C_f}{\lambda\mu}$$

Proof. By the first-order optimality condition, we know that

$$\nabla_y f(x, y^*_\lambda(x)) + \lambda \nabla_y g(x, y^*_\lambda(x)) = 0.$$

Then we have

$$\|y^*_\lambda(x) - y^*(x)\| \leq \frac{1}{\mu} \|\nabla_y g(x, y^*_\lambda(x))\| = \frac{1}{\lambda\mu} \|\nabla_y f(x, y^*_\lambda(x))\| \leq \frac{C_f}{\lambda\mu}$$

□

As a direct consequence, we can show that $\mathcal{L}^*_x$ and $\varphi(x)$ are close.

Lemma B.3. *Under Assumption 3.1, for $\lambda \geq 2L_f/\mu$, it holds that $|\mathcal{L}^*_\lambda(x) - \varphi(x)| \leq D_0/\lambda$, where*

$$D_0 := \left(C_f + \frac{C_f L_g}{2\mu} \right) \frac{C_f}{\mu} = \mathcal{O}(\ell\kappa^2).$$

Proof. A simple calculus shows that

$$|\mathcal{L}^*_\lambda(x) - \varphi(x)|$$

$$
\begin{aligned}
&\leq |f(x, y_\lambda^*(x)) - f(x, y^*(x))| + \lambda |g(x, y_\lambda^*(x)) - g(x, y^*(x))| \\
&\leq C_f \|y_\lambda^*(x) - y^*(x)\| + \frac{\lambda L_g}{2} \|y_\lambda^*(x) - y^*(x)\|^2 \\
&\leq \left(C_f + \frac{C_f L_g}{2\mu} \right) \|y_\lambda^*(x) - y^*(x)\| \\
&\leq \left(C_f + \frac{C_f L_g}{2\mu} \right) \frac{C_f}{\lambda \mu}.
\end{aligned}
$$

□

The following result is the key to designing fully first-order methods for bilevel optimization, first proved in Lemma 3.1 in [57]. We provide the proof here for completeness.

Lemma B.4. *Under Assumption 3.1, for $\lambda \geq 2L_f/\mu$, it holds that $\|\nabla \mathcal{L}_\lambda^*(x) - \nabla \varphi(x)\| \leq D_1/\lambda$, where*

$$
D_1 := \left(L_f + \frac{\rho_g L_g}{\mu} + \frac{C_f L_g \rho_g}{2\mu^2} + \frac{C_f \rho_g}{2\mu} \right) \frac{C_f}{\mu} = \mathcal{O}(\ell \kappa^3). \tag{10}
$$

Proof. Taking total derivative on $\varphi(x) = f(x, y^*(x))$, we obtain the following result:

$$
\nabla \varphi(x) = \nabla_x f(x, y^*(x)) - \nabla_{xy}^2 g(x, y^*(x)) [\nabla_{yy} g(x, y^*(x))]^{-1} \nabla_y f(x, y^*(x)). \tag{11}
$$

Also, we know that

$$
\nabla \mathcal{L}_\lambda^*(x) = \nabla_x f(x, y_\lambda^*(x)) + \lambda (\nabla_x g(x, y_\lambda^*(x)) - \nabla_x g(x, y^*(x))).
$$

By simple calculus, we have

$$
\begin{aligned}
&\nabla \mathcal{L}_\lambda^*(x) - \nabla \varphi(x) \\
&= \nabla_x f(x, y_\lambda^*(x)) - \nabla_x f(x, y^*(x)) \\
&\quad + \nabla_{xy}^2 g(x, y^*(x)) [\nabla_{yy} g(x, y^*(x))]^{-1} (\nabla_y f(x, y^*(x)) - \nabla_y f(x, y_\lambda^*(x)) \\
&\quad + \lambda \nabla_{xy}^2 g(x, y^*(x)) [\nabla_{yy} g(x, y^*(x))]^{-1} \times \\
&\qquad\qquad (\nabla_{yy}^2 g(x, y^*(x)) (y_\lambda^*(x) - y^*(x)) + \nabla_y g(x, y^*(x)) - \nabla_y g(x, y_\lambda^*(x))) \\
&\quad + \lambda (\nabla_x g(x, y_\lambda^*(x)) - \nabla_x g(x, y^*(x)) - \nabla_{xy}^2 g(x, y^*(x)) (y_\lambda^*(x) - y^*(x))).
\end{aligned}
$$

Taking norm on both sides,

$$
\begin{aligned}
\|\nabla \mathcal{L}_\lambda^*(x) - \nabla \varphi(x)\| &\leq L_f \|y_\lambda^*(x) - y^*(x)\| + \frac{\rho_g L_g}{\mu} \|y_\lambda^*(x) - y^*(x)\| \\
&\quad + \frac{\lambda L_g \rho_g}{2\mu} \|y_\lambda^*(x) - y^*(x)\|^2 + \frac{\lambda \rho_g}{2} \|y_\lambda^*(x) - y^*(x)\|^2 \\
&\leq \left(L_f + \frac{\rho_g L_g}{\mu} + \frac{C_f L_g \rho_g}{2\mu^2} + \frac{C_f \rho_g}{2\mu} \right) \|y_\lambda^*(x) - y^*(x)\| \\
&\leq \left(L_f + \frac{\rho_g L_g}{\mu} + \frac{C_f L_g \rho_g}{2\mu^2} + \frac{C_f \rho_g}{2\mu} \right) \frac{C_f}{\lambda \mu}
\end{aligned}
$$

□

Lemma B.5. *Under Assumption 3.1, for $\lambda \geq 2L_f/\mu$, it holds that $\|\nabla y^*(x) - \nabla y_\lambda^*(x)\| \leq D_2/\lambda$, where*

$$
D_2 := \left(\frac{1}{\mu} + \frac{2L_g}{\mu^2} \right) \left(L_f + \frac{C_f \rho_g}{\mu} \right) = \mathcal{O}(\kappa^3).
$$

Proof. Taking derivative on both sides of $\nabla_y g(x, y^*(x)) = 0$ yields

$$\nabla^2_{xy} g(x, y^*(x)) + \nabla y^*(x) \nabla^2_{yy} g(x, y^*(x)) = 0. \tag{12}$$

Rearranging, we can obtain

$$\nabla y^*(x) = -\nabla^2_{xy} g(x, y^*(x)) [\nabla^2_{yy} g(x, y^*(x))]^{-1}. \tag{13}$$

Similarly, we also have

$$\nabla y^*_\lambda(x) = -\nabla^2_{xy} \mathcal{L}_\lambda(x, y^*_\lambda(x)) [\nabla^2_{yy} \mathcal{L}_\lambda(x, y^*_\lambda(x))]^{-1}. \tag{14}$$

Using the matrix identity $A^{-1} - B^{-1} = A^{-1}(B - A)B^{-1}$, we have

$$\begin{aligned}
&\left\| [\nabla^2_{yy} g(x, y^*(x))]^{-1} - \left[\frac{\nabla^2_{yy} \mathcal{L}_\lambda(x, y^*_\lambda(x))}{\lambda} \right]^{-1} \right\| \\
&\leq \left\| [\nabla^2_{yy} g(x, y^*_\lambda(x))]^{-1} \right\| \left\| \frac{\nabla^2_{yy} \mathcal{L}_\lambda(x, y^*_\lambda(x))}{\lambda} - \nabla^2_{yy} g(x, y^*(x)) \right\| \left\| \left[\frac{\nabla^2_{yy} \mathcal{L}_\lambda(x, y^*_\lambda(x))}{\lambda} \right]^{-1} \right\| \\
&\leq \frac{2}{\mu^2} \left\| \frac{\nabla^2_{yy} f(x, y^*_\lambda(x))}{\lambda} + \nabla^2_{yy} g(x, y^*_\lambda(x)) - \nabla^2_{yy} g(x, y^*(x)) \right\| \\
&\leq \frac{2}{\mu^2} \left(\frac{L_f}{\lambda} + \rho_g \| y^*_\lambda(x) - y^*(x) \| \right) \\
&\leq \frac{2}{\lambda \mu^2} \left(L_f + \frac{C_f \rho_g}{\mu} \right).
\end{aligned} \tag{15}$$

