Bilevel Optimization: Convergence Analysis and Enhanced Design

Kaiyi Ji [1] **Junjie Yang** [1] **Yingbin Liang** [1]

Abstract

Bilevel optimization has arisen as a powerful tool for many machine learning problems such as meta-learning, hyperparameter optimization, and reinforcement learning. In this paper, we investigate the nonconvex-strongly-convex bilevel optimization problem. For deterministic bilevel optimization, we provide a comprehensive convergence rate analysis for two popular algorithms respectively based on approximate implicit differentiation (AID) and iterative differentiation (ITD). For the AID-based method, we orderwisely improve the previous convergence rate analysis due to a more practical parameter selection as well as a warm start strategy, and for the ITD-based method we establish the first theoretical convergence rate. Our analysis also provides a quantitative comparison between ITD and AID based approaches. For stochastic bilevel optimization, we propose a novel algorithm named stocBiO, which features a sample-efficient hypergradient estimator using efficient Jacobian- and Hessian-vector product computations. We provide the convergence rate guarantee for stocBiO, and show that stocBiO outperforms the best known computational complexities orderwisely with respect to the condition number κ and the target accuracy ϵ. We further validate our theoretical results and demonstrate the efficiency of bilevel optimization algorithms by the experiments on meta-learning and hyperparameter optimization.

1. Introduction

Bilevel optimization has received significant attention recently and become an influential framework in various machine learning applications including meta-learning (Franceschi et al., 2018; Bertinetto et al., 2018; Rajeswaran et al., 2019; Ji et al., 2020a), hyperparameter optimization (Franceschi et al., 2018; Shaban et al., 2019; Feurer & Hutter, 2019), reinforcement learning (Konda & Tsitsiklis, 2000; Hong et al., 2020), and signal processing (Kunapuli et al., 2008; Flamary et al., 2014). A general bilevel optimization takes the following formulation.

$$\begin{aligned} &\min_{x\in\mathbb{R}^p} \Phi(x) := f(x, y^*(x)) \\ &\text{s.t.} \quad y^*(x) = \arg\min_{y\in\mathbb{R}^q} g(x, y), \end{aligned} \tag{1}$$

where the upper- and inner-level functions f and g are both jointly continuously differentiable. The goal of eq. (1) is to minimize the objective function $\Phi(x)$ with respect to (w.r.t.) x, where $y^*(x)$ is obtained by solving the lower-level minimization problem. In this paper, we focus on the setting where the lower-level function g is strongly convex w.r.t. y, and the upper-level objective function $\Phi(x)$ is nonconvex. Such geometrics commonly exist in many applications such as meta-learning and hyperparameter optimization, where g corresponds to an empirical loss with a strongly-convex regularizer and x are parameters of neural networks.

A broad collection of algorithms have been proposed to solve bilevel optimization problems. For example, Hansen et al. 1992; Shi et al. 2005; Moore 2010 reformulated the bilevel problem in eq. (1) into a single-level constrained problem based on the optimality conditions of the lower-level problem. However, such type of methods often involve a large number of constraints, and are hard to implement in machine learning applications. Recently, more efficient gradient-based bilevel optimization algorithms have been proposed, which can be generally categorized into the approximate implicit differentiation (AID) based approach (Domke, 2012; Pedregosa, 2016; Gould et al., 2016; Liao et al., 2018; Ghadimi & Wang, 2018; Grazzi et al., 2020; Lorraine et al., 2020) and the iterative differentiation (ITD) based approach (Domke, 2012; Maclaurin et al., 2015; Franceschi et al., 2017; 2018; Shaban et al., 2019; Grazzi et al., 2020). However, most of these studies have focused on the asymptotic convergence analysis, and the nonasymptotic convergence rate analysis (that characterizes how fast an algorithm converges) has not been well explored except a few attempts recently. Ghadimi & Wang 2018 provided the convergence rate analysis for the ITD-based approach. Grazzi et al. 2020 provided the iteration complexity for the hypergradient computation via ITD and AID, but did not

[1]Department of Electrical and Computer Engineering, The Ohio State University. Correspondence to: Kaiyi Ji <ji.367@osu.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

characterize the convergence rate for the entire execution of algorithms.

- Thus, the first focus of this paper is to develop a *comprehensive and sharper* theory, which covers a broader class of bilevel optimizers via ITD and AID techniques, and more importantly, improves existing analysis with a more practical parameter selection and orderwisely lower computational complexity.

The *stochastic* bilevel optimization often occurs in applications where fresh data are sampled for algorithm iterations (e.g., in reinforcement learning (Hong et al., 2020)) or the sample size of training data is large (e.g., hyperparameter optimization (Franceschi et al., 2018), Stackelberg game (Roth et al., 2016)). Typically, the objective function is given by

$$\begin{aligned}&\min_{x\in\mathbb{R}^p}\Phi(x)=f(x,y^*(x))=\begin{cases}\frac{1}{n}\sum_{i=1}^n F(x,y^*(x);\xi_i)\\ \mathbb{E}_\xi\left[F(x,y^*(x);\xi)\right]\end{cases}\\ &\text{s.t. } y^*(x)=\arg\min_{y\in\mathbb{R}^q} g(x,y)=\begin{cases}\frac{1}{m}\sum_{i=1}^m G(x,y;\zeta_i)\\ \mathbb{E}_\zeta\left[G(x,y;\zeta)\right]\end{cases}\end{aligned}\tag{2}$$

where $f(x,y)$ and $g(x,y)$ take either the expectation form w.r.t. the random variables ξ and ζ or the finite-sum form over given data $\mathcal{D}_{n,m}=\{\xi_i,\zeta_j,i=1,...,n;j=1,...,m\}$ often with large sizes n and m. During the optimization process, data batch is sampled via the distributions of ξ and ζ or from the set $\mathcal{D}_{n,m}$. For such a stochastic setting, Ghadimi & Wang 2018 proposed a bilevel stochastic approximation (BSA) method via single-sample gradient and Hessian estimates. Based on such a method, Hong et al. 2020 further proposed a two-timescale stochastic approximation (TTSA) algorithm, and showed that TTSA achieves a better trade-off between the complexities of inner- and outer-loop optimization stages than BSA.

- The second focus of this paper is to design a more sample-efficient algorithm for bilevel stochastic optimization, which achieves lower computational complexity by orders of magnitude than BSA and TTSA.

1.1. Main Contributions

Our main contributions lie in developing shaper theory and provably faster algorithms for nonconvex-strongly-convex bilevel deterministic and stochastic optimization problems, respectively. Our analysis involves several new developments, which can be of independent interest.

We first provide a unified convergence rate and complexity analysis for both ITD and AID based bilevel optimizers, which we call as ITD-BiO and AID-BiO. Compared to existing analysis in Ghadimi & Wang 2018 for AID-BiO that requires a continuously increasing number of inner-loop steps to achieve the guarantee, our analysis allows a constant number of inner-loop steps as often used in practice. In addition, we introduce a warm start initialization for the inner-loop updates and the outer-loop hypergradient estimation, which allows us to backpropagate the tracking errors to previous loops, and yields an improved computational complexity. As shown in Table 1, the gradient complexities $\text{Gc}(f,\epsilon)$, $\text{Gc}(g,\epsilon)$, and Jacobian- and Hessian-vector product complexities $\text{JV}(g,\epsilon)$ and $\text{HV}(g,\epsilon)$ of AID-BiO to attain an ϵ-accurate stationary point improve those of Ghadimi & Wang 2018 by the order of κ, $\kappa\epsilon^{-1/4}$, κ, and κ, respectively, where κ is the condition number. In addition, our analysis shows that AID-BiO requires less computations of Jacobian- and Hessian-vector products than ITD-BiO by an order of κ and $\kappa^{1/2}$, which implies that AID can be more computationally and memory efficient than ITD.

