PROVABLY FASTER ALGORITHMS FOR BILEVEL OPTIMIZATION AND APPLICATIONS TO META-LEARNING

Anonymous authors

Paper under double-blind review

ABSTRACT

Bilevel optimization has arisen as a powerful tool for many machine learning problems such as meta-learning, hyper-parameter optimization, reinforcement learning, etc. In this paper, we investigate the nonconvex-strongly-convex bilevel optimization problem, and propose two novel algorithms named deterBiO and stocBiO respectively for the deterministic and stochastic settings. At the core design of deterBiO is the construction of a low-cost and easy-to-implement hypergradient estimator via a simple back-propagation. In addition, stocBiO updates with the mini-batch data sampling rather than the existing single-sample schemes, where a sample-efficient Hessian inverse estimator is proposed. We provide the finite-time convergence guarantee for both algorithms, and show that they outperform the best known computational complexities orderwisely with respect to the condition number κ and/or the target accuracy ϵ. We further demonstrate the superior efficiency of the proposed algorithms by the experiments on meta-learning and hyper-parameter 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), hyper-parameter 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.

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

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 $\Phi(x)$ 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 with respect to (w.r.t.) y, and the upper-level objective function $\Phi(x)$ is nonconvex w.r.t. x. Such types of geometrics commonly exist in many applications including meta-learning and hyper-parameter 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 such types of bilevel optimization problems, which inlcude constraint-based methods (Hansen et al., 1992; Shi et al., 2005; Moore, 2010), implicit gradient based methods (Pedregosa, 2016; Gould et al., 2016; Ghadimi & Wang, 2018), and dynamic system based methods (Maclaurin et al., 2015; Franceschi et al., 2017; 2018; Shaban et al., 2019). Most of these works have focused on the asymptotic convergence analysis except that Ghadimi & Wang (2018) provided the finite-time convergence analysis for their proposed bilevel approximation (BA) method. However, to estimate the hyper-gradient $\nabla\Phi(x)$, BA requires an exact computation of a *Hessian inverse* and high-order partial derivatives, which is often computationally expensive and and hard to implement in deep learning applications such as meta-learning.

- Thus, the first focus of this paper is to design a new bilevel optimization algorithm, which achieves better efficiency in practice, is easy to implement in deep learning, and more importantly, has the finite-time performance guarantee with order-level *lower* computational complexity.

The stochastic bilevel optimization often occurs in applications where fresh data need to be sampled as the algorithms run (e.g., reinforcement learning (Hong et al., 2020)) or the sample size of training data is large (e.g., hyper-parameter optimization (Franceschi et al., 2018), Stackelberg game (Roth et al., 2016)). Typically, the corresponding objective function is given by

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

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, the algorithms sample data batch 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), 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 an order-level lower computational complexity over BSA and TTSA.

1.1 Main Contributions

Our main contributions lie in proposing faster algorithms for the nonconvex-strongly-convex bilevel deterministic and stochastic optimization problems with provably order-level lower computational complexity. Our analysis involves several new developments, which can be of independent interest.

We fist propose a deterministic bilevel optimizer (deterBiO) for solving the problem in eq. (1). Unlike the BA (Ghadimi & Wang, 2018) method that requires an exact Hessian inverse computation, our method constructs a hyper-gradient estimator (which computes Hessian rather than Hessian inverse) via a simple back-propagation over the inner loop, and is easy to implement in deep learning applications. As shown in Table 1, the gradient complexities of our algorithm w.r.t. f and g to attain an ϵ-accurate stationary point improve those of BA (Ghadimi & Wang, 2018) by an order of κ and $\kappa\epsilon^{-1/4}$, respectively, where κ is the condition number. For the hyper-gradient estimation, our algorithm needs totally $\widetilde{\mathcal{O}}(\kappa^4\epsilon^{-1})$ Hessian computations, whereas BA requires the same number of Hessian inverse computations (where each Hessian inverse is much more costly than each Hessian computationally). Technically, our analysis develops a novel characterization on the convergence rate of the back-propagation based hyper-gradient estimator.