Note that the setting of $\lambda \geq 2L_f/\mu$ implies $\|\nabla^2_{xy} \mathcal{L}(\,\cdot\,,\,\cdot\,)\| \leq 2\lambda L_g$, then we further have

$$\begin{aligned}
&\|\nabla y^*_\lambda(x) - \nabla y^*(x)\| \\
&\leq \left\| \nabla^2_{xy} g(x, y^*(x)) - \frac{\nabla^2_{xy} \mathcal{L}_\lambda(x, y^*_\lambda(x))}{\lambda} \right\| \left\| [\nabla^2_{yy} g(x, y^*(x))]^{-1} \right\| \\
&\quad + \left\| \frac{\nabla^2_{xy} \mathcal{L}_\lambda(x, y^*_\lambda(x))}{\lambda} \right\| \left\| [\nabla^2_{yy} g(x, y^*(x))]^{-1} - \left[\frac{\nabla^2_{yy} \mathcal{L}_\lambda(x, y^*_\lambda(x))}{\lambda} \right]^{-1} \right\| \\
&\leq \frac{1}{\mu} \left\| \nabla^2_{xy} g(x, y^*(x)) - \nabla^2_{xy} g(x, y^*_\lambda(x)) - \frac{\nabla^2_{xy} f(x, y^*_\lambda(x))}{\lambda} \right\| + \frac{2L_g}{\lambda \mu^2} \left(L_f + \frac{C_f \rho_g}{\mu} \right) \\
&\leq \left(\frac{1}{\lambda \mu} + \frac{2L_g}{\lambda \mu^2} \right) \left(L_f + \frac{C_f \rho_g}{\mu} \right).
\end{aligned}$$

□

It is clear that $\|\nabla y^*(x)\| \leq L_g/\mu$, therefore $y^*(x)$ is (L_g/μ)-Lipschitz. Below, we show a similar result also holds for $y^*_\lambda(x)$.

Lemma B.6. *Under Assumption 3.1, for $\lambda \geq 2L_f/\mu$, it holds that $\|\nabla y^*_\lambda(x)\| \leq 4L_g/\mu$.*

Proof. Recall Equation 14 that

$$\nabla y^*_\lambda(x) = -\nabla^2_{xy} \mathcal{L}_\lambda(x, y^*_\lambda(x)) [\nabla^2_{yy} \mathcal{L}_\lambda(x, y^*_\lambda(x))]^{-1}.$$

We arrive at the conclusion by noting that $\nabla^2_{xy} \mathcal{L}_\lambda(\,\cdot\,,\,\cdot\,) \preceq 2\lambda L_g$ and $\nabla^2_{yy} \mathcal{L}_\lambda(\,\cdot\,,\,\cdot\,) \succeq \lambda\mu/2$ by Lemma B.1.

□

This implies that $y_\lambda^*(x)$ is $(4L_g/\mu)$-Lipschitz.

Lemma B.7. *Under Assumption 3.1, for $\lambda \geq 2L_f/\mu$, $\nabla \mathcal{L}_\lambda^*(x)$ is D_3-Lipschitz, where*

$$D_3 := L_f + \frac{4L_f L_g}{\mu} + \frac{C_f \rho_g}{\mu} + \frac{C_f L_g \rho_g}{\mu^2} + L_g D_2 = \mathcal{O}(\kappa^3).$$

Proof. Note that

$$\nabla \mathcal{L}_\lambda^*(x) = \underbrace{\nabla_x f(x, y_\lambda^*(x))}_{A(x)} + \underbrace{\lambda(\nabla_x g(x, y_\lambda^*(x)) - \nabla_x g(x, y^*(x)))}_{B(x)}. \tag{16}$$

where $A(x)$ and $B(x)$ are both mappings $\mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$. From Lemma B.6 we know that $y_\lambda^*(x)$ is $(4L_g/\mu)$-Lipschitz. This further implies that $A(x)$ is $(1+4L_g/\mu)L_f$-Lipschitz. Next, we bound the Lipschitz coefficient of $B(x)$ via its derivative $\nabla B(x) : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x} \times \mathbb{R}^{d_x}$, which has the following form by taking total derivative on $B(x)$:

$$\begin{aligned} \nabla B(x) = {} & \lambda(\nabla_{xx}^2 g(x, y_\lambda^*(x)) - \nabla_{xx}^2 g(x, y^*(x))) \\ & + \lambda \left(\nabla y_\lambda^*(x) \nabla_{yx}^2 g(x, y_\lambda^*(x)) - \nabla y^*(x) \nabla_{yx}^2 g(x, y^*(x)) \right). \end{aligned} \tag{17}$$

And we can bound the operator norm of $\nabla B(x)$ by:

$$\begin{aligned} \|\nabla B(x)\| &\leq \lambda \|\nabla_{xx}^2 g(x, y_\lambda^*(x)) - \nabla_{xx}^2 g(x, y^*(x))\| \\ &\quad + \lambda \|\nabla y^*(x)\| \|\nabla_{yx}^2 g(x, y_\lambda^*(x)) - \nabla_{yx}^2 g(x, y^*(x))\| \\ &\quad + \lambda \|\nabla y_\lambda^*(x) - \nabla y^*(x)\| \|\nabla_{yx}^2 g(x, y_\lambda^*(x))\| \\ &\leq \lambda \rho_g \left(1 + \frac{L_g}{\mu}\right) \|y_\lambda^*(x) - y^*(x)\| + \lambda L_g \|\nabla y_\lambda^*(x) - \nabla y^*(x)\| \\ &\leq \left(1 + \frac{L_g}{\mu}\right) \frac{C_f \rho_g}{\mu} + L_g D_2, \end{aligned}$$

where we use Lemma B.6 in the second inequality; Lemma B.2 and B.5 in the third one. □

C Lemmas for Finding Second-Order Stationarity

Lemma C.1. *Under Assumption 3.1 and 3.2, for $\lambda \geq 2L_f/\mu$, we have $\|\nabla^2 y^*(x) - \nabla^2 y_\lambda^*(x)\| \leq D_4/\lambda$, where*

$$D_4 := \frac{2\rho_g}{\lambda \mu^2} \left(1 + \frac{L_g}{\mu}\right)^2 \left(L_f + \frac{C_f \rho_g}{\mu}\right) + \frac{14 L_g \rho_g D_2}{\lambda \mu^2} + \frac{50 L_g^2}{\lambda \mu^3} \left(\frac{C_f \nu_g}{\mu} + \rho_f\right) = \mathcal{O}(\kappa^5).$$

Proof. By taking the derivative with respect to x on

$$\nabla_{xy}^2 g(x, y^*(x)) + \nabla y^*(x) \nabla_{yy}^2 g(x, y^*(x)) = 0,$$

we obtain

$$\begin{aligned} &\nabla_{xxy}^3 g(x, y^*(x)) + \nabla_{yxy}^3 g(x, y^*(x)) \times_1 \nabla y^*(x) + \nabla^2 y^*(x) \times_3 \nabla_{yy}^2 g(x, y^*(x)) \\ &\quad + \nabla_{xyy}^3 g(x, y^*(x)) \times_2 \nabla y^*(x) + \nabla_{yyy}^3 g(x, y^*(x)) \times_1 \nabla y^*(x) \times_2 \nabla y^*(x) = 0. \end{aligned}$$