We then propose a stochastic bilevel optimizer (stocBiO) to solve the stochastic bilevel optimization problem in eq. (2). Our algorithm features a *mini-batch* hypergradient estimation via implicit differentiation, where the core design involves a sample-efficient hypergradient estimator via the Neumann series. As shown in Table 2, the gradient complexities of our proposed algorithm w.r.t. F and G improve upon those of BSA (Ghadimi & Wang, 2018) by an order of κ and ϵ^{-1}, respectively. In addition, the Jacobian-vector product complexity $\text{JV}(G,\epsilon)$ of our algorithm improves that of BSA by an order of κ. In terms of the target accuracy ϵ, our computational complexities improve those of TTSA (Hong et al., 2020) by an order of $\epsilon^{-1/2}$.

Our results further provide the theoretical complexity guarantee for ITD-BiO, AID-BiO and stocBiO in meta-learning and hyperparameter optimization. The experiments validate our theoretical results for deterministic bilevel optimization, and demonstrate the superior efficiency of stocBiO for stochastic bilevel optimization.

1.2. Related Work

Bilevel optimization approaches: Bilevel optimization was first introduced by Bracken & McGill 1973. Since then, a number of bilevel optimization algorithms have been proposed, which include but not limited to constraint-based methods (Shi et al., 2005; Moore, 2010) and gradient-based methods (Domke, 2012; Pedregosa, 2016; Gould et al., 2016; Maclaurin et al., 2015; Franceschi et al., 2018; Ghadimi & Wang, 2018; Liao et al., 2018; Shaban et al., 2019; Hong et al., 2020; Liu et al., 2020; Li et al., 2020; Grazzi et al., 2020; Lorraine et al., 2020; Ji & Liang, 2021). Among them, Ghadimi & Wang 2018; Hong et al. 2020 provided the complexity analysis for their proposed methods for the nonconvex-strongly-convex bilevel optimization problem. For such a problem, this paper develops a general and enhanced convergence rate analysis for gradient-based

Table 1. Comparison of bilevel deterministic optimization algorithms.

Algorithm	Gc(f,ϵ)	Gc(g,ϵ)	JV(g,ϵ)	HV(g,ϵ)
AID-BiO (Ghadimi & Wang, 2018)	$\mathcal{O}(\kappa^4\epsilon^{-1})$	$\mathcal{O}(\kappa^5\epsilon^{-5/4})$	$\mathcal{O}\left(\kappa^4\epsilon^{-1}\right)$	$\widetilde{\mathcal{O}}\left(\kappa^{4.5}\epsilon^{-1}\right)$
AID-BiO (this paper)	$\mathcal{O}(\kappa^3\epsilon^{-1})$	$\mathcal{O}(\kappa^4\epsilon^{-1})$	$\mathcal{O}\left(\kappa^3\epsilon^{-1}\right)$	$\mathcal{O}\left(\kappa^{3.5}\epsilon^{-1}\right)$
ITD-BiO (this paper)	$\mathcal{O}(\kappa^3\epsilon^{-1})$	$\widetilde{\mathcal{O}}(\kappa^4\epsilon^{-1})$	$\widetilde{\mathcal{O}}\left(\kappa^4\epsilon^{-1}\right)$	$\widetilde{\mathcal{O}}\left(\kappa^4\epsilon^{-1}\right)$

Gc(f,ϵ) and Gc(g,ϵ): number of gradient evaluations w.r.t. f and g. κ : condition number.
JV(g,ϵ): number of Jacobian-vector products $\nabla_x\nabla_y g(x,y)v$. Notation $\widetilde{\mathcal{O}}$: omit $\log\frac{1}{\epsilon}$ terms.
HV(g,ϵ): number of Hessian-vector products $\nabla_y^2 g(x,y)v$.

Table 2. Comparison of bilevel stochastic optimization algorithms.

Algorithm	Gc(F,ϵ)	Gc(G,ϵ)	JV(G,ϵ)	HV(G,ϵ)
TTSA (Hong et al., 2020)	$\mathcal{O}(\text{poly}(\kappa)\epsilon^{-\frac{5}{2}})^*$	$\mathcal{O}(\text{poly}(\kappa)\epsilon^{-\frac{5}{2}})$	$\mathcal{O}(\text{poly}(\kappa)\epsilon^{-\frac{5}{2}})$	$\mathcal{O}(\text{poly}(\kappa)\epsilon^{-\frac{5}{2}})$
BSA (Ghadimi & Wang, 2018)	$\mathcal{O}(\kappa^6\epsilon^{-2})$	$\mathcal{O}(\kappa^9\epsilon^{-3})$	$\mathcal{O}\left(\kappa^6\epsilon^{-2}\right)$	$\widetilde{\mathcal{O}}\left(\kappa^6\epsilon^{-2}\right)$
stocBiO (this paper)	$\mathcal{O}(\kappa^5\epsilon^{-2})$	$\mathcal{O}(\kappa^9\epsilon^{-2})$	$\mathcal{O}\left(\kappa^5\epsilon^{-2}\right)$	$\widetilde{\mathcal{O}}\left(\kappa^6\epsilon^{-2}\right)$

* We use poly(κ) because Hong et al. 2020 does not provide the explicit dependence on κ.

bilevel optimizers for the deterministic setting, and proposes a novel algorithm for the stochastic setting with order-level lower computational complexity than the existing results.

Other types of loss geometries have also been studied. For example, Liu et al. 2020; Li et al. 2020 assumed that the lower- and upper-level functions $g(x,\cdot)$ and $f(x,\cdot)$ are convex and strongly convex, and provided an asymptotic analysis for their methods. Ghadimi & Wang 2018; Hong et al. 2020 studied the setting where $\Phi(\cdot)$ is strongly convex or convex, and $g(x,\cdot)$ is strongly convex.

Bilevel optimization in meta-learning: Bilevel optimization framework has been successfully applied to meta-learning recently (Snell et al., 2017; Franceschi et al., 2018; Rajeswaran et al., 2019; Zügner & Günnemann, 2019; Ji et al., 2020a;b). For example, Snell et al. 2017 proposed a bilevel optimization procedure for meta-learning to learn a common embedding model for all tasks. Rajeswaran et al. 2019 reformulated the model-agnostic meta-learning (MAML) (Finn et al., 2017) as bilevel optimization, and proposed iMAML via implicit gradient. Our work provides a theoretical guarantee for two popular types of bilevel optimizer, i.e., AID-BiO and ITD-BiO, for meta-learning.

Bilevel optimization in hyperparameter optimization: Hyperparameter optimization has become increasingly important as a powerful tool in the automatic machine learning (autoML) (Okuno et al., 2018; Yu & Zhu, 2020). Recently, various bilevel optimization algorithms have been proposed for hyperparameter optimization, which include implicit differentiation based methods (Pedregosa, 2016), dynamical system based methods via reverse or forward gradient computation (Franceschi et al., 2017; 2018; Shaban et al., 2019), etc. Our work demonstrates superior efficiency of the proposed stocBiO algorithm in hyperparameter optimization.

2. Algorithms

In this section, we describe two popular types of *deterministic* bilevel optimization algorithms, and propose a new algorithm for *stochastic* bilevel optimization.

2.1. Algorithms for Deterministic Bilevel Optimization

As shown in Algorithm 1, we describe two popular types of deterministic bilevel optimizers respectively based on AID and ITD (referred to as AID-BiO and ITD-BiO) for solving the problem eq. (1).

Both AID-BiO and ITD-BiO update in a nested-loop manner. In the inner loop, both of them run D steps of gradient decent (GD) to find an approximation point y_k^D close to $y^*(x_k)$. Note that we choose the initialization y_k^0 of each inner loop as the output y_{k-1}^D of the preceding inner loop rather than a random start. Such a *warm start* allows us to backpropagate the tracking error $\|y_k^D - y^*(x_k)\|$ to previous loops, and yields an improved computational complexity.