We then propose a stochastic bilevel optimizer (stocBiO) to solve the problem eq. (2). Our algorithm features a *mini-batch* hyper-gradient estimation via implicit differentiation, where the core design involves a warm-start initialization for the inner loop and a sample-efficient mini-batch Hessian inverse estimator. 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 second-order derivative complexity $\text{Hxy}(G, \epsilon)$ of our algorithm improves that of BSA by an order of κ. In terms of the accuracy ϵ, our gradient and second-order derivative complexities improve those of TTSA (Hong et al., 2020) by an order of $\epsilon^{-0.5}$ and $\epsilon^{-0.5}$, respectively.

We further provide the theoretical complexity guarantee of deterBiO and stocBiO in meta-learning and hyper-parameter optimization, and demonstrate their superior efficiency by experiments. Due to the space limitations, we present all theoretical and empirical results on hyper-parameter optimization in the supplementary materials.

1.2 Related Work

Bilevel optimization approaches: Bilevel optimization was first introduced by Bracken & McGill (1973). Since then, quite a few bilevel optimization algorithms have been proposed, which include but not limited to constraint-based methods (Shi et al., 2005; Moore, 2010), gradient-based methods (Pedregosa, 2016; Gould et al., 2016; Maclaurin et al., 2015; Franceschi et al., 2018; Ghadimi

Table 1: Comparison of bilevel deterministic optimization algorithms.

Algorithm	Gc(f,ϵ)	Gc(g,ϵ)	Hxy(g,ϵ)	Hyy(g,ϵ)/Hyy$^{\text{inv}}(g,\epsilon)$ *
BA (Ghadimi & Wang, 2018)	$\mathcal{O}(\kappa^4\epsilon^{-1})$	$\mathcal{O}(\kappa^5\epsilon^{-5/4})$	$\mathcal{O}\left(\kappa^4\epsilon^{-1}\right)$	$\mathcal{O}\left(\kappa^4\epsilon^{-1}\right)$ Hessian inverse
deterBiO (ours)	$\mathcal{O}(\kappa^3\epsilon^{-1})$	$\widetilde{\mathcal{O}}(\kappa^4\epsilon^{-1})$	$\widetilde{\mathcal{O}}\left(\kappa^4\epsilon^{-1}\right)$	$\widetilde{\mathcal{O}}\left(\kappa^4\epsilon^{-1}\right)$ Hessian

Gc(f,ϵ) and Gc(g,ϵ): number of gradient evaluations w.r.t. f and g. κ : the condition number.
Hxy(g,ϵ): number of second-order derivative $\nabla_x\nabla_y g(x,y)$. Notation $\widetilde{\mathcal{O}}$: omit $\log\frac{1}{\epsilon}$ terms.
Hyy(g,ϵ) and Hyy$^{\text{inv}}(g,\epsilon)$: number of evaluations of **Hessian** $\nabla_y^2 g$ and **Hessian inverse** $[\nabla_y^2 g]^{-1}$.
* In last column, each Hessian inverse is much more costly than Hessian computationally.

Table 2: Comparison of bilevel stochastic optimization algorithms.

Algorithm	Gc(F,ϵ)	Gc(G,ϵ)	Hxy(G,ϵ)	Hyy(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 (ours)	$\mathcal{O}(\kappa^5\epsilon^{-2})$	$\mathcal{O}(\kappa^9\epsilon^{-2})$	$\mathcal{O}\left(\kappa^5\epsilon^{-2}\right)$	$\widetilde{\mathcal{O}}\left(\kappa^6\epsilon^{-2}\right)$

* We use poly(κ) because Hong et al. (2020) do not provide the dependences on κ.

& Wang, 2018; Shaban et al., 2019; Hong et al., 2020; Liu et al., 2020; Li et al., 2020), etc. Among them, Ghadimi & Wang (2018); Hong et al. (2020) provided the finite-time complexity analysis for their proposed methods for the nonconvex-strongly-convex bilevel optimization problem. For such a problem, this paper proposes two novel algorithms respectively for the deterministic and stochastic settings with order-level lower computational complexity than the existing results.

Some works have studied other types of loss geometries. 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 employed in 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 a bilevel optimization problem, and proposed iMAML via implicit gradient. This paper demonstrates the better efficiency of deterBiO than the existing representative methods in meta-learning.