Rearranging to get

$$\begin{aligned} \nabla^2 y^*(x) = {} & - \left(\nabla_{xxy}^3 g(x, y^*(x)) + \nabla_{yxy}^3 g(x, y^*(x)) \times_1 \nabla y^*(x)\right) \times_3 [\nabla_{yy}^2 g(x, y^*(x))]^{-1} \\ & - \nabla_{xyy}^3 g(x, y^*(x)) \times_2 \nabla y^*(x) \times_3 [\nabla_{yy}^2 g(x, y^*(x))]^{-1} \\ & - \nabla_{yyy}^3 g(x, y^*(x)) \times_1 \nabla y^*(x) \times_2 \nabla y^*(x) \times_3 [\nabla_{yy}^2 g(x, y^*(x))]^{-1}. \end{aligned} \tag{18}$$

Similarly,

$$\begin{aligned}\nabla^2 y_\lambda^*(x) = &-\left(\nabla^3_{xxy}\mathcal{L}_\lambda(x,y_\lambda^*(x)) + \nabla^3_{yxy}\mathcal{L}_\lambda(x,y_\lambda^*(x)) \times_1 \nabla y_\lambda^*(x)\right) \times_3 [\nabla^2_{yy}\mathcal{L}_\lambda(x,y_\lambda^*(x))]^{-1} \\ &- \nabla^3_{xyy}\mathcal{L}_\lambda(x,y_\lambda^*(x)) \times_2 \times_3 [\nabla^2_{yy}\mathcal{L}_\lambda(x,y_\lambda^*(x))]^{-1} \\ &- \nabla y_\lambda^*(x) + \nabla^3_{yyy}\mathcal{L}_\lambda(x,y_\lambda^*(x)) \times_1 \nabla y_\lambda^*(x) \times_2 \nabla y_\lambda^*(x) \times_3 [\nabla^2_{yy}\mathcal{L}_\lambda(x,y_\lambda^*(x))]^{-1}.\end{aligned} \tag{19}$$

Note that

$$\left\|\nabla^3_{xxy} g(x,y^*(x)) - \frac{\nabla^3_{xxy}\mathcal{L}_\lambda(x,y_\lambda^*(x))}{\lambda}\right\| \le \nu_g \|y_\lambda^*(x) - y^*(x)\| + \frac{\rho_f}{\lambda} \le \frac{1}{\lambda}\left(\frac{C_f\nu_g}{\mu} + \rho_f\right),$$

and

$$\begin{aligned}&\left\|\nabla^3_{yxy} g(x,y^*(x)) \times_1 \nabla y^*(x) - \frac{\nabla^3_{yxy}\mathcal{L}_\lambda(x,y_\lambda^*(x)) \times_1 \nabla y_\lambda^*(x)}{\lambda}\right\| \\ &\le \|\nabla y^*(x) - \nabla y_\lambda^*(x)\| \|\nabla^3_{yxy} g(x,y^*(x))\| + \|\nabla y_\lambda^*(x)\| \left\|\nabla^3_{yxy} g(x,y^*(x)) - \frac{\nabla^3_{yxy}\mathcal{L}_\lambda(x,y_\lambda^*(x))}{\lambda}\right\| \\ &\le \frac{\rho_g D_2}{\lambda} + \frac{4L_g}{\lambda\mu}\left(\frac{C_f\nu_g}{\mu} + \rho_f\right),\end{aligned}$$

and

$$\begin{aligned}&\left\|\nabla^3_{yyy} g(x,y^*(x)) \times_1 \nabla y^*(x) \times_2 \nabla y^*(x) - \frac{\nabla^3_{yyy}\mathcal{L}_\lambda(x,y_\lambda^*(x)) \times_1 \nabla y_\lambda^*(x) \times_2 \nabla y_\lambda^*(x)}{\lambda}\right\| \\ &\le \|\nabla y^*(x)\| \|\nabla^3_{yyy} g(x,y^*(x))\| \|\nabla y^*(x) - \nabla y_\lambda^*(x)\| \\ &\quad + \|\nabla y_\lambda^*(x)\| \|\nabla^3_{yyy} g(x,y^*(x))\| \|\nabla y^*(x) - \nabla y_\lambda^*(x)\| \\ &\quad + \|\nabla y_\lambda^*(x)\|^2 \left\|\nabla^3_{xxy} g(x,y^*(x)) - \frac{\nabla^3_{xxy}\mathcal{L}_\lambda(x,y_\lambda^*(x))}{\lambda}\right\| \\ &\le \frac{5L_g\rho_g D_2}{\lambda\mu} + \frac{16L_g^2}{\lambda\mu^2}\left(\frac{C_f\nu_g}{\mu} + \rho_f\right),\end{aligned}$$

we can obtain that

$$\begin{aligned}&\|\nabla^2 y^*(x) - \nabla^2 y_\lambda^*(x)\| \\ &\le \rho_g\left(1 + \frac{L_g}{\mu}\right)^2 \left\|[\nabla^2_{yy} g(x,y^*(x))]^{-1} - \left[\frac{\nabla^2_{yy}\mathcal{L}_\lambda(x,y_\lambda^*(x))}{\lambda}\right]^{-1}\right\| \\ &\quad + \left(\frac{7L_g\rho_g D_2}{\lambda\mu} + \frac{25L_g^2}{\lambda\mu^2}\left(\frac{C_f\nu_g}{\mu} + \rho_f\right)\right) \left\|\left[\frac{\nabla^2_{yy}\mathcal{L}_\lambda(x,y_\lambda^*(x))}{\lambda}\right]^{-1}\right\| \\ &\le \frac{2\rho_g}{\lambda\mu^2}\left(1 + \frac{L_g}{\mu}\right)^2\left(L_f + \frac{C_f\rho_g}{\mu}\right) + \frac{14L_g\rho_g D_2}{\lambda\mu^2} + \frac{50L_g^2}{\lambda\mu^3}\left(\frac{C_f\nu_g}{\mu} + \rho_f\right),\end{aligned}$$

where we use Equation 15 in the second inequality. □

Lemma C.2. *Under Assumption 3.1 and 3.2, for $\lambda \ge 2L_f/\mu$, $\nabla^2\mathcal{L}_\lambda^*(x)$ is D_5-Lipschitz, where*

$$\begin{aligned}D_5 := &\left(1 + \frac{4L_g}{\mu}\right)^2\left(3\rho_f + \frac{2L_f\rho_g}{\mu}\right) + \left(1 + \frac{L_g}{\mu}\right)^2\frac{C_f\nu_g}{\mu} \\ &+ \left(2 + \frac{5L_g}{\mu}\right)D_2\rho_g + \left(1 + \frac{L_g}{\mu}\right)^2\frac{C_f\rho_g^2}{\mu^2} + L_g D_4 = \mathcal{O}(\ell\kappa^5).\end{aligned}$$

Proof. Similar to the proof of Lemma B.7, we split $\nabla^2\mathcal{L}_\lambda^*(x)$ into two terms:

$$\nabla^2\mathcal{L}_\lambda^*(x) = \nabla A(x) + \nabla B(x),$$

where both the mappings $A(x)$ and $B(x)$ both follow the definitions in Equation 16. Taking total derivative on $A(x)$, we obtain

$$\nabla A(x) = \nabla_{xx}^2 f(x, y_\lambda^*(x)) + \nabla y_\lambda^*(x)\nabla_{yx}^2 f(x, y_\lambda^*(x)).$$

And recall Equation 17 that

$$\begin{aligned}\nabla B(x) = \lambda(&\nabla_{xx}^2 g(x, y_\lambda^*(x)) - \nabla_{xx}^2 g(x, y^*(x))) \\ &+ \lambda\left(\nabla y_\lambda^*(x)\nabla_{yx}^2 g(x, y_\lambda^*(x)) - \nabla y^*(x)\nabla_{yx}^2 g(x, y^*(x))\right).\end{aligned}$$

Then we bound the Lipschitz coefficient of $\nabla A(x)$ and $\nabla B(x)$, respectively.