At the outer loop, AID-BiO first solves v_k^N from a linear system $\nabla_y^2 g(x_k,y_k^D)v = \nabla_y f(x_k,y_k^D)$[1] using N steps of conjugate-gradient (CG) starting from v_k^0 (where we also adopt a warm start with $v_k^0 = v_{k-1}^N$), and then constructs

$$\widehat{\nabla}\Phi(x_k) = \nabla_x f(x_k,y_k^T) - \nabla_x\nabla_y g(x_k,y_k^T)v_k^N \qquad (3)$$

as an estimate of the true hypergradient $\nabla\Phi(x_k)$, whose form is given by the following proposition.

Proposition 1. *Hypergradient $\nabla\Phi(x_k)$ takes the forms of*

$$\nabla\Phi(x_k) = \nabla_x f(x_k,y^*(x_k)) - \nabla_x\nabla_y g(x_k,y^*(x_k))v_k^*, \quad (4)$$

[1] Solving this linear system is equivalent to solving a quadratic programming $\min_v \frac{1}{2}v^T\nabla_y^2 g(x_k,y_k^D)v - v^T\nabla_y f(x_k,y_k^D)$.

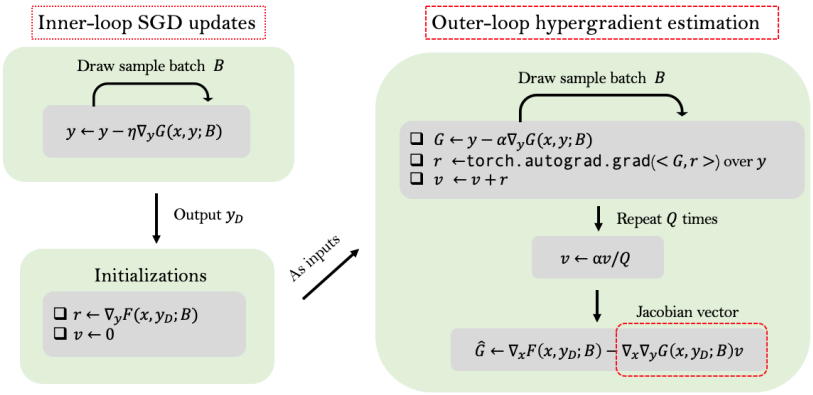


Figure 1. Illustration of hyperparameter estimation in our proposed stocBiO algorithm. Note that the hyperparameter estimation (lines 9-10 in Algorithm 2) involves only computations of automatic differentiation over scalar $< G_j(y), r_i >$ w.r.t. y. In addition, our implementation applies the function *torch.autograd.grad* in PyTorch, which automatically determines the size of Jacobians. More details can be found in our code via https://github.com/JunjieYang97/StocBio_hp.

Algorithm 1 Bilevel algorithms via AID or ITD

Input: K, D, N, stepsizes α, β, initializations x_0, y_0, v_0.
for $k = 0, 1, 2, ..., K$ **do**
  Set $y_k^0 = y_{k-1}^D$ if $k > 0$ and y_0 otherwise
  for $t = 1, ..., D$ **do**
    Update $y_k^t = y_k^{t-1} - \alpha \nabla_y g(x_k, y_k^{t-1})$
  end for
  Hypergradient estimation via
  AID: 1) set $v_k^0 = v_{k-1}^N$ if $k > 0$ and v_0 otherwise
    2) solve v_k^N from $\nabla_y^2 g(x_k, y_k^D) v = \nabla_y f(x_k, y_k^D)$ via N steps of CG starting from v_k^0
    3) get Jacobian-vector product $\nabla_x \nabla_y g(x_k, y_k^D) v_k^N$ via automatic differentiation
    4) $\widehat{\nabla} \Phi(x_k) = \nabla_x f(x_k, y_k^D) - \nabla_x \nabla_y g(x_k, y_k^D) v_k^N$
  ITD: compute $\widehat{\nabla} \Phi(x_k) = \frac{\partial f(x_k, y_k^D)}{\partial x_k}$ via backpropagation
  Update $x_{k+1} = x_k - \beta \widehat{\nabla} \Phi(x_k)$
end for

where v_k^* is the solution of the following linear system

$$\nabla_y^2 g(x_k, y^*(x_k)) v = \nabla_y f(x_k, y^*(x_k)).$$

As shown in Domke 2012; Grazzi et al. 2020, the construction of eq. (3) involves only Hessian-vector products in solving v_N via CG and Jacobian-vector product $\nabla_x \nabla_y g(x_k, y_k^D) v_k^N$, which can be efficiently computed and stored via existing automatic differentiation packages.

As a comparison, the outer loop of ITD-BiO computes the gradient $\frac{\partial f(x_k, y_k^D(x_k))}{\partial x_k}$ as an approximation of the hypergradient $\nabla \Phi(x_k) = \frac{\partial f(x_k, y^*(x_k))}{\partial x_k}$ via backpropagation, where we write $y_k^D(x_k)$ because the output y_k^D of the inner loop has a dependence on x_k through the inner-loop iterative GD updates. The explicit form of the estimate $\frac{\partial f(x_k, y_k^D(x_k))}{\partial x_k}$ is given by the following proposition via the chain rule. For notation simplification, let $\prod_{j=D}^{D-1}(\cdot) = I$.

Proposition 2. *$\frac{\partial f(x_k, y_k^D(x_k))}{\partial x_k}$ takes the analytical form of:*

$$\frac{\partial f(x_k, y_k^D)}{\partial x_k} = \nabla_x f(x_k, y_k^D) - \alpha \sum_{t=0}^{D-1} \nabla_x \nabla_y g(x_k, y_k^t) \times \prod_{j=t+1}^{D-1} (I - \alpha \nabla_y^2 g(x_k, y_k^j)) \nabla_y f(x_k, y_k^D).$$

Proposition 2 shows that the differentiation involves the computations of second-order derivatives such as Hessian $\nabla_y^2 g(\cdot, \cdot)$. Since efficient Hessian-free methods have been successfully deployed in the existing automatic differentiation tools, computing these second-order derivatives reduces to more efficient computations of Jacobian- and Hessian-vector products.

2.2. Algorithm for Stochastic Bilevel Optimization

We propose a new stochastic bilevel optimizer (stocBiO) in Algorithm 2 to solve the problem eq. (2). It has a double-loop structure similar to Algorithm 1, but runs D steps of stochastic gradient decent (SGD) at the inner loop to obtain an approximated solution y_k^D. Based on the output y_k^D of the inner loop, stocBiO first computes a gradient $\nabla_y F(x_k, y_k^D; \mathcal{D}_F)$ over a sample batch $\mathcal{D}_F$, and then computes a vector v_Q as an estimated solution of the linear system $\nabla_y^2 g(x_k, y^*(x_k)) v = \nabla_y f(x_k, y^*(x_k))$ via Algorithm 3. Here, v_Q takes a form of

$$v_Q = \eta \sum_{q=-1}^{Q-1} \prod_{j=Q-q}^{Q} (I - \eta \nabla_y^2 G(x_k, y_k^D; \mathcal{B}_j)) v_0, \quad (5)$$

Algorithm 2 Stochastic bilevel optimizer (stocBiO)

