A Value-Function-based Interior-point Method
for Non-convex Bi-level Optimization

Risheng Liu [1 2 3] **Xuan Liu** [1 2] **Xiaoming Yuan** [4] **Shangzhi Zeng** [4] **Jin Zhang** [5 6]

Abstract

Bi-level optimization model is able to capture a wide range of complex learning tasks with practical interest. Due to the witnessed efficiency in solving bi-level programs, gradient-based methods have gained popularity in the machine learning community. In this work, we propose a new gradient-based solution scheme, namely, the Bi-level Value-Function-based Interior-point Method (BVFIM). Following the main idea of the log-barrier interior-point scheme, we penalize the regularized value function of the lower level problem into the upper level objective. By further solving a sequence of differentiable unconstrained approximation problems, we consequently derive a sequential programming scheme. The numerical advantage of our scheme relies on the fact that, when gradient methods are applied to solve the approximation problem, we successfully avoid computing any expensive Hessian-vector or Jacobian-vector product. We prove the convergence without requiring any convexity assumption on either the upper level or the lower level objective. Experiments demonstrate the efficiency of the proposed BVFIM on non-convex bi-level problems.

1. Introduction

In recent years, with the rapid growth of complexity of machine learning tasks, an increasing number of models with hierarchical structures have arisen from various fields (Dempe, 2018; Liu et al., 2021). In these models, according to different hierarchies, parameters are mainly divided into two types, for instance, hyper-parameters and model parameters in hyper-parameter optimization (Franceschi et al., 2017; Okuno et al., 2018; Mackay et al., 2018), network structures and weights in neural architecture search (Liu et al., 2019a; Liang et al., 2019; Chen et al., 2019), meta learners and base learners in meta learning (Franceschi et al., 2018; Rajeswaran et al., 2019; Zügner & Günnemann, 2019), generators and discriminators in adversarial learning (Pfau & Vinyals, 2016), actors and critics in reinforcement learning (Yang et al., 2019), different components in image processing (Liu et al., 2019b; 2020a;b;c), etc.

Bi-Level Optimizations (BLOs) have been recognized as important tools to capture these machine learning applications with hierarchical structures. Mathematically, BLOs can be formulated as the following optimization problem:

$$\min_{\mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathbb{R}^n} F(\mathbf{x}, \mathbf{y}), \text{ s.t. } \mathbf{y} \in \mathcal{S}(\mathbf{x}), \tag{1}$$

where $\mathcal{S}(\mathbf{x}) := \arg \min_{\mathbf{y}} f(\mathbf{x}, \mathbf{y})$, the UL constraint $\mathcal{X} \subset \mathbb{R}^m$ is a compact set, and the Upper-Level (UL) objective $F : \mathcal{X} \times \mathbb{R}^n \to \mathbb{R}$ and the Lower-Level (LL) objective $f : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$ are continuously differentiable and jointly continuous functions. Indeed, BLOs are by nature a class of hierarchical problem with optimization problems in the constraints. Specifically, aiming to optimize the UL objective F, LL variable $\mathbf{y}$ is selected from the optimal solution set $\mathcal{S}(\mathbf{x})$ of the LL problem governed by UL variable $\mathbf{x}$. Due to the hierarchical structure and nested relationship between UL and LL problems, BLOs are challenging both theoretically and numerically, especially when the LL solution set $\mathcal{S}(\mathbf{x})$ is not singleton (Jeroslow, 1985).

1.1. Related Work

BLOs in Eq. (1) are generalizations of several well-known optimization problems as well-noted. Since BLOs can model highly complicated learning tasks as aforementioned, it is not surprising that they are hard to solve. In early works, the standard methods for solving BLOs are the random search (Bergstra & Bengio, 2012) which evaluates the learning models through randomly sampled parameter values, or the more sophisticated Bayesian optimization (Hutter et al., 2011). Unfortunately, these gradient-free methods

[1]DUT-RU International School of Information Science and Engineering, Dalian University of Technology. [2]Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province. [3]Pazhou Lab, Guangzhou. [4]Department of Mathematics, The University of Hong Kong. [5]Department of Mathematics, Southern University of Science and Technology. [6]National Center for Applied Mathematics Shenzhen.. Correspondence to: Jin Zhang <zhangj9@sustech.edu.cn>.

Table 1. Comparing the theoretical results among BVFIM and existing gradient-based methods. $C^1(\mathcal{X} \times \mathbb{R}^n)$ denotes the set of all continuously differentiable functions on $\mathcal{X} \times \mathbb{R}^n$.