Bilevel optimization in hyper-parameter optimization: Hyper-parameter 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 in the context of hyper-parameter 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. This paper demonstrates the superior efficiency of the proposed stocBiO algorithm in hyper-parameter optimization.

2 Proposed Algorithms

In this section, we propose two algorithms respectively for the deterministic and stochastic bilevel optimization problems, and describe the key design components involved in these algorithms.

2.1 Algorithm for Deterministic Bilevel Optimization

In order to efficiently solve the deterministic bilevel objective problem eq. (1), we propose a deterministic bilevel optimizer (deterBiO) as given in Algorithm 1.

deterBiO updates in a nested-loop manner. At the beginning of each outer loop, the inner loop runs T steps of gradient decent (GD) to find an approximation point y_k^T close to $y^*(x_k)$. At the outer loop, deterBiO computes the gradient $\frac{\partial f(x_k, y_k^T(x_k))}{\partial x_k}$ as an approximation of the hyper-gradient

Algorithm 1 Deterministic bilevel optimizer (deterBiO)

Input: lower- and upper-level stepsizes $\alpha, \beta > 0$, initializations x_0 and $\{y_k^0, k = 0, ..., K\}$.
for $k = 0, 1, 2, ..., K$ **do**
  for $t = 1, ..., T$ **do**
    Update $y_k^t = y_k^{t-1} - \alpha \nabla_y g(x_k, y_k^{t-1})$
  end for
  Compute gradient $\frac{\partial f(x_k, y_k^T)}{x_k}$ via back-propagation w.r.t. x_k
  Update $x_{k+1} = x_k - \beta \frac{\partial f(x_k, y_k^T)}{\partial x_k}$
end for

$\nabla \Phi(x_k) = \frac{\partial f(x_k, y^*(x_k))}{\partial x_k}$ via back-propagation, where we write $y_k^T(x_k)$ because the output y_k^T of the inner loop has a dependence on x_k through the inner-loop iterative GD updates, and x_k and y_k^0 are treated to be mutually-independent during the differentiation process at the kth outer loop. Our design is easy to implement in deep learning applications such as meta-learning via the automatic differentiation package. The explicit form of the estimate $\frac{\partial f(x_k, y_k^T(x_k))}{\partial x_k}$ is given by the following proposition via the chain rule. Let $\prod_{j=T}^{T-1}(\cdot) = I$ for notation simplification.

Proposition 1. *The gradient* $\frac{\partial f(x_k, y_k^T(x_k))}{\partial x_k}$ *takes the following analytical form:*

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

Proposition 1 shows that the differentiation involves computations of second-order derivatives such as Hessian $\nabla_y^2 g(\cdot, \cdot)$. Since efficient Hessian-free methods, e.g., conjugate-gradient (CG), have been successfully deployed in the existing automatic differentiation tools, the computations of second-order derivatives are efficient and scalable to the problem dimension. As a comparison, Ghadimi & Wang (2018) constructs the outer gradient using the implicit differentiation formulation, and requires to compute Hessian inverse $(\nabla_y^2 g)^{-1}$ exactly and explicitly, which is harder to implement and much more costly than Hessian computation in our algorithm in deep learning applications.

2.2 Algorithm for Stochastic Bilevel Optimization

We propose a stochastic bilevel optimizer (stocBiO) in Algorithm 2 to solve the problem eq. (2). It has a double-loop structure similar to deterBiO, but runs T steps of stochastic gradient decent (SGD) at the inner loop to obtain an approximated solution y_k^T. Note that we choose the initialization y_k^0 of each inner loop as the output y_{k-1}^T of the preceding inner loop rather than a random start as in Ghadimi & Wang (2018). Such a warm start allows us to back-propagate the tracking error $\|y_k^T - y^*(x_k)\|$ to previous loops, and results in an improved computational complexity. Based on the output y_k^T of the inner loop, stocBiO then constructs an outer-loop gradient estimate

$$\widehat{\nabla} \Phi(x_k) = \nabla_x F(x_k, y_k^T; \mathcal{D}_F) - \nabla_x \nabla_y G(x_k, y_k^T; \mathcal{D}_G) H^{\text{inv}}(x_k, y_k^T; \mathcal{D}_H) \nabla_y F(x_k, y_k^T; \mathcal{D}_F) \quad (3)$$