Input: K, D, Q, stepsizes α and β, initializations x_0 and y_0.
for $k = 0, 1, 2, ..., K$ **do**
  Set $y_k^0 = y_{k-1}^D$ if $k > 0$ and y_0 otherwise
  for $t = 1, ..., D$ **do**
    Draw a sample batch $\mathcal{S}_{t-1}$
    Update $y_k^t = y_k^{t-1} - \alpha \nabla_y G(x_k, y_k^{t-1}; \mathcal{S}_{t-1})$
  end for
  Draw sample batches $\mathcal{D}_F, \mathcal{D}_H$ and $\mathcal{D}_G$
  Compute gradient $v_0 = \nabla_y F(x_k, y_k^D; \mathcal{D}_F)$
  Construct estimate v_Q via Algorithm 3 given v_0
  Compute $\nabla_x \nabla_y G(x_k, y_k^D; \mathcal{D}_G) v_Q$
  Compute gradient estimate $\widehat{\nabla}\Phi(x_k)$ via eq. (6)
  Update $x_{k+1} = x_k - \beta \widehat{\nabla}\Phi(x_k)$
end for

Algorithm 3 Construct v_Q given v_0

Input: Integer Q, samples $\mathcal{D}_H = \{\mathcal{B}_j\}_{j=1}^Q$ and constant η.
for $j = 1, 2, ..., Q$ **do**
  Sample $\mathcal{B}_j$ and compute $G_j(y) = y - \eta \nabla_y G(x, y; \mathcal{B}_j)$
end for
Set $r_Q = v_0$
for $i = Q, ..., 1$ **do**
  $r_{i-1} = \partial\big(G_i(y) r_i\big)/\partial y = r_i - \eta \nabla_y^2 G(x, y; \mathcal{B}_i) r_i$ via automatic differentiation
end for
Return $v_Q = \eta \sum_{i=0}^Q r_i$

where $v_0 = \nabla_y F(x_k, y_k^D; \mathcal{D}_F)$, $\{\mathcal{B}_j, j = 1, ..., Q\}$ are mutually-independent sample sets, Q and η are constants, and we let $\prod_{Q+1}^Q(\cdot) = I$ for notational simplification. Our construction of v_Q, i.e., Algorithm 3, is motived by the Neumann series $\sum_{i=0}^\infty U^k = (I - U)^{-1}$, and involves only Hessian-vector products rather than Hessians, and hence is computationally and memory efficient. **This procedure is illustrated in Figure 1.**

Then, we construct

$$\widehat{\nabla}\Phi(x_k) = \nabla_x F(x_k, y_k^D; \mathcal{D}_F) - \nabla_x \nabla_y G(x_k, y_k^D; \mathcal{D}_G) v_Q \quad (6)$$

as an estimate of hypergradient $\nabla\Phi(x_k)$. Compared to the deterministic case, it is more challenging to design a sample-efficient Hypergradient estimator in the stochastic case. For example, instead of choosing the same batch sizes for all $\mathcal{B}_j, j = 1, ..., Q$ in eq. (5), our analysis captures the different impact of components $\nabla_y^2 G(x_k, y_k^D; \mathcal{B}_j), j = 1, ..., Q$ on the hypergradient estimation variance, and inspires an adaptive and more efficient choice by setting $|\mathcal{B}_{Q-j}|$ to decay exponentially with j from 0 to $Q - 1$. By doing so, we achieve an improved complexity.

3. Definitions and Assumptions

Let $z = (x, y)$ denote all parameters. For simplicity, suppose sample sets $\mathcal{S}_t$ for all $t = 0, ..., D - 1$, $\mathcal{D}_G$ and $\mathcal{D}_F$ have the sizes of S, D_g and D_f, respectively. In this paper, we focus on the following types of loss functions for both the deterministic and stochastic cases.

Assumption 1. *The lower-level function $g(x, y)$ is μ-strongly-convex w.r.t. y and the total objective function $\Phi(x) = f(x, y^*(x))$ is nonconvex w.r.t. x. For the stochastic setting, the same assumptions hold for $G(x, y; \zeta)$ and $\Phi(x)$, respectively.*

Since $\Phi(x)$ is nonconvex, algorithms are expected to find an ϵ-accurate stationary point defined as follows.

Definition 1. *We say $\bar{x}$ is an ϵ-accurate stationary point for the objective function $\Phi(x)$ in eq. (2) if $\mathbb{E}\|\nabla\Phi(\bar{x})\|^2 \le \epsilon$, where $\bar{x}$ is the output of an algorithm.*

In order to compare the performance of different bilevel algorithms, we adopt the following metrics of complexity.

Definition 2. *For a function $f(x, y)$ and a vector v, let $\mathrm{Gc}(f, \epsilon)$ be the number of the partial gradient $\nabla_x f$ or $\nabla_y f$, and let $\mathrm{JV}(g, \epsilon)$ and $\mathrm{HV}(g, \epsilon)$ be the number of Jacobian-vector products $\nabla_x \nabla_y g(x, y)v$. and Hessian-vector products $\nabla_y^2 g(x, y)v$. For the stochastic case, similar metrics are adopted but w.r.t. the stochastic function $F(x, y; \xi)$.*

We take the following standard assumptions on the loss functions in eq. (2), which have been widely adopted in bilevel optimization (Ghadimi & Wang, 2018; Ji et al., 2020a).

Assumption 2. *The loss function $f(z)$ and $g(z)$ satisfy*

- *The function $f(z)$ is M-Lipschitz, i.e., for any z, z',*
$$|f(z) - f(z')| \le M\|z - z'\|.$$
- *$\nabla f(z)$ and $\nabla g(z)$ are L-Lipschitz, i.e., for any z, z',*
$$\|\nabla f(z) - \nabla f(z')\| \le L\|z - z'\|,$$
$$\|\nabla g(z) - \nabla g(z')\| \le L\|z - z'\|.$$

For the stochastic case, the same assumptions hold for $F(z; \xi)$ and $G(z; \zeta)$ for any given ξ and ζ.

As shown in Proposition 1, the gradient of the objective function $\Phi(x)$ involves the second-order derivatives $\nabla_x \nabla_y g(z)$ and $\nabla_y^2 g(z)$. The following assumption imposes the Lipschitz conditions on such high-order derivatives, as also made in Ghadimi & Wang 2018.

Assumption 3. *Suppose the derivatives $\nabla_x \nabla_y g(z)$ and $\nabla_y^2 g(z)$ are τ- and ρ- Lipschitz, i.e.,*

- *For any z, z', $\|\nabla_x \nabla_y g(z) - \nabla_x \nabla_y g(z')\| \le \tau\|z - z'\|$.*
- *For any z, z', $\|\nabla_y^2 g(z) - \nabla_y^2 g(z')\| \le \rho\|z - z'\|$.*

For the stochastic case, the same assumptions hold for $\nabla_x \nabla_y G(z; \zeta)$ and $\nabla_y^2 G(z; \zeta)$ for any ζ.

As typically adopted in the analysis for stochastic optimization, we make the following bounded-variance assumption for the lower-level stochastic function $G(z; \zeta)$.

Assumption 4. *Gradient $\nabla G(z;\zeta)$ has a bounded variance, i.e., $\mathbb{E}_\xi\|\nabla G(z;\zeta)-\nabla g(z)\|^2\le\sigma^2$ for some σ.*

4. Main Results for Bilevel Optimization

4.1. Deterministic Bilevel Optimization

We first characterize the convergence and complexity of AID-BiO. Let $\kappa=\frac{L}{\mu}$ denote the condition number.

Theorem 1 (AID-BiO)**.** *Suppose Assumptions 1, 2, 3 hold. Define a smoothness parameter $L_\Phi=L+\frac{2L^2+\tau M^2}{\mu}+\frac{\rho LM+L^3+\tau ML}{\mu^2}+\frac{\rho L^2M}{\mu^3}=\Theta(\kappa^3)$, choose the stepsizes $\alpha\le\frac{1}{L}$, $\beta=\frac{1}{8L_\Phi}$, and set the inner-loop iteration number $D\ge\Theta(\kappa)$ and the CG iteration number $N\ge\Theta(\sqrt{\kappa})$, where the detailed forms of D,N can be found in Appendix E. Then, the outputs of AID-BiO satisfy*

$$\frac{1}{K}\sum_{k=0}^{K-1}\|\nabla\Phi(x_k)\|^2\le\frac{64L_\Phi(\Phi(x_0)-\inf_x\Phi(x))+5\Delta_0}{K},$$

where $\Delta_0=\|y_0-y^*(x_0)\|^2+\|v_0^*-v_0\|^2>0$.