From Equation 18 we can calculate that

$$\|\nabla^2 y^*(x)\| \le \left(1 + \frac{L_g}{\mu}\right)^2 \frac{\rho_g}{\mu}. \tag{20}$$

Similarly, from Equation 19 we can calculate that

$$\|\nabla^2 y_\lambda^*(x)\| \le \left(1 + \frac{4L_g}{\mu}\right)^2 \left(\frac{\rho_f}{\lambda} + \rho_g\right)\frac{2}{\mu} \le \left(1 + \frac{4L_g}{\mu}\right)^2 \left(\frac{2\rho_f}{L_f} + \frac{2\rho_g}{\mu}\right). \tag{21}$$

From Lemma B.6 we know that $y_\lambda^*(x)$ is $(4L_g/\mu)$-Lipschitz. This further implies that both $\nabla_{xx}^2 f(x, y_\lambda^*(x))$ and $\nabla_{yx}^2 f(x, y_\lambda^*(x))$ are $(1 + 4L_g/\mu)\rho_f$-Lipschitz. Then, for any $x_1, x_2 \in \mathbb{R}^{d_x}$, we have

$$\begin{aligned}&\|\nabla A(x_1) - \nabla A(x_2)\| \\ \le\ &\|\nabla_{xx}^2 f(x_1, y_\lambda^*(x_1)) - \nabla_{xx}^2 f(x_2, y_\lambda^*(x_2))\| \\ &+ \|\nabla y_\lambda^*(x_1)\nabla_{yx}^2 f(x_1, y_\lambda^*(x_1)) - \nabla y_\lambda^*(x_2)\nabla_{yx}^2 f(x_1, y_\lambda^*(x_1))\| \\ &+ \|\nabla y_\lambda^*(x_2)\nabla_{yx}^2 f(x_1, y_\lambda^*(x_1)) - \nabla y_\lambda^*(x_2)\nabla_{yx}^2 f(x_2, y_\lambda^*(x_2))\| \\ \le\ &\|\nabla_{xx}^2 f(x_1, y_\lambda^*(x_1)) - \nabla_{xx}^2 f(x_2, y_\lambda^*(x_2))\| \\ &+ \|\nabla y_\lambda^*(x_1) - \nabla y_\lambda^*(x_2)\|\|\nabla_{yx}^2 f(x_1, y_\lambda^*(x_1))\| \\ &+ \|\nabla y_\lambda^*(x_2)\|\|\nabla_{yx}^2 f(x_1, y_\lambda^*(x_1)) - \nabla_{yx}^2 f(x_2, y_\lambda^*(x_2))\| \\ \le\ &\underbrace{\left(\left(1 + \frac{4L_g}{\mu}\right)^2 \rho_f + \left(1 + \frac{4L_g}{\mu}\right)^2 \left(\frac{2\rho_f}{L_f} + \frac{2\rho_g}{\mu}\right) L_f\right)}_{C_1} \|x_1 - x_2\|,\end{aligned}$$

where C_1 gives the upper bound of the Lipschitz coefficient of the mapping $\nabla A(x)$.

To bound the Lipschitz coefficient of $\nabla B(x)$, we first derive the explicit form of the mapping $\nabla^2 B(x) : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x} \times \mathbb{R}^{d_x} \times \mathbb{R}^{d_x}$ by:

$$\begin{aligned}\nabla^2 B(x) = &\nabla^2 y_\lambda^*(x) \times_3 [\nabla_{yx}^2 f(x, y_\lambda^*(x))]^\top \\ &+ \lambda(\nabla_{xxx}^3 g(x, y_\lambda^*(x)) - \nabla_{xxx}^3 g(x, y^*(x))) \\ &+ \lambda(\nabla_{yxx}^3 g(x, y_\lambda^*(x)) \times_1 \nabla y_\lambda^*(x) - \nabla_{yxx}^3 g(x, y^*(x)) \times_1 \nabla y^*(x)) \\ &+ \lambda\left(\nabla_{xyx}^3 g(x, y_\lambda^*(x)) \times_2 \nabla y_\lambda^*(x) - \nabla_{xyx}^3 g(x, y^*(x)) \times_2 \nabla y^*(x)\right) \\ &+ \lambda\left(\nabla_{yyx}^3 g(x, y_\lambda^*(x)) \times_1 \nabla y_\lambda^*(x) \times_2 \nabla y_\lambda^*(x) - \nabla_{yyx}^3 g(x, y^*(x)) \times_1 \nabla y^*(x) \times_2 \nabla y^*(x)\right) \\ &+ \lambda\left(\nabla^2 y_\lambda^*(x) \times_3 [\nabla_{yx}^2 g(x, y_\lambda^*(x))]^\top - \nabla^2 y^*(x) \times_3 [\nabla_{yx}^2 g(x, y^*(x))]^\top\right).\end{aligned}$$

Then we can bound the Lipschitz coefficient of $\nabla B(x)$ via the operator norm of $\nabla^2 B(x)$:

$$C_2 := \|\nabla^2 B(x)\|$$

$$
\begin{aligned}
&\leq \|\nabla^3_{xxx} g(x, y^*(x)) - \nabla^3_{xxx} g(x, y^*_\lambda(x))\| \\
&\quad + \lambda \|\nabla y^*(x)\| \|\nabla^3_{yxx} g(x, y^*(x)) - \nabla^3_{yxx} g(x, y^*_\lambda(x))\| \\
&\quad + \lambda \|\nabla y^*_\lambda(x) - \nabla y^*(x)\| \|\nabla^3_{yxx} g(x, y^*_\lambda(x))\| \\
&\quad + \lambda \|\nabla y^*(x)\| \|\nabla^3_{xyx} g(x, y^*(x)) - \nabla^3_{xyx} g(x, y^*_\lambda(x))\| \\
&\quad + \lambda \|\nabla y^*(x) - \nabla y^*_\lambda(x)\| \|\nabla^3_{xyx} g(x, y^*_\lambda(x))\| \\
&\quad + \lambda \|\nabla y^*(x)\| \|\nabla^3_{yyx} g(x, y^*(x))\| \|\nabla y^*_\lambda(x) - \nabla y^*(x)\| \\
&\quad + \lambda \|\nabla y^*_\lambda(x)\| \|\nabla^3_{yyx} g(x, y^*(x))\| \|\nabla y^*_\lambda(x) - \nabla y^*(x)\| \\
&\quad + \lambda \|\nabla y^*(x)\|^2 \|\nabla^3_{yyx} g(x, y^*(x)) - \nabla^3_{yyx} g(x, y^*_\lambda(x))\| \\
&\quad + \lambda \|\nabla^2 y^*(x)\| \|\nabla^2_{yx} g(x, y^*(x)) - \nabla^2_{yx} g(x, y^*_\lambda(x))\| \\
&\quad + \lambda \|\nabla^2 y^*(x) - \nabla^2 y^*_\lambda(x)\| \|\nabla^2_{yx} g(x, y^*_\lambda(x))\|.
\end{aligned}
$$

Then we can bound C_2 using the upper bound for $\|\nabla y^*(x)\|$ given by Equation 13, that for $\|\nabla y^*_\lambda(x)\|$ by Lemma B.6, that for $\|\nabla y^*(x) - \nabla y^*_\lambda(x)\|$ by Lemma B.5, that for $\|\nabla^2 y^*(x)\|$ by Equation 20 and that for $\|\nabla^2 y^*(x) - \nabla^2 y^*_\lambda(x)\|$ by Equation C.1. Finally, we can obtain

$$
\begin{aligned}
C_1 + C_2 \leq &\left(1 + \frac{4L_g}{\mu}\right)^2 \left(3\rho_f + \frac{2L_f \rho_g}{\mu}\right) + \left(1 + \frac{L_g}{\mu}\right)^2 \frac{C_f \nu_g}{\mu} \\
&+ \left(2 + \frac{5L_g}{\mu}\right) D_2 \rho_g + \left(1 + \frac{L_g}{\mu}\right)^2 \frac{C_f \rho_g^2}{\mu^2} + L_g D_4.
\end{aligned}
$$