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

Proposition 2. *Recalling the definition* $\Phi(x) := f(x, y^*(x))$*, it holds that*

$$\nabla \Phi(x_k) = \nabla_x f(x_k, y^*(x_k)) - \nabla_x \nabla_y g(x_k, y^*(x_k)) \big[\nabla_y^2 g(x_k, y^*(x_k))\big]^{-1} \nabla_y f(x_k, y^*(x_k)). \quad (4)$$

An important component of our algorithm is the Hessian inverse estimator $H^{\text{inv}}(\cdot)$ in eq. (3) . Compared to the deterministic case, designing a sample-efficient Hessian inverse estimator in the stochastic case is more challenging. Using the Taylor expansion, we construct a mini-batch Hessian inverse estimator in Algorithm 3, which takes the form of

$$\text{(Hessian inverse estimator):} \quad H^{\text{inv}}(x_k, y_k^T; \mathcal{D}_H) = \eta \sum_{q=0}^{Q} \prod_{j=1}^{q} (I - \eta \nabla_y^2 G(x_k, y_k^T; \mathcal{B}_j)), \quad (5)$$

where $\{\mathcal{B}_j, j = 1, ..., Q\}$ are mutually-independent sample sets, and Q and η are constants. Instead of choosing the same batch sizes for all $\mathcal{B}_j, j = 1, ..., Q$ in eq. (5), our analysis captures the different

Algorithm 2 Stochastic bilevel optimizer (stocBiO)

Input: Stepsizes $\alpha, \beta > 0$, initializations x_0 and y_0.
for $k = 0, 1, 2, ..., K$ **do**
  Set $y_k^0 = y_{k-1}^T$ if $k > 0$ and y_0 otherwise
  for $t = 1, ..., T$ **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 $\mathcal{D}_H, \mathcal{D}_G, \mathcal{D}_F$
  Construct Hessian inverse estimator via Algorithm 3
  Compute gradient estimate $\widehat{\nabla}\Phi(x_k)$ via eq. (3)
  Update $x_{k+1} = x_k - \beta \widehat{\nabla}\Phi(x_k)$
end for

Algorithm 3 Construct Hessian inverse estimator $H^{\text{inv}}(x, y; \mathcal{D}_H)$

Input: An integer Q, $\mathcal{D}_H = \{\mathcal{B}_j\}_{j=1}^Q$, a constant $\eta > 0$.
for $j = 1, 2, ..., Q$ **do**
  Draw a sample batch $\mathcal{B}_j$
  Compute Hessian $\nabla_y^2 G(x, y; \mathcal{B}_j)$
end for
Construct $H^{\text{inv}}(x, y; \mathcal{D}_H)$ via eq. (5)

impact of components $\nabla_y^2 G(x_k, y_k^T; \mathcal{B}_j), j = 1, ..., Q$ on the Hessian inverse estimation variance, and inspires an adaptive and more efficient choice by setting $|\mathcal{B}_j|$ to decay *exponentially* with j. By doing so, we achieve an improved gradient and Hessian complexity.

3 Definitions and Assumptions

Let $z = (x, y)$ denote all parameters. For simplicity, suppose sample sets $\mathcal{S}_t$ for all $t = 0, ..., T-1$, $\mathcal{D}_G$ 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 the objective function $\Phi(x)$ 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 $\mathbb{E}\|\nabla\Phi(\bar{x})\|^2 \leq \epsilon$, 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 computational complexity.