In order to achieve an ϵ-accurate stationary point, the complexities satisfy

- *Gradient:* $\mathrm{Gc}(f,\epsilon)=\mathcal{O}(\kappa^3\epsilon^{-1}),\mathrm{Gc}(g,\epsilon)=\mathcal{O}(\kappa^4\epsilon^{-1})$.
- *Jacobian- and Hessian-vector product complexities:* $\mathrm{JV}(g,\epsilon)=\mathcal{O}\left(\kappa^3\epsilon^{-1}\right),\mathrm{HV}(g,\epsilon)=\mathcal{O}\left(\kappa^{3.5}\epsilon^{-1}\right)$.

As shown in Table 1, the complexities $\mathrm{Gc}(f,\epsilon)$, $\mathrm{Gc}(g,\epsilon)$, $\mathrm{JV}(g,\epsilon)$ and $\mathrm{HV}(g,\epsilon)$ of our analysis improves that of Ghadimi & Wang 2018 (eq. (2.30) therein) by the order of κ, $\kappa\epsilon^{-1/4}$, κ and κ. Such an improvement is achieved by a refined analysis with a constant number of inner-loop steps, and by a warm start strategy to backpropagate the tracking errors $\|y_k^D-y^*(x_k)\|$ and $\|v_k^N-v_k^*\|$ to previous loops, as also demonstrated by our meta-learning experiments. We next characterize the convergence and complexity performance of the ITD-BiO algorithm.

Theorem 2 (ITD-BiO)**.** *Suppose Assumptions 1, 2, and 3 hold. Define L_Φ as in Theorem 1, and choose $\alpha\le\frac{1}{L}$, $\beta=\frac{1}{4L_\Phi}$ and $D\ge\Theta(\kappa\log\frac{1}{\epsilon})$, where the detailed form of D can be found in Appendix F. Then, we have*

$$\frac{1}{K}\sum_{k=0}^{K-1}\|\nabla\Phi(x_k)\|^2\le\frac{16L_\Phi(\Phi(x_0)-\inf_x\Phi(x))}{K}+\frac{2\epsilon}{3}.$$

In order to achieve an ϵ-accurate stationary point, the complexities satisfy

- *Gradient:* $\mathrm{Gc}(f,\epsilon)=\mathcal{O}(\kappa^3\epsilon^{-1}),\mathrm{Gc}(g,\epsilon)=\widetilde{\mathcal{O}}(\kappa^4\epsilon^{-1})$.
- *Jacobian- and Hessian-vector product complexity:* $\mathrm{JV}(g,\epsilon)=\widetilde{\mathcal{O}}\big(\kappa^4\epsilon^{-1}\big),\mathrm{HV}(g,\epsilon)=\widetilde{\mathcal{O}}\big(\kappa^4\epsilon^{-1}\big)$.

By comparing Theorem 1 and Theorem 2, it can be seen that the complexities $\mathrm{JV}(g,\epsilon)$ and $\mathrm{HV}(g,\epsilon)$ of AID-BiO are better than those of ITD-BiO by the order of κ and $\kappa^{0.5}$, which implies that AID-BiO is more computationally and memory efficient than ITD-BiO, as verified in Figure 2.

4.2. Stochastic Bilevel Optimization

We first characterize the bias and variance of an important component v_Q in eq. (5).

Proposition 3. *Suppose Assumptions 1, 2 and 3 hold. Let $\eta\le\frac{1}{L}$ and choose $|\mathcal{B}_{Q+1-j}|=BQ(1-\eta\mu)^{j-1}$ for $j=1,...,Q$, where $B\ge\frac{1}{Q(1-\eta\mu)^{Q-1}}$. Then, the bias satisfies*

$$\left\|\mathbb{E}v_Q-[\nabla_y^2g(x_k,y_k^D)]^{-1}\nabla_yf(x_k,y_k^D)\right\|\le\mu^{-1}(1-\eta\mu)^{Q+1}M. \quad (7)$$

Furthermore, the estimation variance is given by

$$\mathbb{E}\|v_Q-[\nabla_y^2g(x_k,y_k^D)]^{-1}\nabla_yf(x_k,y_k^D)\|^2\le\frac{4\eta^2L^2M^2}{\mu^2}\frac{1}{B}+\frac{4(1-\eta\mu)^{2Q+2}M^2}{\mu^2}+\frac{2M^2}{\mu^2D_f}. \quad (8)$$

Proposition 3 shows that if we choose Q, B and D_f at the order level of $\mathcal{O}(\log\frac{1}{\epsilon})$, $\mathcal{O}(1/\epsilon)$ and $\mathcal{O}(1/\epsilon)$, the bias and variance are smaller than $\mathcal{O}(\epsilon)$, and the required number of samples is $\sum_{j=1}^Q BQ(1-\eta\mu)^{j-1}=\mathcal{O}\left(\epsilon^{-1}\log\frac{1}{\epsilon}\right)$. Note that the chosen batch size $|\mathcal{B}_{Q+1-j}|$ exponentially decays w.r.t. the index j. In comparison, the uniform choice of all $|\mathcal{B}_j|$ would yield a worse complexity of $\mathcal{O}\big(\epsilon^{-1}(\log\frac{1}{\epsilon})^2\big)$.

We next analyze stocBiO when $\Phi(x)$ is nonconvex.

Theorem 3. *Suppose Assumptions 1, 2, 3 and 4 hold. Define $L_\Phi=L+\frac{2L^2+\tau M^2}{\mu}+\frac{\rho LM+L^3+\tau ML}{\mu^2}+\frac{\rho L^2M}{\mu^3}$, and choose $\beta=\frac{1}{4L_\Phi},\eta<\frac{1}{L}$, and $D\ge\Theta(\kappa\log\kappa)$, where the detailed form of D can be found in Appendix G.3. We have*

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\nabla\Phi(x_k)\|^2\le\mathcal{O}\Big(\frac{L_\Phi}{K}+\kappa^2(1-\eta\mu)^{2Q}+\frac{\kappa^5\sigma^2}{S}+\frac{\kappa^2}{D_g}+\frac{\kappa^2}{D_f}+\frac{\kappa^2}{B}\Big). \quad (9)$$

In order to achieve an ϵ-accurate stationary point, the complexities satisfy

- *Gradient:* $\mathrm{Gc}(F,\epsilon)=\mathcal{O}(\kappa^5\epsilon^{-2}),\mathrm{Gc}(G,\epsilon)=\mathcal{O}(\kappa^9\epsilon^{-2})$.
- *Jacobian- and Hessian-vector product complexities:* $\mathrm{JV}(G,\epsilon)=\mathcal{O}(\kappa^5\epsilon^{-2}),\mathrm{HV}(G,\epsilon)=\widetilde{\mathcal{O}}(\kappa^6\epsilon^{-2})$.

Theorem 3 shows that stocBiO converges sublinearly with the convergence error decaying exponentially w.r.t. Q and sublinearly w.r.t. the batch sizes S,D_g,D_f for gradient estimation and B for Hessian inverse estimation. In addition, it

can be seen that the number D of the inner-loop steps is at a constant level, rather than a typical choice of $\Theta(\log(\frac{1}{\epsilon}))$.

As shown in Table 2, the gradient complexities of our proposed algorithm in terms of F and G improve those of BSA in Ghadimi & Wang 2018 by an order of κ and ϵ^{-1}, respectively. In addition, the Jacobian-vector product complexity $\text{JV}(G,\epsilon)$ of our algorithm improves that of BSA by the order of κ. In terms of the accuracy ϵ, our gradient, Jacobian- and Hessian-vector product complexities improve those of TTSA in Hong et al. 2020 all by an order of $\epsilon^{-0.5}$.