□

Lemma C.3. *Under Assumption 3.1 and 3.2, for $\lambda \geq 2L_f/\mu$, we have $\|\nabla^2 \mathcal{L}^*_\lambda(x) - \nabla^2 \varphi(x)\| \leq D_6/\lambda$, where*

$$
D_6 := 2L_g D_2^2 + \left(1 + \frac{L_g}{\mu}\right)^2 \left(\frac{C_f \rho_f}{\mu} + \frac{C_f L_f \rho_g}{\mu^2} + \frac{C_f^2 \nu_g}{2\mu^2} + \frac{C_f^2 \rho_g^2}{2\mu^3}\right) = \mathcal{O}(\ell \kappa^6).
$$

Proof. Taking total derivative on $\nabla \varphi(x) = \nabla_x f(x, y^*(x)) + \nabla y^*(x) \nabla_y f(x, y^*(x))$ yields

$$
\begin{aligned}
\nabla^2 \varphi(x) = &\nabla^2_{xx} f(x, y^*(x)) + \nabla y^*(x) \nabla^2_{yx} f(x, y^*(x)) + \nabla^2 y^*(x) \bar{\times}_3 \nabla_y f(x, y^*(x)) \\
&+ \nabla^2_{xy} f(x, y^*(x)) [\nabla y^*(x)]^\top + \nabla y^*(x) \nabla^2_{yy} f(x, y^*(x)) [\nabla y^*(x)]^\top.
\end{aligned} \tag{22}
$$

Plug into the close form of $\nabla^2 y^*(x)$ given by Equation 18, we arrive at

$$
\begin{aligned}
&\nabla^2 \varphi(x) \\
&= \underbrace{\nabla^2_{xx} f(x, y^*(x)) - \nabla^3_{xxy} g(x, y^*(x)) \times_3 [\nabla^2_{yy} g(x, y^*(x))]^{-1} \bar{\times}_3 \nabla_y f(x, y^*(x))}_{\text{(I)}} \\
&\quad + \underbrace{\left(\nabla^2_{yx} f(x, y^*(x)) - \nabla^3_{xyy} g(x, y^*(x)) \times_3 [\nabla^2_{yy} g(x, y^*(x)]^{-1} \bar{\times}_3 \nabla_y f(x, y^*(x))\right) \times_1 \nabla y^*(x)}_{\text{(II)}} \\
&\quad + \underbrace{\left(\nabla^2_{xy} f(x, y^*(x)) - \nabla^3_{xyy} g(x, y^*(x)) \times_3 [\nabla^2_{yy} g(x, y^*(x)]^{-1} \bar{\times}_3 \nabla_y f(x, y^*(x))\right) \times_2 \nabla y^*(x)}_{\text{(III)}} \\
&\quad + \underbrace{\left(\nabla^2_{yy} f(x, y^*(x)) - \nabla^3_{yyy} g(x, y^*(x)) \times_3 [\nabla^2_{yy} g(x, y^*(x)]^{-1} \bar{\times}_3 \nabla_y f(x, y^*(x))\right) \times_1 \nabla y^*(x) \times_2 \nabla y^*(x)}_{\text{(IV)}}.
\end{aligned}
$$

Recall that

$$
\begin{aligned}
\nabla^2 \mathcal{L}^*_\lambda(x) = &\nabla^2_{xx} f(x, y^*_\lambda(x)) + \lambda(\nabla^2_{xx} g(x, y^*_\lambda(x)) - \nabla^2_{xx} g(x, y^*(x))) \\
&+ \nabla y^*_\lambda(x) \nabla^2_{yx} \mathcal{L}_\lambda(x, y^*_\lambda(x)) - \lambda \nabla y^*(x) \nabla^2_{yx} g(x, y^*(x)).
\end{aligned}
$$

Let

$$\begin{aligned}\tilde{\nabla}^2\mathcal{L}_\lambda^*(x) = &\underbrace{\nabla_{xx}^2 f(x, y_\lambda^*(x)) + \lambda(\nabla_{xx}^2 g(x, y_\lambda^*(x)) - \nabla_{xx}^2 g(x, y^*(x)))}_{\text{(I')}} \\ &+ \underbrace{\nabla y^*(x)\left(\nabla_{yx}^2\mathcal{L}_\lambda(x, y_\lambda^*(x)) - \lambda\nabla_{yx}^2 g(x, y^*(x))\right)}_{\text{(II')}} \\ &+ \underbrace{\left(\nabla_{xy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x)) - \lambda\nabla_{xy}^2 g(x, y^*(x))\right)[\nabla y^*(x)]^\top}_{\text{(III')}} \\ &+ \underbrace{\nabla y^*(x)\left(\nabla_{yy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x)) - \lambda\nabla_{yy}^2 g(x, y^*(x))\right)[\nabla y^*(x)]^\top}_{\text{(IV')}}.\end{aligned}$$

Recall Equation 14 that

$$\nabla y_\lambda^*(x) = -\nabla_{xy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x))[\nabla_{yy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x))]^{-1}.$$

Then using equivalent transformation and the identity

$$UAU^\top - VAV^\top = (U-V)AV^\top + VA(U-V)^\top + (U-V)A(U-V)^\top$$

for a matrices U, V and A, we can show that

$$\begin{aligned}&\tilde{\nabla}^2\mathcal{L}_\lambda^*(x) - \nabla^2\mathcal{L}_\lambda^*(x) \\ =\ &(\nabla y^*(x) - \nabla y_\lambda^*(x))\nabla_{yx}^2\mathcal{L}_\lambda(x, y_\lambda^*(x)) + \\ &+ \nabla_{xy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x))[\nabla y^*(x)]^\top + \nabla y^*(x)\nabla_{yy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x))[\nabla y^*(x)]^\top \\ =\ &(\nabla y^*(x) - \nabla y_\lambda^*(x))\nabla_{yx}^2\mathcal{L}_\lambda(x, y_\lambda^*(x)) + \nabla_{xy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x))[\nabla y^*(x) - \nabla y_\lambda^*(x)]^\top \\ &+ \nabla y^*(x)\nabla_{yy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x))[\nabla y^*(x)]^\top - \nabla y_\lambda^*(x)\nabla_{yy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x))[\nabla y_\lambda^*(x)]^\top \\ =\ &(\nabla y^*(x) - \nabla y_\lambda^*(x))\nabla_{yx}^2\mathcal{L}_\lambda(x, y_\lambda^*(x)) + \nabla_{xy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x))[\nabla y^*(x) - \nabla y_\lambda^*(x)]^\top \\ &+ (\nabla y^*(x) - \nabla y_\lambda^*(x))\nabla_{yy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x))[\nabla y_\lambda^*(x)]^\top \\ &+ \nabla y_\lambda^*(x)\nabla_{yy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x))(\nabla y^*(x) - \nabla y_\lambda^*(x))^\top \\ &+ (\nabla y^*(x) - \nabla y_\lambda^*(x))\nabla_{yy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x))[\nabla y^*(x) - \nabla y_\lambda^*(x)]^\top \\ =\ &(\nabla y^*(x) - \nabla y_\lambda^*(x))\nabla_{yy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x))[\nabla y^*(x) - \nabla y_\lambda^*(x)]^\top.\end{aligned}$$