Category	Methods	LLC	LLS		Required Conditions
EGBM	FHG/RHG	w/	w/	UL	$F(\mathbf{x}, \mathbf{y}) \in C^1(\mathcal{X} \times \mathbb{R}^n)$.
				LL	$f(\mathbf{x}, \mathbf{y}) \in C^1(\mathbb{R}^m \times \mathbb{R}^n)$.
	TRHG	w/	w/	UL	$F(\mathbf{x}, \mathbf{y}) \in C^1(\mathcal{X} \times \mathbb{R}^n)$ is bounded below.
				LL	$f(\mathbf{x}, \mathbf{y}) \in C^1(\mathbb{R}^m \times \mathbb{R}^n)$ is L_f-smooth and strongly convex.
	BDA	w/	w/o	UL	$F(\mathbf{x}, \mathbf{y}) \in C^1(\mathcal{X} \times \mathbb{R}^n)$ is strongly convex.
				LL	$f(\mathbf{x}, \mathbf{y}) \in C^1(\mathbb{R}^m \times \mathbb{R}^n)$ is level-bounded in $\mathbf{y}$.
IGBM	CG/Neumann	w/	w/	UL	$F(\mathbf{x}, \mathbf{y}) \in C^1(\mathcal{X} \times \mathbb{R}^n)$.
				LL	$f(\mathbf{x}, \mathbf{y}) \in C^1(\mathbb{R}^m \times \mathbb{R}^n)$, and $\frac{\partial^2 f(\mathbf{x}, \mathbf{y})}{\partial \mathbf{y}^2}$ is invertible.
IPM	BVFIM	w/o	w/o	UL	$F(\mathbf{x}, \mathbf{y}) \in C^1(\mathcal{X} \times \mathbb{R}^n)$.
				LL	$f(\mathbf{x}, \mathbf{y}) \in C^1(\mathbb{R}^m \times \mathbb{R}^n)$.

are only capable of handling more than twenty or so hyper-parameters. Gradient-based optimization methods, on the other hand, can handle up to hundreds of hyper-parameters. Specifically, the gradient-based approaches transform the BLO into a Single-Level Optimization (SLO) problem and solve it by using gradient-type methods. To reformulate the BLO into an SLO problem and calculate the gradients, one can replace the LL with a dynamics iteration and derive the gradient by Automatic Differentiation (AD), or apply the implicit function theorem to the optimality conditions of the LL problem. According to such different features of calculations, these gradient-based methods for BLOs can be classified into two main categories: explicit and implicit. For theoretical support, they both rely on the singleton of LL solution set $\mathcal{S}(\mathbf{x})$ (a.k.a. LLS), which is restrictive in applications. For numerical implementation, they both require repeated products of vectors and matrices, which are extraordinarily costly in computation power. In response to these two limitations, by reformulating the BLO into an SLO problem via value function approach (Outrata, 1990; Ye & Zhu, 1995), using the smoothing technique on value function via regularization, and penalizing the smoothed value function to the UL objective via the log-barrier penalty, this paper proposes a value-function-based interior-point gradient-type method.

Explicit Gradient-Based Methods (EGBMs). The key idea underlying this type of approaches is to hierarchically calculate gradients of UL and LL objectives. Specifically, under the LLS and Lower Level Convexity (LLC), the works in (Franceschi et al., 2017; 2018) first calculate gradient representations of the LL objective and then perform either reverse or forward gradient computations (termed as Reverse Hyper-Gradient (RHG) and Forward Hyper-Gradient (FHG)) for the UL sub-problem. In order to address the LLS restriction, under the LLC, (Liu et al., 2020d) proposes Bilevel Descent Aggregation (BDA) which characterizes an aggregation of both the LL and the UL descent information.

However, since the EGBMs method requires calculating AD for the entire trajectory of the dynamic iteration of LL, the computation load is heavy to calculate the gradient with reasonable preciseness. In order to reduce the amount of computation, (Shaban et al., 2019) proposes Truncated Reverse Hyper-Gradient (TRHG) to truncate the gradient trajectory. However, the efficiency of TRHG is sensitive to the truncated path length. Small truncated path length may deteriorate the accuracy of the calculated gradient and large truncated path length cannot reduce the computation cost. Although (Liu et al., 2019a) uses the difference of vectors to approximate gradient, there is no theoretical justification for such type of approximation.