5. Applications to Meta-Learning

Consider the few-shot meta-learning problem with m tasks $\{\mathcal{T}_i, i=1,...,m\}$ sampled from distribution $P_{\mathcal{T}}$. Each task $\mathcal{T}_i$ has a loss function $\mathcal{L}(\phi, w_i; \xi)$ over each data sample ξ, where ϕ are the parameters of an embedding model shared by all tasks, and w_i are the task-specific parameters. The goal of this framework is to find good parameters ϕ for all tasks, and building on the embedded features, each task then adapts its own parameters w_i by minimizing its loss.

The model training takes a bilevel procedure. In the lower-level stage, building on the embedded features, the base learner of task $\mathcal{T}_i$ searches w_i^* as the minimizer of its loss over a training set $\mathcal{S}_i$. In the upper-level stage, the meta-learner evaluates the minimizers $w_i^*, i=1,...,m$ on held-out test sets, and optimizes ϕ of the embedding model over all tasks. Let $\widetilde{w} = (w_1,...,w_m)$ denote all task-specific parameters. Then, the objective function is given by

$$\min_{\phi} \mathcal{L}_{\mathcal{D}}(\phi, \widetilde{w}^*) = \frac{1}{m}\sum_{i=1}^{m} \underbrace{\frac{1}{|\mathcal{D}_i|}\sum_{\xi\in\mathcal{D}_i} \mathcal{L}(\phi, w_i^*; \xi)}_{\mathcal{L}_{\mathcal{D}_i}(\phi, w_i^*)\text{: task-specific upper-level loss}}$$

$$\text{s.t. } \widetilde{w}^* = \arg\min_{\widetilde{w}} \mathcal{L}_{\mathcal{S}}(\phi, \widetilde{w}) = \frac{\sum_{i=1}^{m} \mathcal{L}_{\mathcal{S}_i}(\phi, w_i)}{m}, \quad (10)$$

where $\mathcal{L}_{\mathcal{S}_i}(\phi, w_i) = \frac{1}{|\mathcal{S}_i|}\sum_{\xi\in\mathcal{S}_i} \mathcal{L}(\phi, w_i; \xi) + \mathcal{R}(w_i)$ with a strongly-convex regularizer $\mathcal{R}(w_i)$, e.g., L^2, and $\mathcal{S}_i, \mathcal{D}_i$ are the training and test datasets of task $\mathcal{T}_i$. Note that the lower-level problem is equivalent to solving each w_i^* as a minimizer of the task-specific loss $\mathcal{L}_{\mathcal{S}_i}(\phi, w_i)$ for $i=1,...,m$. In practice, w_i often corresponds to the parameters of the last *linear* layer of a neural network and ϕ are the parameters of the remaining layers (e.g., 4 convolutional layers in Bertinetto et al. 2018; Ji et al. 2020a), and hence the lower-level function is *strongly-convex* w.r.t. $\widetilde{w}$ and the upper-level function $\mathcal{L}_{\mathcal{D}}(\phi, \widetilde{w}^*(\phi))$ is generally nonconvex w.r.t. ϕ. In addition, due to the small sizes of datasets $\mathcal{D}_i$ and $\mathcal{S}_i$ in few-shot learning, all updates for each task $\mathcal{T}_i$ use *full gradient descent* without data resampling. As a result, AID-BiO and ITD-BiO in Algorithm 1 can be applied here. In some applications where the number m of tasks is large, it is more efficient to sample a batch $\mathcal{B}$ of i.i.d. tasks from $\{\mathcal{T}_i, i=1,...,m\}$ at each meta (outer) iteration, and optimizes the mini-batch versions $\mathcal{L}_{\mathcal{D}}(\phi, \widetilde{w}; \mathcal{B}) = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \mathcal{L}_{\mathcal{D}_i}(\phi, w_i)$ and $\mathcal{L}_{\mathcal{S}}(\phi, \widetilde{w}; \mathcal{B}) = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \mathcal{L}_{\mathcal{S}_i}(\phi, w_i)$ instead.

We next provide the convergence result of ITD-BiO for this case, and that of AID-BiO can be similarly derived.

Theorem 4. *Suppose Assumptions 1, 2 and 3 hold and suppose each task loss $\mathcal{L}_{\mathcal{S}_i}(\phi,\cdot)$ is μ-strongly-convex. Choose the same parameters β, D as in Theorem 2. Then, we have*

$$\frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}\|\nabla\Phi(\phi_k)\|^2 \leq \mathcal{O}\left(\frac{1}{K} + \frac{\kappa^2}{|\mathcal{B}|}\right).$$

Theorem 4 shows that compared to the full batch case (i.e., without task sampling) in eq. (10), task sampling introduces a variance term $\mathcal{O}(\frac{1}{|\mathcal{B}|})$ due to the stochastic nature of the algorithm.

5.1. Experiments

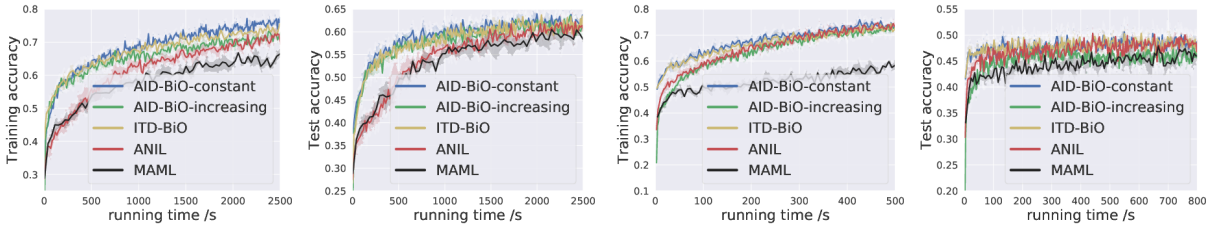
To validate our theoretical results for deterministic bilevel optimization, we compare the performance among the following four algorithms: ITD-BiO, AID-BiO-constant (AID-BiO with a constant number of inner-loop steps as in our analysis), AID-BiO-increasing (AID-BiO with an increasing number of inner-loop steps under analysis in Ghadimi & Wang 2018), and two popular meta-learning algorithms MAML[2] (Finn et al., 2017) and ANIL[3] (Raghu et al., 2019). We conduct experiments over a 5-way 5-shot task on two datasets: FC100 and miniImageNet. The results are averaged over 10 trials with different random seeds. Due to the space limitations, we provide the model architectures and hyperparameter settings in Appendix A.

It can be seen from Figure 2 that for both the miniImageNet and FC100 datasets, AID-BiO-constant converges faster than AID-BiO-increasing in terms of both the training accuracy and test accuracy, and achieves a better final test accuracy than ANIL and MAML. This demonstrates the superior improvement of our developed analysis over existing analysis in Ghadimi & Wang 2018 for AID-BiO algorithm. Moreover, it can be observed that AID-BiO is slightly faster than ITD-BiO in terms of the training accuracy and test accuracy. This is in consistence with our theoretical results.

We also compare the robustness between the bilevel optimizer ITD-BiO (AID-BiO performs similarly to ITD-BiO in terms of the convergence rate) and ANIL when the number T (i.e., D in Algorithm 1) of inner-loop steps is relatively large. It can be seen from Figure 3 that when the number

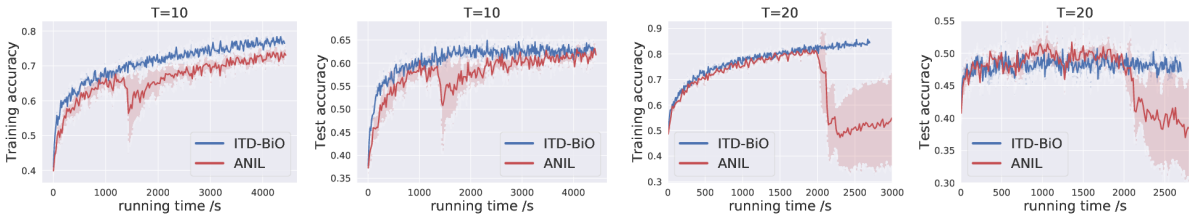
[2]MAML consists of an inner loop for task adaptation and an outer loop for meta initialization training.