Noting that $\nabla_{yy}^2\mathcal{L}_\lambda(\,\cdot\,,\,\cdot\,) \preceq 2\lambda L_g$, we have

$$\|\tilde{\nabla}^2\mathcal{L}_\lambda^*(x) - \nabla^2\mathcal{L}_\lambda^*(x)\| \le 2\lambda L_g\|\nabla y^*(x) - \nabla y_\lambda^*(x)\|^2 \le \frac{2L_g D_2^2}{\lambda}.$$

Next, we bound difference between $\tilde{\nabla}^2\mathcal{L}_\lambda^*(x)$ and $\nabla^2\mathcal{L}_\lambda^*(x)$. For the difference between (I) and (I′),

$$\begin{aligned}&\text{(I)} - \text{(I}') \\ =\ &\nabla_{xx}^2 f(x, y^*(x)) - \nabla_{xx}^2 f(x, y_\lambda^*(x)) \\ &+ \nabla_{xxy}^3 g(x, y^*(x)) \times_3 [\nabla_{yy}^2 g(x, y^*(x))]^{-1} \bar{\times}_3 (\nabla_y f(x, y_\lambda^*(x)) - \nabla_y f(x, y^*(x))) \\ &+ \lambda\left(\nabla_{xx}^2 g(x, y^*(x)) - \nabla_{xx}^2 g(x, y_\lambda^*(x)) + \nabla_{xxy}^3 g(x, y^*(x)) \bar{\times}_3 (y_\lambda^*(x) - y^*(x))\right) \\ &+ \lambda\nabla_{xxy}^3 g(x, y^*(x)) \times_3 [\nabla_{yy}^2 g(x, y^*(x))]^{-1} \\ &\qquad \bar{\times}_3 \left(\nabla_y g(x, y_\lambda^*(x)) - \nabla_y g(x, y^*(x)) - \nabla_{yy}^2 g(x, y^*(x))(y_\lambda^*(x) - y^*(x))\right).\end{aligned}$$

Therefore,

$$\|(\text{I}) - (\text{I}')\| \le \rho_f\|y^*(x) - y_\lambda^*(x)\| + \frac{L_f\rho_g}{\mu}\|y^*(x) - y_\lambda^*(x)\|$$

$$+\frac{\lambda\nu_g}{2}\|y^*(x)-y_\lambda^*(x)\|^2+\frac{\lambda\rho_g^2}{2\mu}\|y^*(x)-y_\lambda^*(x)\|^2$$
$$\le \frac{C_f\rho_f}{\lambda\mu}+\frac{C_fL_f\rho_g}{\lambda\mu^2}+\frac{C_f^2\nu_g}{2\lambda\mu^2}+\frac{C_f^2\rho_g^2}{2\lambda\mu^3}.$$

Using $\|\nabla y^*(x)\|\le L_g/\mu$, we can similarly bound the difference between (II) and (II$'$) by

$$\begin{aligned}
&\|(\mathrm{II})-(\mathrm{II}')\|\\
&\le \|\nabla y^*(x)\|\Big\|\nabla^2_{yx}f(x,y^*(x))-\nabla^3_{yxy}g(x,y^*(x))\times_3[\nabla^2_{yy}g(x,y^*(x)]^{-1}\bar{\times}_3\nabla_y f(x,y^*(x))\\
&\quad-\nabla^2_{yx}f(x,y_\lambda^*(x))-\lambda\left(\nabla^2_{yx}g(x,y_\lambda^*(x))-\nabla^2_{yx}g(x,y^*(x))\right)\Big\|\\
&\le \frac{L_g}{\mu}\left(\frac{C_f\rho_f}{\lambda\mu}+\frac{C_fL_f\rho_g}{\lambda\mu^2}+\frac{C_f^2\nu_g}{2\lambda\mu^2}+\frac{C_f^2\rho_g^2}{2\lambda\mu^3}\right),
\end{aligned}$$

And the bound for (III) and (III$'$) is the same. Finally, we know that

$$\begin{aligned}
&\|(\mathrm{IV})-(\mathrm{IV}')\|\\
&\le \|\nabla y^*(x)\|^2\Big\|\nabla^2_{yy}f(x,y^*(x))-\nabla^3_{yyy}g(x,y^*(x))\times_3[\nabla^2_{yy}g(x,y^*(x))]^{-1}\bar{\times}_3\nabla_y f(x,y^*(x))\\
&\quad-\nabla^2_{yy}f(x,y_\lambda^*(x))-\lambda\left(\nabla^2_{yy}g(x,y_\lambda^*(x))-\nabla^2_{yy}g(x,y^*(x))\right)\Big\|\\
&\le \frac{L_g^2}{\mu^2}\left(\frac{C_f\rho_f}{\lambda\mu}+\frac{C_fL_f\rho_g}{\lambda\mu^2}+\frac{C_f^2\nu_g}{2\lambda\mu^2}+\frac{C_f^2\rho_g^2}{2\lambda\mu^3}\right).
\end{aligned}$$

Combing the above bounds completes our proof. □

D Proofs of Finding First-Order Stationarity

We recall the standard result for gradient descent for strongly convex functions, which is used for proving the linear convergence of $y_t^k\to y_\lambda^*(x_t)$ and $z_t^k\to y^*(x_t)$.

Theorem D.1 (Theorem 3.10 by Bubeck et al. [13]). *Suppose $h(x):\mathbb{R}^d\to\mathbb{R}$ is β-gradient Lipschitz and α-strongly convex. Consider the following update of gradient descent:*

$$x_{t+1}=x_t-\frac{1}{\beta}\nabla h(x_t).$$

Let $x^*=\arg\min_{x\in\mathbb{R}^d}h(x)$. Then it holds that

$$\|x_{t+1}-x^*\|^2\le\left(1-\frac{\alpha}{\beta}\right)\|x_t-x^*\|^2.$$

Below, we prove that F^{2}BA is indeed near-optimal.

Theorem 4.1. *Suppose Assumption 3.1 holds. Define $\Delta:=\varphi(x_0)-\inf_{x\in\mathbb{R}^{d_x}}\varphi(x)$ and $R:=\|y_0-y^*(x_0)\|^2$. Let $\eta\asymp\ell^{-1}\kappa^{-3}$, $\lambda\asymp\max\left\{\kappa/R,\ \ell\kappa^2/\Delta,\ \ell\kappa^3/\epsilon\right\}$ and set other parameters in Algorithm 1 as*

$$\alpha=\frac{1}{L_g},\ \tau=\frac{1}{2\lambda L_g},\ K=\mathcal{O}\left(\frac{L_g}{\mu}\log\left(\frac{\lambda L_g}{\mu}\right)\right),$$

then it can find an ϵ-first-order stationary point of $\varphi(x)$ within $T=\mathcal{O}(\ell\kappa^4\epsilon^{-2}\log(\ell\kappa/\epsilon))$ first-order oracle calls, where ℓ, κ are defined in Definition 3.1.

Proof. Let L be the gradient Lipschitz coefficient of $\mathcal{L}_\lambda^*(x)$. According to Lemma 4.1 and the setting of λ, we have

a. $\sup_{x\in\mathbb{R}^{d_x}} \|\nabla\mathcal{L}_\lambda^*(x) - \nabla\varphi(x)\| = \mathcal{O}(\epsilon)$.

b. $\mathcal{L}_\lambda^*(x_0) - \inf_{x\in\mathbb{R}^{d_x}} \mathcal{L}_\lambda^*(x) = \mathcal{O}(\Delta)$.

c. $L := \sup_{x\in\mathbb{R}^{d_x}} \|\nabla^2\mathcal{L}_\lambda^*(x)\| = \mathcal{O}(\ell\kappa^3)$.

Due to Lemma B.2, we also have

$$\|y_0 - y_\lambda^*(x_0)\|^2 + \|y_0 - y^*(x_0)\|^2 = \mathcal{O}(R).$$

Now it suffices to show that the algorithm can find an ϵ-first-order stationary of $\mathcal{L}_\lambda^*(x)$.