Definition 2. *For a function $f(x, y)$, let $\text{Gc}(f, \epsilon)$ be the number of the partial gradient $\nabla_x f$ or $\nabla_y f$, $\text{Hyy}(f, \epsilon)$ and $\text{Hyy}^{-1}(f, \epsilon)$ be the number of Hessian $\nabla_x^2 f$ or $\nabla_y^2 f$ and Hessian inverse $(\nabla_x^2 f)^{-1}$ or $(\nabla_y^2 f)^{-1}$, respectively, and $\text{Hxy}(f, \epsilon)$ be the number of second-order derivative $\nabla_x \nabla_y f$. 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*

- *$f(z)$ is M-Lipschitz, i.e., for any z, z', $|f(z) - f(z')| \leq M\|z - z'\|$.*
- *Gradients $\nabla f(z)$ and $\nabla f(z)$ are L-Lipschitz, i.e., for any z, z',*

$$\|\nabla f(z) - \nabla f(z')\| \leq L\|z - z'\|, \ \|\nabla g(z) - \nabla g(z')\| \leq L\|z - z'\|.$$

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

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

4 Main Results for Bilevel Optimization

4.1 Deterministic Bilevel Optimization

We first characterize an important estimation property of the outer-loop gradient estimator $\frac{\partial f(x_k, y_k^T)}{\partial x_k}$ in Algorithm 1 for approximating the true gradient $\nabla \Phi(x_k)$ based on Proposition 1.

Proposition 3. *Suppose Assumptions 1, 2 and 3 hold. Choose $\alpha \leq \frac{1}{L}$. Then, we have*

$$\Big\| \frac{\partial f(x_k, y_k^T)}{\partial x_k} - \nabla \Phi(x_k) \Big\| \leq \Big(\frac{L(L+\mu)(1-\alpha\mu)^{\frac{T}{2}}}{\mu} + \frac{2M(\tau\mu + L\rho)}{\mu^2}(1-\alpha\mu)^{\frac{T-1}{2}} \Big) \|y_k^0 - y^*(x_k)\| + \frac{LM(1-\alpha\mu)^T}{\mu}.$$

Proposition 3 shows that the gradient estimation error $\big\| \frac{\partial f(x_k, y_k^T)}{\partial x_k} - \nabla \Phi(x_k) \big\|$ decays exponentially w.r.t. the number T of the inner-loop steps. The proof of Proposition 3 needs to characterize the rate of the sequence $\big(\frac{\partial y_k^t}{\partial x_k}, t = 0, ..., T \big)$ converging to $\frac{\partial y^*(x_k)}{\partial x_k}$ via the differentiation over all corresponding points along the inner-loop GD path as well as the optimality of the point $y^*(x_k)$.

Based on Proposition 3, we next characterize the convergence and complexity performance of the proposed deterBiO algorithm. Let $\kappa = \frac{L}{\mu}$ denote the condition number.

Theorem 1. *Suppose Assumptions 1, 2 and 3 hold. Define a smoothness parameter $L_\Phi = L + \frac{2L^2+\tau M^2}{\mu} + \frac{\rho LM + L^3 + \tau ML}{\mu^2} + \frac{\rho L^2 M}{\mu^3} = \Theta(\kappa^3)$ and choose the outer stepsize $\beta = \frac{1}{4L_\Phi}$ and $T = \log\Big(\max\big\{ \frac{3LM}{\mu}, 9\Delta L^2 (1+\frac{L}{\mu})^2, \frac{36\Delta M^2(\tau\mu+L\rho)^2}{(1-\alpha\mu)\mu^4} \big\} \frac{9}{2\epsilon} \Big) / \log \frac{1}{1-\alpha\mu} = \Theta(\kappa \log \frac{1}{\epsilon})$. Then, we have*

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

In order to achieve an ϵ-accurate stationary point, we have

- *Gradient complexity:* $\mathrm{Gc}(f,\epsilon) = \mathcal{O}(\kappa^3 \epsilon^{-1}), \mathrm{Gc}(g,\epsilon) = \mathcal{O}(\kappa^4 \epsilon^{-1} \log(\frac{1}{\epsilon}))$.
- *High-order complexity:* $\mathrm{Hxy}(g,\epsilon) = \mathcal{O}\left(\kappa^4 \epsilon^{-1} \log \frac{1}{\epsilon}\right), \mathrm{Hyy}(g,\epsilon) = \mathcal{O}\left(\kappa^4 \epsilon^{-1} \log \frac{1}{\epsilon}\right)$.