Implicit Gradient-Based Methods (IGBMs). To reduce the computational burden, another method is to decouple the calculation process of the UL gradient from the dynamic system. For this purpose, IGBMs, which people also refer to as implicit differentiation (Pedregosa, 2016; Rajeswaran et al., 2019; Lorraine et al., 2020), replace the LL sub-problem with an implicit equation. Specifically, when the LL problem is strongly convex (of course in this case the LLS is met), taking advantage of the celebrated implicit function theorem, the gradient of the UL objective is calculated by implicit differential equations. However, this scheme needs to repeatedly compute the inverse of the Hessian matrix. Although in practice, the Conjugate Gradient (CG) method or Neumann method are involved for fast inverse computation, however, repeated products of vectors and matrices are still in the core. Therefore, it is still expensive to compute and leads to numerical instabilities, especially when the system is ill-conditioned. Although, (Rajeswaran et al., 2019) proposes adding a large quadratic term on the LL objective to eliminate the ill-conditionedness, this approach may change the solution set of the BLO problem and cannot generate solutions with desired properties.

Mathematical requirements for the mentioned methods are

listed in Table 1. Indeed, as shown in Table 1, for the theoretical guarantee, except BDA, the LLS and LLC assumptions are required by both of the two classes, which are actually too tough to be satisfied in real-world complex tasks. Recently, (Ji et al., 2020; Ji & Liang, 2021) give a comprehensive study on the nonasymptotic convergence properties of both implicit and explicit gradient-based methods. Specifically, in (Ji & Liang, 2021), the authors provide lower complexity bounds for gradient-type methods and propose accelerated bi-level algorithms to achieve such optimal complexity.

1.2. Our Contributions

In this paper, to address the issues shared by both the explicit and implicit methods, i.e., restrictive LLS and LLC assumptions in theory and repeated Hessian- and Jacobian-vector products in numerical calculation, we initialize a new BLOs solution scheme called Bi-level Value-Function-based Interior-point Method (BVFIM). Our starting point is that the inner Simple Bi-level Optimization (SBO) sub-problem in Eq. (3) can be approximated by more straightforward strategies, i.e., value function approach, a smoothing technique based on regularization and the log-barrier penalization. In particular, we begin with a reformulation of the inner SBO sub-problem by transforming the LL problem solution set constraint into an inequality constraint, using the value function of the regularized LL problem. Motivated by an idea behind the Interior-point Penalty Method (IPM), we consider a log-barrier function to penalize the value function constraint to the objective of the inner SBO sub-problem. The log-barrier penalty together with regularization further results in a series of SLO problems, thereby approximating the inner SBO sub-problem. Note that thanks to the regularizations, the approximation sub-problems are standard unconstrained differentiable optimization problems, which are therefore numerically trackable by popular gradient-type methods. In the absence of the LLS and LLC assumptions, we prove that BVFIM converges to a true solution of the original BLO. Table 1 compares the convergence results of BVFIM and the existing methods. It can be seen that, for the LL sub-problem, assumptions required in previous methods are essentially more restrictive than that in BVFIM. More importantly, when solving BLOs without LLS and LLC, no theoretical results can be obtained for these classical methods while BVFIM can still obtain the same convergence properties as that in LLS and LLC scenario. Now, we briefly streamline the novelty of our approach and contributions.

1. By penalizing the value function of the LL problem through the log-barrier penalty, together with a regularization smoothing technique, BVFIM constructs a series of unconstrained smooth approximated SLO sub-

   problems which are numerically trackable by gradient-type optimization methods.

2. Compared with the existing gradient-based schemes, the theoretical validity of the proposed BVFIM does not rely on the restrictive LLS and LLC assumptions, thereby capturing a much wider range of bi-level learning tasks.

3. The proposed BVFIM avoids the computationally expensive Hessian-vector and Jacobian-vector products computation. Therefore, BVFIM can effectively reduce the numerical cost, especially when the LL problem is of large scale.

2. Gradient-based Methods for BLOs

In this section, from the optimistic bi-level viewpoint, we offer a unified framework that contains existing implicit and explicit gradient methods as special cases. The framework is also used to analyze our BVFIM later on. Due to the unities of algorithmic forms, the superiority of the proposed BVFIM is clearly observed.