We begin from the following descent lemma for gradient descent. Let $\eta \le 1/(2L)$, then

$$\begin{aligned}
\mathcal{L}_\lambda^*(x_{t+1}) &\le \mathcal{L}_\lambda^*(x_t) + \langle\nabla\mathcal{L}_\lambda^*(x_t), x_{t+1} - x_t\rangle + \frac{L}{2}\|x_{t+1} - x_t\|^2 \\
&= \mathcal{L}_\lambda^*(x_t) - \frac{\eta}{2}\|\nabla\mathcal{L}_\lambda^*(x_t)\|^2 - \left(\frac{\eta}{2} - \frac{\eta^2 L}{2}\right)\|\hat{\nabla}\mathcal{L}_\lambda^*(x_t)\|^2 + \frac{\eta}{2}\|\hat{\nabla}\mathcal{L}_\lambda^*(x_t) - \nabla\mathcal{L}_\lambda^*(x_t)\|^2 \qquad (23)\\
&\le \mathcal{L}_\lambda^*(x_t) - \frac{\eta}{2}\|\nabla\mathcal{L}_\lambda^*(x_t)\|^2 - \frac{1}{4\eta}\|x_{t+1} - x_t\|^2 + \frac{\eta}{2}\|\hat{\nabla}\mathcal{L}_\lambda^*(x_t) - \nabla\mathcal{L}_\lambda^*(x_t)\|^2.
\end{aligned}$$

Note that

$$\|\hat{\nabla}\mathcal{L}_\lambda^*(x_t) - \nabla\mathcal{L}_\lambda^*(x_t)\| \le 2\lambda L_g\|y_t^K - y_\lambda^*(x_t)\| + \lambda L_g\|z_t^K - y^*(x_t)\|.$$

By Theorem D.1, we have

$$\begin{aligned}
\|y_t^K - y_\lambda^*(x_t)\|^2 &\le \exp\left(-\frac{\mu K}{4L_g}\right)\|y_t^0 - y_\lambda^*(x_t)\|^2 \\
\|z_t^K - y^*(x_t)\|^2 &\le \exp\left(-\frac{\mu K}{L_g}\right)\|z_t^0 - y^*(x_t)\|^2
\end{aligned}$$

Therefore, we have

$$\|\hat{\nabla}\mathcal{L}_\lambda^*(x_t) - \nabla\mathcal{L}_\lambda^*(x_t)\|^2 \le 4\lambda^2 L_g^2 \exp\left(-\frac{\mu K}{4L_g}\right)\left(\|y_t^0 - y_\lambda^*(x_t)\|^2 + \|z_t^0 - y^*(x_t)\|^2\right) \qquad (24)$$

By Young's inequality,

$$\begin{aligned}
\|y_{t+1}^0 - y_\lambda^*(x_{t+1})\|^2 &\le 2\|y_t^K - y_\lambda^*(x_t)\|^2 + 2\|y_\lambda^*(x_{t+1}) - y_\lambda^*(x_t)\|^2 \\
&\le 2\exp\left(-\frac{\mu K}{4L_g}\right)\|y_t^0 - y_\lambda^*(x_t)\|^2 + \frac{32L_g^2}{\mu^2}\|x_{t+1} - x_t\|^2,
\end{aligned}$$

where the second inequality follows Lemma B.6 that $y_\lambda^*(x)$ is $(4L_g/\mu)$-Lipschitz. Similarly, we can derive the recursion about $\|z_t^0 - y^*(x_t)\|^2$.

Put them together and let $K \ge 8L_g/\mu$, we have

$$\begin{aligned}
&\|y_{t+1}^0 - y_\lambda^*(x_{t+1})\|^2 + \|z_{t+1}^0 - y^*(x_{t+1})\|^2 \qquad (25)\\
&\le 2\exp\left(-\frac{\mu K}{4L_g}\right)\left(\|y_t^0 - y_\lambda^*(x_t)\|^2 + \|z_t^0 - y^*(x_t)\|^2\right) + \frac{34L_g^2}{\mu^2}\|x_{t+1} - x_t\|^2 \qquad (26)\\
&\le \frac{1}{2}\left(\|y_t^0 - y_\lambda^*(x_t)\|^2 + \|z_t^0 - y^*(x_t)\|^2\right) + \frac{34L_g^2}{\mu^2}\|x_{t+1} - x_t\|^2. \qquad (27)
\end{aligned}$$

Telescoping over t yields

$$\|y_t^0 - y_\lambda^*(x_t)\|^2 + \|z_t^0 - y^*(x_t)\|^2$$

$$\leq \underbrace{\left(\frac{1}{2}\right)^t \left(\|y_0 - y_\lambda^*(x_0)\|^2 + \|y_0 - y^*(x_0)\|^2\right) + \frac{34L_g^2}{\mu^2}\sum_{j=0}^{t-1}\left(\frac{1}{2}\right)^{t-1-j}\|x_{j+1} - x_j\|^2}_{:=(*)}.$$

Plug into Equation 24, which, in conjunction with Equation 23, yields that

$$\mathcal{L}_\lambda^*(x_{t+1}) \leq \mathcal{L}_\lambda^*(x_t) - \frac{\eta}{2}\|\nabla\mathcal{L}_\lambda^*(x_t)\|^2 - \frac{1}{4\eta}\|x_{t+1} - x_t\|^2 + 2\eta \times \underbrace{\lambda^2 L_g^2 \exp\left(-\frac{\mu K}{4L_g}\right)}_{:=\gamma} \times (*).$$

Telescoping over t further yields

$$\begin{aligned}\frac{\eta}{2}\sum_{t=0}^{T-1}\|\nabla\mathcal{L}_\lambda^*(x_t)\|^2 \leq \mathcal{L}_\lambda^*(x_0) - \inf_{x\in\mathbb{R}^{d_x}}\mathcal{L}_\lambda^*(x) + 4\eta\gamma\left(\|y_0 - y_\lambda^*(x_0)\|^2 + \|y_0 - y^*(x_0)\|^2\right) \\ - \left(\frac{1}{4\eta} - \frac{136\eta\gamma L_g^2}{\mu^2}\right)\sum_{t=0}^{T-1}\|x_{t+1} - x_t\|^2\end{aligned} \tag{28}$$

Let $K = \mathcal{O}(\kappa\log(\lambda\kappa)) = \mathcal{O}(\kappa\log(\ell\kappa/\epsilon))$ such that γ is sufficiently small with

$$\gamma \leq \min\left\{\frac{\mu^2}{1088\eta^2 L_g^2}, \frac{1}{4\eta}\right\}.$$

Then we have,

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla\mathcal{L}_\lambda^*(x_t)\|^2 \leq \frac{2}{\eta T}\left(\mathcal{L}_\lambda^*(x_0) - \inf_{x\in\mathbb{R}^{d_x}}\mathcal{L}_\lambda^*(x) + \|y_0 - y_\lambda^*(x_0)\|^2 + \|y_0 - y^*(x_0)\|^2\right).$$

This concludes the proof. □

E Proofs of Finding Second-Order Stationarity

The following theorem generalizes the result of Perturbed Gradient Descent [52] to Inexact Perturbed Gradient Descent by allowing an inexact gradient $\hat{\nabla}h(x)$ that is close to the exact gradient $\nabla h(x)$.