It can be seen from Table 1 that the gradient complexity w.r.t. f and g of our algorithm improves that of BA (Ghadimi & Wang, 2018) (eq. (2.30) therein) by an order of κ and $\kappa\epsilon^{-1/4}$. In contrast to BA (Ghadimi & Wang, 2018) that requires computations of Hessian inverse, our algorithm requires only the computations of Hessian (at the same number), which are much more efficient.

4.2 Stochastic Bilevel Optimization

We first characterize the bias and variance of $H^{\mathrm{inv}}(x_k, y_k^T; \mathcal{D}_H)$ in the following proposition.

Proposition 4. *Suppose Assumptions 1, 2 and 3 hold. Let the constant $\eta \leq \frac{1}{L}$ and choose the batch sizes $|\mathcal{B}_j| = BQ(1-\eta\mu)^{j-1}$ for $j = 1, ..., Q$, where $B \geq \frac{1}{Q(1-\eta\mu)^{Q-1}}$. Then, the bias satisfies*

$$\|\mathbb{E}_{\mathcal{D}_H} H^{\mathrm{inv}}(x_k, y_k^T; \mathcal{D}_H) - [\nabla_y^2 g(x_k, y_k^T)]^{-1}\| \leq \mu^{-1}(1-\eta\mu)^Q. \tag{6}$$

Furthermore, the estimation variance is given by

$$\mathbb{E}_{\mathcal{D}_H} \|H^{\mathrm{inv}}(x_k, y_k^T; \mathcal{D}_H) - [\nabla_y^2 g(x_k, y_k^T)]^{-1}\|^2 \leq \frac{2\eta^2 L^2}{\mu^2} \frac{1}{B} + \frac{2(1-\eta\mu)^{2Q+2}}{\mu^2}. \tag{7}$$

Proposition 4 shows that if we choose Q and B at an order level of $\mathcal{O}(\log \frac{1}{\epsilon})$ 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 $|\mathcal{B}_j|$ exponentially decays w.r.t. j. As a 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 provide the convergence and complexity analysis for the proposed stocBiO algorithm for the case where the objective function $\Phi(x) := f(x, y^*(x))$ is nonconvex.

Theorem 2. *Suppose Assumptions 1, 2, 3 and 4 hold. Define parameter* $L_\Phi = L + \frac{2L^2+\tau M^2}{\mu} + \frac{\rho LM+L^3+\tau ML}{\mu^2} + \frac{\rho L^2 M}{\mu^3} = \mathcal{O}\left(\kappa^3\right)$*, and choose stepsize* $\beta = \frac{1}{4L_\Phi}$ *and set* $\eta < \frac{1}{L}$ *in Algorithm 3. Set* $T \geq \max\left\{\frac{\log\left(12+\frac{48\beta^2L^2}{\mu^2}(L+\frac{L^2}{\mu}+\frac{M\tau}{\mu}+\frac{LM\rho}{\mu^2})^2\right)}{2\log(\frac{L+\mu}{L-\mu})}, \frac{\log\left(\sqrt{\beta}(L+\frac{L^2}{\mu}+\frac{M\tau}{\mu}+\frac{LM\rho}{\mu^2})\right)}{\log(\frac{L+\mu}{L-\mu})}\right\}$*. Then, we have*

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\nabla\Phi(x_k)\|^2 \leq \frac{32L_\Phi(\Phi(x_0)-\inf_x\Phi(x)+\frac{5}{2}\|y_0-y^*(x_0)\|^2)}{K} + \frac{52L^2M^2}{\mu^2}(1-\eta\mu)^{2Q}$$
$$+\frac{40\big(L+\frac{L^2}{\mu}+\frac{M\tau}{\mu}+\frac{LM\rho}{\mu^2}\big)^2}{L\mu}\frac{\sigma^2}{S} + \frac{22L^2M^2}{\mu^2 D_g} + \frac{(8\mu^2+22L^2)M^2}{\mu^2 D_f} + \frac{44L^2M^2}{\mu^2 B}. \quad (8)$$

In order to achieve an ϵ*-accurate stationary point, we have*

- *Gradient complexity:* $\mathrm{Gc}(F,\epsilon) = \mathcal{O}(\kappa^5\epsilon^{-2}), \mathrm{Gc}(G,\epsilon) = \mathcal{O}(\kappa^9\epsilon^{-2})$.
- *High-order complexity:* $\mathrm{Hxy}(G,\epsilon) = \mathcal{O}(\kappa^5\epsilon^{-2})$, $\mathrm{Hyy}(G,\epsilon) = \mathcal{O}(\kappa^6\epsilon^{-2}\log\frac{\kappa^2}{\epsilon})$.