To address the LLS restriction, we regard BLOs from an optimistic bi-level perspective. Specifically, we transform BLOs in Eq. (1) into the following form:

$$\min_{\mathbf{x} \in \mathcal{X}} \varphi(\mathbf{x}), \qquad (2)$$

where $\varphi(\mathbf{x})$ is the value function of the inner SBO sub-problem, i.e.,

$$\varphi(\mathbf{x}) := \inf_{\mathbf{y} \in \mathcal{S}(\mathbf{x})} F(\mathbf{x}, \mathbf{y}) \quad \text{with } \mathcal{S}(\mathbf{x}) := \arg\min_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}).$$
$$(3)$$

However, $\varphi(\mathbf{x})$ is a value function of the inner SBO problem, by nature it is very ill-conditioned, namely, non-smooth, non-convex and usually with jumps. We next present existing several types of approximations for the gradient (grad. for short) of $\varphi(\mathbf{x})$ in a unified manner, i.e.,

$$\overbrace{\frac{\partial \varphi(\mathbf{x})}{\partial \mathbf{x}}}^{\text{grad. of } \mathbf{x}} = \overbrace{\frac{\partial F(\mathbf{x}, \mathbf{y})}{\partial \mathbf{x}}}^{\text{direct grad. of } \mathbf{x}} + \overbrace{G(\mathbf{x})}^{\text{indirect grad. of } \mathbf{x}}. \qquad (4)$$

Gradient-based methods, either implicit or explicit, access the gradient of $\varphi(\mathbf{x})$ by approximation with Hessian- and Jacobian-vector products, which results in additional theoretical and computational burdens. Based on the different calculation methods for $G(\mathbf{x})$, we classify the existing gradient-based methods into EGBMs and IGBMs.

EGBMs. With an initialization point $\mathbf{y}_0$, a sequence $\{\mathbf{y}_t(\mathbf{x})\}_{t=0}^T$ parameterized by $\mathbf{x}$ is generated as

$$\mathbf{y}_{t+1}(\mathbf{x}) = \Phi_t(\mathbf{x}, \mathbf{y}_t), \ t = 0, \cdots, T-1, \qquad (5)$$

where $\Phi_t(\mathbf{x}, \mathbf{y}_t(\mathbf{x}))$ is a smooth mapping that represents the operation performed by the t-th step of an optimization algorithm for solving the LL problem. For example, when the gradient descent method is considered, $\Phi(\mathbf{x}, \mathbf{y}_t(\mathbf{x})) = \mathbf{y}_t(\mathbf{x}) - s\frac{\partial f}{\partial \mathbf{y}}(\mathbf{x}, \mathbf{y}_{t-1}(\mathbf{x}))$ where $s > 0$ denoted the corresponding step size. Approximating $\varphi(\mathbf{x})$ by $\varphi_T(\mathbf{x}) := F(\mathbf{x}, \mathbf{y}_T(\mathbf{x}))$, an approximation of $\frac{\partial \varphi(\mathbf{x})}{\partial \mathbf{x}}$ is given by $\frac{\partial \varphi_T(\mathbf{x})}{\partial \mathbf{x}} = \frac{\partial F(\mathbf{x}, \mathbf{y})}{\partial \mathbf{x}} + \frac{\partial F(\mathbf{x}, \mathbf{y}_T)}{\partial \mathbf{y}} \frac{\partial \mathbf{y}_T(\mathbf{x})}{\partial \mathbf{x}}$. Specifically, in this setting, the indirect gradient $G(\mathbf{x})$ is specialized as:

$$G(\mathbf{x}) = \left(\frac{\partial \mathbf{y}_T(\mathbf{x})}{\partial \mathbf{x}}\right)^\top \frac{\partial F(\mathbf{x}, \mathbf{y}_T)}{\partial \mathbf{y}}, \qquad (6)$$

where

$$\frac{\partial \mathbf{y}_t(\mathbf{x})}{\partial \mathbf{x}} = \left(\frac{\partial \mathbf{y}_{t-1}(\mathbf{x})}{\partial \mathbf{x}}\right)^\top \frac{\partial \Phi(\mathbf{x}, \mathbf{y}_{t-1})}{\partial \mathbf{y}} + \frac{\partial \Phi(\mathbf{x}, \mathbf{y}_{t-1})}{\partial \mathbf{x}}. \quad (7)$$

To ensure the quality of the estimation of $G(\mathbf{x})$, EGBMs need LLC, most of them need LLS (except BDA), and the number of iterations T should be large enough. Note that Eq. (7) contains Hessian- and Jacobian-vector products when Φ denotes the gradient descent operator. Thus computing $G(\mathbf{x})$ is time-consuming due to Eq. (7).