Algorithm 4 Inexact Perturbed Gradient Descent

1: **for** $t = 0, 1, \cdots, T-1$
2: **if** $\|\hat{\nabla}h(x_t)\| \leq \frac{4}{5}\epsilon$ **and** no perturbation added in the last $\mathcal{T}$ steps
3: $x_t = x_t - \eta\xi_t$, where $\xi_t \sim \mathbb{B}(r)$
4: **end if**
5: $x_{t+1} = x_t - \eta\hat{\nabla}h(x_t)$
6: **end for**

Theorem E.1 (Lemma 3.9 by Huang et al. [45]). *Suppose $h(z) : \mathbb{R}^d \to \mathbb{R}$ is L-gradient Lipschitz and ρ-Hessian Lipschitz. Set the parameters in Algorithm 4 as*

$$\eta = \frac{1}{L}, \quad r = \frac{\epsilon}{400\iota^3}, \quad \mathcal{T} = \frac{L}{\sqrt{\rho\epsilon}}\cdot\iota \ \text{ with } \ \iota \geq 1 \ \text{ and } \ \delta \geq \frac{L\sqrt{d}}{\sqrt{\rho\epsilon}}\iota^2 2^{8-\iota}. \tag{29}$$

Once the following inequality holds in each iteration:

$$\|\hat{\nabla}h(x_t) - \nabla h(x_t)\| \leq \underbrace{\min\left\{\frac{1}{20\iota^2}, \frac{1}{16\iota^2 2^\iota}\right\}}_{:=\zeta}\epsilon, \tag{30}$$

then we can find a point x_t satisfying

$$\|\nabla h(x_t)\| \le \epsilon, \quad \nabla^2 h(x_t) \succeq -\sqrt{\rho\epsilon}\, I_d$$

within $T = \mathcal{O}\left(\iota^4 \Delta L \epsilon^{-2}\right)$ iterations with probability $1-\delta$, where $\Delta = h(x_0) - \inf_{x\in\mathbb{R}^d} h(x)$.

Then we can show the convergence of perturbed F^{2}BA.

Theorem 5.1. *Suppose both Assumption 3.1 and 3.2 hold. Define $\Delta := \varphi(x_0) - \inf_{x\in\mathbb{R}^{d_x}} \varphi(x)$ and $R := \|y_0 - y^*(x_0)\|^2$. Let $\eta \asymp \ell^{-1}\kappa^{-3}$, $\lambda \asymp \max\left\{\kappa/R,\ \ell\kappa^2/\Delta,\ \ell\kappa^3/\epsilon,\ \ell\kappa^6/\sqrt{\epsilon}\right\}$ and set other parameters in Algorithm 1 as*

$$\alpha = \frac{1}{L_g},\ \tau = \frac{1}{2\lambda L_g},\ K_t = \tilde{\mathcal{O}}\left(\frac{L_g \log \delta_t}{\mu}\right),$$

where δ_t is defined via the recursion

$$\delta_{t+1} = \frac{1}{2}\delta_t + \frac{34L_g^2}{\mu^2}\|x_{t+1} - x_t\|^2, \quad \delta_0 = \mathcal{O}(R), \tag{9}$$

then it can find an ϵ-second-order stationary point of $\varphi(x)$ within $T = \tilde{\mathcal{O}}\left(\ell\kappa^4\epsilon^{-2}\right)$ first-order oracle calls, where ℓ, κ are defined in Definition 3.3 and the notation $\tilde{\mathcal{O}}(\cdot)$ hides logarithmic factors of d_x, κ, ℓ, and ϵ.

Proof. Let L be the gradient Lipschitz coefficient and ρ be the Hessian Lipschitz coefficient of $\mathcal{L}_\lambda^*(x)$, respectively. According to Lemma 4.1, Lemma 5.1, Lemma B.2 and the setting of λ, we have

a. $\sup_{x\in\mathbb{R}^{d_x}} \|\nabla\mathcal{L}_\lambda^*(x) - \nabla\varphi(x)\| = \mathcal{O}(\epsilon)$.

b. $\mathcal{L}_\lambda^*(x_0) - \inf_{x\in\mathbb{R}^{d_x}} \mathcal{L}_\lambda^*(x) = \mathcal{O}(\Delta)$.

c. $\|y_0 - y_\lambda^*(x_0)\|^2 + \|y_0 - y^*(x_0)\|^2 = \mathcal{O}(R)$.

d. $L := \sup_{x\in\mathbb{R}^{d_x}} \|\nabla^2\mathcal{L}_\lambda^*(x)\| = \mathcal{O}(\ell\kappa^3)$.

e. $\rho := \sup_{x\in\mathbb{R}^{d_x}} \|\nabla^3\mathcal{L}_\lambda^*(x)\| = \mathcal{O}(\ell\kappa^5)$.

Then it suffices to show that the algorithm can find an ϵ-second-order stationary point of $\mathcal{L}_\lambda^*(x)$ under our choice of parameters. We apply Theorem E.1 to prove this result. Recall ζ defined in Equation 30. It remains to show that

$$\|\nabla\mathcal{L}_\lambda^*(x_t) - \nabla\mathcal{L}_\lambda^*(x_t)\| \le \zeta \tag{31}$$

holds for all t. Recall that

$$\|\hat{\nabla}\mathcal{L}_\lambda^*(x_t) - \nabla\mathcal{L}_\lambda^*(x_t)\| \le 2\lambda L_g\|y_t^K - y_\lambda^*(x_t)\| + \lambda L_g\|z_t^K - y^*(x_t)\|.$$

If we let

$$K_t = \frac{4L_g}{\mu}\log\left(\frac{4\lambda L_g(\|y_t^0 - y_\lambda^*(x_t)\|^2 + \|z_t^0 - y^*(x_t)\|^2)}{\zeta}\right) \tag{32}$$

Then by Theorem D.1, we have

$$\max\left\{\|y_t^K - y_\lambda^*(x_t)\|,\ \|z_t^K - y^*(x_t)\|\right\} \le \frac{\zeta}{4\lambda L_g}, \tag{33}$$

which implies Equation 31. The only remaining issue is that the algorithm does not know the value of $\delta_t := \|y_t^0 - y_\lambda^*(x_t)\|^2 + \|z_t^0 - y^*(x_t)\|^2$ in Equation 32. We address this issue by showing a upper bound of it. Suppose that we have already upper bounded δ_t, we can then use it to upper bound δ_{t+1}. By Equation 25, if we let $K_t \ge 8L_g/\mu$ for all t, then we can show that

$$\delta_{t+1} \le \frac{1}{2}\delta_t + \frac{34L_g^2}{\mu^2}\|x_{t+1} - x_t\|^2.$$

Note that the value of $\|x_{t+1} - x_t\|^2$ is known in advance. This recursion gives a valid upper bound of δ_t that can be used to specify K_t in the algorithm. Algorithm using this choice of K_t can provably find an ϵ-second-order stationary point of $\mathcal{L}_\lambda^*(x)$ within

$$\sum_{t=0}^{T-1} K_t \leq \frac{4L_g}{\mu} \sum_{t=0}^{T-1} \log\left(\frac{4\lambda L_g \delta_t}{\zeta}\right) \leq \frac{4L_g T}{\mu} \log\left(\frac{4\lambda L_g \sum_{t=0}^{T-1} \delta_t}{\zeta T}\right) \tag{34}$$

first-order oracle calls. Telescoping over t, we obtain

$$\sum_{t=0}^{T-1} \delta_t \leq 2\delta_0 + \frac{68L_g^2}{\mu^2} \sum_{t=0}^{T-1} \|x_{t+1} - x_t\|^2. \tag{35}$$

By Equation 28, if we let $K_t \geq \Omega(\kappa \log(\lambda\kappa))$ for all t, we can obtain

$$\frac{1}{8\eta} \sum_{t=0}^{T-1} \|x_{t+1} - x_t\|^2 \leq \mathcal{L}_\lambda^*(x_0) - \inf_{x \in \mathbb{R}^{d_x}} \mathcal{L}_\lambda^*(x) + \|y_0 - y_\lambda^*(x_0)\|^2 + \|y_0 - y^*(x_0)\|^2.$$

Plugging into Equation 35 and then Equation 34 yields the upper complexity bound of $\tilde{\mathcal{O}}(\ell\kappa^4\epsilon^{-2})$ as claimed. □