Theorem 2 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 total number T of the inner-loop steps is chosen at nearly 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 second-order derivative complexity $\mathrm{Hxy}(G,\epsilon)$ of our algorithm improves that of BSA by an order of κ. In terms of the accuracy ϵ, our gradient and second-order derivative complexities improve those of TTSA in Hong et al. (2020) by an order of $\epsilon^{-0.5}$ and $\epsilon^{-0.5}$, respectively.

5 Applications to Meta-Learning

5.1 Meta-Learning with Common Embedding Model

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 function 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. Specifically, let $\widetilde{w} = (w_1,...,w_m)$ denote all task-specific parameters. Then, the objective function is given by

$$\min_\phi \mathcal{L}_{\mathcal{D}}(\phi,\widetilde{w}^*) := \frac{1}{m}\sum_{i=1}^m \underbrace{\frac{1}{|\mathcal{D}_i|}\sum_{\xi\in\mathcal{D}_i}\mathcal{L}(\phi, w_i^*;\xi)}_{\mathcal{L}_{\mathcal{D}_i}(\phi,w_i^*)\text{: task-specific upper-level loss}}$$
$$\text{s.t. } \widetilde{w}^* = \arg\min_{\widetilde{w}}\mathcal{L}_{\mathcal{S}}(\phi,\widetilde{w}) = \frac{1}{m}\sum_{i=1}^m\Big\{\arg\min_{w_i}\Big(\underbrace{\frac{1}{|\mathcal{S}_i|}\sum_{\xi\in\mathcal{S}_i}\mathcal{L}(\phi,w_i;\xi)+\mathcal{R}(w_i)}_{\mathcal{L}_{\mathcal{S}_i}(\phi,w_i)\text{: task-specific lower-level loss}}\Big)\Big\}, \quad (9)$$

where $\mathcal{S}_i$ and $\mathcal{D}_i$ are the training and test datasets of task $\mathcal{T}_i$, and $\mathcal{R}(w_i)$ is a strongly-convex regularizer, e.g., L^2 regularization. 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, our proposed deterBiO 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. The following theorem provides the convergence analysis of deterBiO for this case.

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

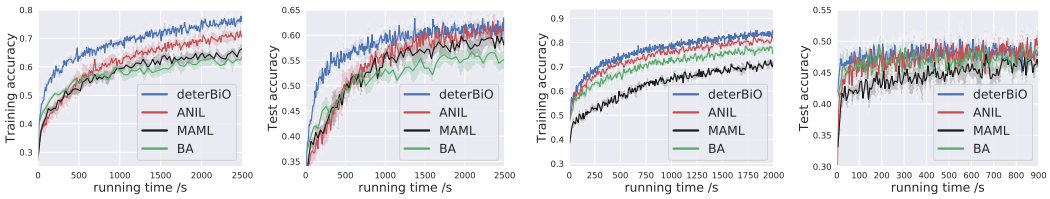
$$\frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}\|\nabla\Phi(\phi_k)\|^2 \leq \frac{16L_\Phi(\Phi(\phi_0) - \inf_\phi \Phi(\phi))}{K} + \frac{2\epsilon}{3} + \Big(1 + \frac{L}{\mu}\Big)^2 \frac{M^2}{8|\mathcal{B}|},$$

where $\Phi(\phi) := \mathcal{L}_{\mathcal{D}}(\phi, \widetilde{w}^*(\phi))$ and the constant L_Φ is given in Theorem 1.

Theorem 3 shows that compared to the full batch (i.e., without task sampling) case in eq. (9), the task sampling introduces a variance term $\mathcal{O}(\frac{1}{|\mathcal{B}|})$ due to the stochastic nature of the algorithm.