[3]ANIL refers to almost no inner loop, which is an efficient MAML variant with task adaption on the last-layer of parameters.



(a) dataset: miniImageNet (b) dataset: FC100

Figure 2. Comparison of various bilevel algorithms on meta-learning. For each dataset, left plot: training accuracy v.s. running time; right plot: test accuracy v.s. running time.



(a) $T = 10$, miniImageNet dataset (b) $T = 20$, FC100 dataset

Figure 3. Comparison of ITD-BiO and ANIL with a relatively large inner-loop iteration number T.

of inner-loop steps is large, i.e., $T = 10$ for miniImageNet and $T = 20$ for FC100, the bilevel optimizer ITD-BiO converges stably with a small variance, whereas ANIL suffers from a sudden descent at 1500s on miniImageNet and even diverges after 2000s on FC100.

6. Applications to Hyperparameter Optimization

The goal of hyperparameter optimization (Franceschi et al., 2018; Feurer & Hutter, 2019) is to search for representation or regularization parameters λ to minimize the validation error evaluated over the learner's parameters w^*, where w^* is the minimizer of the inner-loop regularized training error. Mathematically, the objective function is given by

$$\min_{\lambda} \mathcal{L}_{\mathcal{D}_{\text{val}}}(\lambda) = \frac{1}{|\mathcal{D}_{\text{val}}|} \sum_{\xi \in \mathcal{D}_{\text{val}}} \mathcal{L}(w^*; \xi)$$
$$\text{s.t. } w^* = \arg\min_{w} \underbrace{\frac{1}{|\mathcal{D}_{\text{tr}}|} \sum_{\xi \in \mathcal{D}_{\text{tr}}} \big(\mathcal{L}(w, \lambda; \xi) + \mathcal{R}_{w,\lambda}\big)}_{\mathcal{L}_{\mathcal{D}_{\text{tr}}}(w,\lambda)}, \quad (11)$$

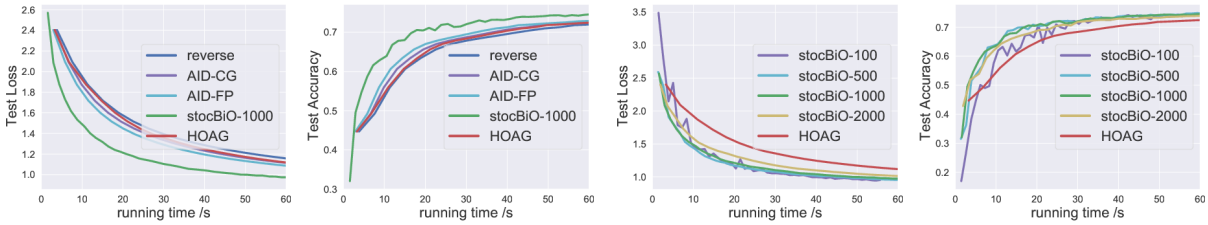
where $\mathcal{D}_{\text{val}}$ and $\mathcal{D}_{\text{tr}}$ are validation and training data, $\mathcal{L}$ is the loss, and $\mathcal{R}_{w,\lambda}$ is a regularizer. In practice, the lower-level function $\mathcal{L}_{\mathcal{D}_{\text{tr}}}(w, \lambda)$ is often strongly-convex w.r.t. w. For example, for the data hyper-cleaning application proposed by Franceschi et al. 2018; Shaban et al. 2019, the predictor is modeled by a linear classifier, and the loss function $\mathcal{L}(w; \xi)$ is convex w.r.t. w and $\mathcal{R}_{w,\lambda}$ is a strongly-convex regularizer, e.g., L^2 regularization. The sample sizes of $\mathcal{D}_{\text{val}}$ and $\mathcal{D}_{\text{tr}}$ are often large, and stochastic algorithms are preferred for achieving better efficiency. As a result, the above hyperparameter optimization falls into the stochastic bilevel optimization we study in eq. (2), and we can apply the proposed stocBiO here. Furthermore, Theorem 3 establishes its performance guarantee.

6.1. Experiments

We compare our proposed **stocBiO** with the following baseline bilevel optimization algorithms.

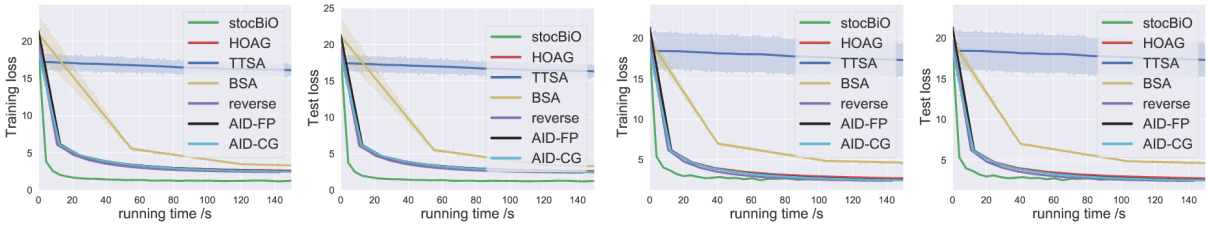
- **BSA** (Ghadimi & Wang, 2018): implicit gradient based stochastic bilevel optimizer via single-sample sampling.
- **TTSA** (Hong et al., 2020): two-time-scale stochastic optimizer via single-sample data sampling.
- **HOAG** (Pedregosa, 2016): a hyperparameter optimization algorithm with approximate gradient. We use the implementation in the repository `https://github.com/fabianp/hoag`.
- **reverse** (Franceschi et al., 2017): an iterative differentiation based method that approximates the hypergradient via backpropagation. We use its implementation in `https://github.com/prolearner/hypertorch`.
- **AID-FP** (Grazzi et al., 2020): AID with the fixed-point method. We use its implementation in `https://github.com/prolearner/hypertorch`
- **AID-CG** (Grazzi et al., 2020): AID with the conjugate gradient method. We use its implementation in `https://github.com/prolearner/hypertorch`.

We demonstrate the effectiveness of the proposed stocBiO algorithm on two experiments: data hyper-cleaning and



(a) Test loss and test accuracy v.s. running time (b) Convergence rate with different batch sizes

Figure 4. Comparison of various stochastic bilevel algorithms on logistic regression on 20 Newsgroup dataset.



(a) Corruption rate $p = 0.1$ (b) Corruption rate $p = 0.4$

Figure 5. Comparison of various stochastic bilevel algorithms on hyperparameter optimization at different corruption rates. For each corruption rate p, left plot: training loss v.s. running time; right plot: test loss v.s. running time.

logistic regression. Due to the space limitations, we provide the details of the objective functions and hyperparameter settings in Appendix B.

Logistic Regression on 20 Newsgroup: As shown in Figure 4(a), the proposed stocBiO achieves the fastest convergence rate as well as the best test accuracy among all comparison algorithms. This demonstrates the practical advantage of our proposed algorithm stocBiO. Note that we do not include BSA and TTSA in the comparison, because they converge too slowly with a large variance, and are much worse than the other competing algorithms. In addition, we investigate the impact of the batch size on the performance of our stocBiO in Figure 4(b). It can be seen that stocBiO outperforms HOAG under the batch sizes of $100, 500, 1000, 2000$. This shows that the performance of stocBiO is not very sensitive to the batch size, and hence the tuning of the batch size is easy to handle in practice.