IGBMs. Under LLS assumption, $\varphi(\mathbf{x}) = F(\mathbf{x}, \mathbf{y}^*(\mathbf{x}))$ with unique $\mathbf{y}^*(\mathbf{x}) \in \mathcal{S}(\mathbf{x})$. When $\mathbf{y}^*(\mathbf{x})$ is a differentiable function with respect to $\mathbf{x}$, the gradient $\frac{\partial \varphi(\mathbf{x})}{\partial \mathbf{x}}$ can be calculated through chain rule as $\frac{\partial \varphi(\mathbf{x})}{\partial \mathbf{x}} = \frac{\partial F(\mathbf{x}, \mathbf{y})}{\partial \mathbf{x}} + \frac{\partial F(\mathbf{x}, \mathbf{y}^*(\mathbf{x}))}{\partial \mathbf{y}} \frac{\partial \mathbf{y}^*(\mathbf{x})}{\partial \mathbf{x}}$. To obtain $\frac{\partial \mathbf{y}^*(\mathbf{x})}{\partial \mathbf{x}}$, assuming that $\frac{\partial^2 f}{\partial \mathbf{y}^2}$ is invertible, implicit function theorem is applied on the optimality condition of the LL problem (namely, $0 = \frac{\partial f(\mathbf{x}, \mathbf{y}^*(\mathbf{x}))}{\partial \mathbf{y}}$), i.e.,

$$\frac{\partial \mathbf{y}^*(\mathbf{x})}{\partial \mathbf{x}} = -\left(\frac{\partial^2 f(\mathbf{x}, \mathbf{y}^*(\mathbf{x}))}{\partial \mathbf{y}^2}\right)^{-1} \frac{\partial^2 f(\mathbf{x}, \mathbf{y}^*(\mathbf{x}))}{\partial \mathbf{y} \partial \mathbf{x}}. \quad (8)$$

Then $G(\mathbf{x})$ is specialized as

$$G(\mathbf{x}) = -\left(\frac{\partial^2 f(\mathbf{x}, \mathbf{y}^*(\mathbf{x}))}{\partial \mathbf{y} \partial \mathbf{x}}\right)^\top \left(\frac{\partial^2 f(\mathbf{x}, \mathbf{y}^*(\mathbf{x}))}{\partial \mathbf{y}^2}\right)^{-1}$$
$$\frac{\partial F(\mathbf{x}, \mathbf{y}^*(\mathbf{x}))}{\partial \mathbf{y}}. \quad (9)$$

Note that for the sake of approximation quality, the LL objective f must be strongly convex (or $\frac{\partial^2 f(\mathbf{x}, \mathbf{y}^*(\mathbf{x}))}{\partial \mathbf{y}^2}$ is invertible). In practice, the calculation of $G(\mathbf{x})$ given in Eq. (9) is based on numerical approximations. In particular, CG or Neumann methods which involve Hessian-vector product computations are applied. Thus, the computation of $G(\mathbf{x})$ is numerically expensive and the accuracy of the approximation heavily depends on the condition number of Hessian or the strong convexity constant of f (Grazzi et al., 2020).

In general, both EGBMs and IGBMs require LLC and LLS (except BDA) for theoretical convergence. From a computational point of view, the implementation of IGBMs involve repeating productions of Hessian-vector and Jacobian-vector.

3. Bi-level Value-Function-based Interior-point Method

In this section, we propose a new algorithm that penalizes the regularized value function of the LL problem into the UL objective, and thus approximates the value function of the inner SBO sub-problem $\varphi(\mathbf{x})$, called Bi-level Value-Function-based Interior-point Method (BVFIM).

Focusing on $\varphi(\mathbf{x})$ which is in general non-smooth, to apply gradient-type methods, we will design a series of differentiable functions to approximate $\varphi(\mathbf{x})$. Recall that $\varphi(\mathbf{x})$ denotes the value function of the following parameterized SBO:

$$\min_{\mathbf{y} \in \mathbb{R}^n} F(\mathbf{x}, \mathbf{y}), \text{ s.t. } \mathbf{y} \in \arg\min_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}). \quad (10)$$