5.2 Experiments

We compare the performance among the following four algorithms: deterBiO proposed in this paper, bilevel optimizer BA[1] (Ghadimi & Wang, 2018), and two popular meta-learning algorithms MAML (Finn et al., 2017) and ANIL[2] (Raghu et al., 2019). We conduct experiments over a 5-way 5-shot task on two benchmark datasets: FC100 and miniImageNet, and the results are averaged over 10 trails with different random seeds. Due to the space limitations, we provide the dataset description, model architectures, hyper-parameter settings and additional experiments in Appendix B.



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

Figure 1: Convergence rate of various algorithms on meta-learning. For each dataset, left plot: training accuracy v.s. running time; right plot: test accuracy v.s. running time.

It can be seen from Figure 1 that for both the miniImageNet and FC100 datasets, the proposed deterBiO algorithm converges much faster (nearly 2 times faster in Figure 1(a)) than other competing algorithms in terms of both the training accuracy and test accuracy, and further achieves a better final test accuracy than ANIL and MAML. In addition, our experiments in Appendix B.3 show that deterBiO converges more stably than ANIL when the number of inner-loop steps increases.

6 Conclusion

In this paper, we propose two novel algorithms for the nonconvex-strongly-convex bilevel optimization, and show that their computational complexities outperform the best known results order-wisely. We also provide the theoretical guarantee of the proposed algorithms in meta-learning and hyper-parameter optimization, and demonstrate their effectiveness via experiments. We anticipate that the finite-sum 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.

[1] We use a conjugate-gradient (CG) based approach to estimate the Hessian inverse in BA.

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

REFERENCES

Sebastien M.R. Arnold, Praateek Mahajan, Debajyoti Datta, and Ian Bunner. *learn2learn*, 2019. `https://github.com/learnables/learn2learn`.

Luca Bertinetto, Joao F Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations (ICLR)*, 2018.

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

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

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

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

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

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning (ICML)*, pp. 1568–1577, 2018.

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

Stephen Gould, Basura Fernando, Anoop Cherian, Peter Anderson, Rodrigo Santa Cruz, and Edison Guo. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*, 2016.

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

Mingyi Hong, Hoi-To Wai, Zhaoran Wang, and Zhuoran Yang. A two-timescale framework for bilevel optimization: Complexity analysis and application to actor-critic. *arXiv preprint arXiv:2007.05170*, 2020.

Kaiyi Ji, Jason D Lee, Yingbin Liang, and H Vincent Poor. Convergence of meta-learning with task-specific adaptation over partial parameters. *arXiv preprint arXiv:2006.09486*, 2020a.

Kaiyi Ji, Junjie Yang, and Yingbin Liang. Multi-step model-agnostic meta-learning: Convergence and improved algorithms. *arXiv preprint arXiv:2002.07836*, 2020b.

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

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

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Gautam Kunapuli, Kristin P Bennett, Jing Hu, and Jong-Shi Pang. Classification model selection via bilevel programming. *Optimization Methods & Software*, 23(4):475–489, 2008.

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

Junyi Li, Bin Gu, and Heng Huang. Improved bilevel model: Fast and optimal algorithm with theoretical guarantee. *arXiv preprint arXiv:2009.00690*, 2020.

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 *International Conference on Machine Learning (ICML)*, 2020.

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

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

Takayuki Okuno, Akiko Takeda, and Akihiro Kawana. Hyperparameter learning via bilevel nonsmooth optimization. *arXiv preprint arXiv:1806.01520*, 2018.

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

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

Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of MAML. *International Conference on Learning Representations (ICLR)*, 2019.

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

Aaron Roth, Jonathan Ullman, and Zhiwei Steven Wu. Watch and learn: Optimizing from revealed preferences feedback. In *Annual ACM Symposium on Theory of Computing (STOC)*, pp. 949–962, 2016.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 3(115):211–252, 2015.

Amirreza Shaban, Ching-An Cheng, Nathan Hatch, and Byron Boots. Truncated back-propagation for bilevel optimization. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1723–1732, 2019.

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

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

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

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

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