Data Hyper-Cleaning on MNIST. It can be seen from Figures 5 and 6 that our proposed stocBiO algorithm achieves the fastest convergence rate among all competing algorithms in terms of both the training loss and the test loss. It is also observed that such an improvement is more significant when the corruption rate p is smaller. We note that the stochastic algorithm TTSA converges very slowly with a large variance. This is because TTSA updates the costly outer loop more frequently than other algorithms, and has a larger variance due to the single-sample data sampling. As a comparison, our stocBiO has a much smaller variance for hypergradient estimation as well as a much faster convergence rate. This validates our theoretical results in Theorem 3.

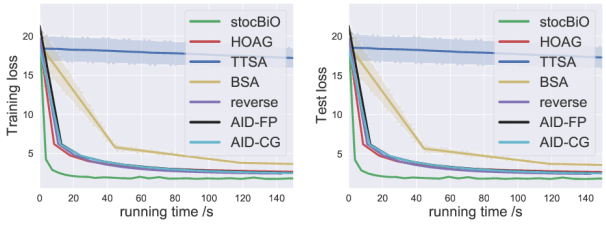


Figure 6. Convergence of algorithms at corruption rate $p = 0.2$.

7. Conclusion

In this paper, we develop a general and enhanced convergence rate analysis for the nonconvex-strongly-convex bilevel deterministic optimization, and propose a novel algorithm for the stochastic setting and show that its computational complexity outperforms the best known results orderwisely. Our results also provide the theoretical guarantee for various bilevel optimizers in meta-learning and hyperparameter optimization. Our experiments validate our theoretical results and demonstrate the superior performance of the proposed algorithm. We anticipate that the convergence rate analysis that we develop will be useful for analyzing other bilevel optimization problems with different loss geometries, and the proposed algorithms will be useful for other applications such as reinforcement learning and Stackelberg game.

Acknowledgements

The work was supported in part by the U.S. National Science Foundation under the grants CCF-1909291 and CCF-1900145.

References

Arnold, S. M., Mahajan, P., Datta, D., and Bunner, I. *learn2learn*, 2019. https://github.com/learnables/learn2learn.

Bertinetto, L., Henriques, J. F., Torr, P., and Vedaldi, A. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations (ICLR)*, 2018.

Bracken, J. and McGill, J. T. Mathematical programs with optimization problems in the constraints. *Operations Research*, 21(1):37–44, 1973.

Domke, J. Generic methods for optimization-based modeling. In *Artificial Intelligence and Statistics (AISTATS)*, pp. 318–326, 2012.

Feurer, M. and Hutter, F. Hyperparameter optimization. In *Automated Machine Learning*, pp. 3–33. Springer, Cham, 2019.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. International Conference on Machine Learning (ICML)*, pp. 1126–1135, 2017.

Flamary, R., Rakotomamonjy, A., and Gasso, G. Learning constrained task similarities in graphregularized multi-task learning. *Regularization, Optimization, Kernels, and Support Vector Machines*, pp. 103, 2014.

Franceschi, L., Donini, M., Frasconi, P., and Pontil, M. Forward and reverse gradient-based hyperparameter optimization. In *International Conference on Machine Learning (ICML)*, pp. 1165–1173, 2017.

Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., and Pontil, M. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning (ICML)*, pp. 1568–1577, 2018.

Ghadimi, S. and Wang, M. Approximation methods for bilevel programming. *arXiv preprint arXiv:1802.02246*, 2018.

Gould, S., Fernando, B., Cherian, A., Anderson, P., Cruz, R. S., and Guo, E. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*, 2016.

Grazzi, R., Franceschi, L., Pontil, M., and Salzo, S. On the iteration complexity of hypergradient computation. In *Proc. International Conference on Machine Learning (ICML)*, 2020.

Hansen, P., Jaumard, B., and Savard, G. New branch-and-bound rules for linear bilevel programming. *SIAM Journal on Scientific and Statistical Computing*, 13(5):1194–1217, 1992.

Hong, M., Wai, H.-T., Wang, Z., and Yang, Z. A two-timescale framework for bilevel optimization: Complexity analysis and application to actor-critic. *arXiv preprint arXiv:2007.05170*, 2020.

Ji, K. and Liang, Y. Lower bounds and accelerated algorithms for bilevel optimization. *arXiv preprint arXiv:2102.03926*, 2021.

Ji, K., Lee, J. D., Liang, Y., and Poor, H. V. Convergence of meta-learning with task-specific adaptation over partial parameters. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020a.

Ji, K., Yang, J., and Liang, Y. Multi-step model-agnostic meta-learning: Convergence and improved algorithms. *arXiv preprint arXiv:2002.07836*, 2020b.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2014.

Konda, V. R. and Tsitsiklis, J. N. Actor-critic algorithms. In *Advances in neural information processing systems (NeurIPS)*, pp. 1008–1014, 2000.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. 2009.

Kunapuli, G., Bennett, K. P., Hu, J., and Pang, J.-S. Classification model selection via bilevel programming. *Optimization Methods & Software*, 23(4):475–489, 2008.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, J., Gu, B., and Huang, H. Improved bilevel model: Fast and optimal algorithm with theoretical guarantee. *arXiv preprint arXiv:2009.00690*, 2020.

Liao, R., Xiong, Y., Fetaya, E., Zhang, L., Yoon, K., Pitkow, X., Urtasun, R., and Zemel, R. Reviving and improving recurrent back-propagation. In *Proc. International Conference on Machine Learning (ICML)*, 2018.

Liu, R., Mu, P., Yuan, X., Zeng, S., and Zhang, J. A generic first-order algorithmic framework for bi-level programming beyond lower-level singleton. In *International Conference on Machine Learning (ICML)*, 2020.

Lorraine, J., Vicol, P., and Duvenaud, D. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1540–1552. PMLR, 2020.

Maclaurin, D., Duvenaud, D., and Adams, R. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning (ICML)*, pp. 2113–2122, 2015.

Moore, G. M. *Bilevel programming algorithms for machine learning model selection*. Rensselaer Polytechnic Institute, 2010.

Okuno, T., Takeda, A., and Kawana, A. Hyperparameter learning via bilevel nonsmooth optimization. *arXiv preprint arXiv:1806.01520*, 2018.

Oreshkin, B., López, P. R., and Lacoste, A. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 721–731, 2018.

Pedregosa, F. Hyperparameter optimization with approximate gradient. In *International Conference on Machine Learning (ICML)*, pp. 737–746, 2016.

Raghu, A., Raghu, M., Bengio, S., and Vinyals, O. Rapid learning or feature reuse? towards understanding the effectiveness of MAML. *International Conference on Learning Representations (ICLR)*, 2019.

Rajeswaran, A., Finn, C., Kakade, S. M., and Levine, S. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 113–124, 2019.

Roth, A., Ullman, J., and Wu, Z. S. Watch and learn: Optimizing from revealed preferences feedback. In *Annual ACM Symposium on Theory of Computing (STOC)*, pp. 949–962, 2016.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 3(115):211–252, 2015.

Shaban, A., Cheng, C.-A., Hatch, N., and Boots, B. Truncated back-propagation for bilevel optimization. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1723–1732, 2019.

Shi, C., Lu, J., and Zhang, G. An extended kuhn–tucker approach for linear bilevel programming. *Applied Mathematics and Computation*, 162(1):51–63, 2005.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Vinyals, O., Blundell, C., Lillicrap, T., and Wierstra, D. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Yu, T. and Zhu, H. Hyper-parameter optimization: A review of algorithms and applications. *arXiv preprint arXiv:2003.05689*, 2020.

Zügner, D. and Günnemann, S. Adversarial attacks on graph neural networks via meta learning. In *International Conference on Learning Representations (ICLR)*, 2019.