We can equivalently reformulate Eq. (10) into an SLO problem by using the value function of its LL problem, i.e., $f^*(\mathbf{x}) = \min_{\mathbf{y}} f(\mathbf{x}, \mathbf{y})$. As a consequence, instead of Eq. (10), we study the following parameterized SLO:

$$\min_{\mathbf{y} \in \mathbb{R}^n} F(\mathbf{x}, \mathbf{y}), \text{ s.t. } f(\mathbf{x}, \mathbf{y}) \leq f^*(\mathbf{x}). \quad (11)$$

However, the inequality constraint $f(\mathbf{x}, \mathbf{y}) \leq f^*(\mathbf{x})$ is ill-posed, in the sense that $f^*(\mathbf{x})$ is non-smooth and the constraint does not satisfy any standard regularity condition. To circumvent such difficulty, inspired by (Borges et al., 2020), we relax such constraint by replacing $f^*(\mathbf{x})$ with the value function of the regularized LL problem, i.e.,

$$f_\mu^*(\mathbf{x}) = \min_{\mathbf{y} \in \mathbb{R}^n} f(\mathbf{x}, \mathbf{y}) + \frac{\mu_1}{2}\|\mathbf{y}\|^2 + \mu_2, \quad (12)$$

where $\mu = (\mu_1, \mu_2)$, and μ_1, μ_2 are two positive constants. $\frac{\mu_1}{2}\|\mathbf{y}\|^2$ is introduced for guaranteeing the smoothness of $f_\mu^*(\mathbf{x})$ and μ_2 is for ensuring the feasibility of the relaxed inequality constraint $f(\mathbf{x}, \mathbf{y}) < f_\mu^*(\mathbf{x})$. In the following, f_μ^* is shown to be differentiable under mild conditions. Now, penalizing the relax inequality constraint $f(\mathbf{x}, \mathbf{y}) \leq f_\mu^*(\mathbf{x})$ to the objective by the log-barrier penalty gives us the following smooth approximation of $\varphi(\mathbf{x})$:

$$\varphi_{\mu,\theta,\tau}(\mathbf{x}) = \min_{\mathbf{y} \in \mathbb{R}^n} F(\mathbf{x}, \mathbf{y}) + \frac{\theta}{2}\|\mathbf{y}\|^2 - \tau\ln(f_\mu^*(\mathbf{x}) - f(\mathbf{x}, \mathbf{y})), \quad (13)$$

where $(\mu, \theta, \tau) > 0$. The additional regularized term $\frac{\theta}{2}\|\mathbf{y}\|^2$ is for ensuring the smoothness of $\varphi_{\mu,\theta,\tau}(\mathbf{x})$. It will be shown in the next section that $\varphi_{\mu,\theta,\tau}(\mathbf{x}) \to \varphi(\mathbf{x})$ as $(\mu, \theta, \tau) \to 0$ and $\tau\ln\mu_2 \to 0$. Now, we show the differentiability of $\varphi_{\mu,\theta,\tau}(\mathbf{x})$ and give the formula of its gradient.

Proposition 1. *Suppose $F(\mathbf{x}, \mathbf{y})$ and $f(\mathbf{x}, \mathbf{y})$ are continuously differentiable. Given $\mathbf{x} \in \mathcal{X}$ and $\mu, \theta, \tau > 0$, when*

$$\mathbf{y}^*_{\mu,\theta,\tau}(\mathbf{x})$$
$$= \operatorname*{argmin}_{\mathbf{y} \in \mathbb{R}^n} F(\mathbf{x}, \mathbf{y}) + \frac{\theta}{2}\|\mathbf{y}\|^2 - \tau \ln(f^*_\mu(\mathbf{x}) - f(\mathbf{x}, \mathbf{y})),$$

(14)

and

$$\mathbf{z}^*_\mu(\mathbf{x}) = \operatorname*{argmin}_{\mathbf{y} \in \mathbb{R}^n} f(\mathbf{x}, \mathbf{y}) + \frac{\mu_1}{2}\|\mathbf{y}\|^2 + \mu_2, \quad (15)$$

are unique, then $\varphi_{\mu,\theta,\tau}$ is differentiable and

$$\frac{\partial \varphi_{\mu,\theta,\tau}}{\partial \mathbf{x}}(\mathbf{x}) = \frac{\partial F(\mathbf{x}, \mathbf{y}^*_{\mu,\theta,\tau}(\mathbf{x}))}{\partial \mathbf{x}} + G(\mathbf{x}), \quad (16)$$

where

$$G(\mathbf{x}) = \frac{\tau \left(\frac{\partial f(\mathbf{x}, \mathbf{y}^*_{\mu,\theta,\tau}(\mathbf{x}))}{\partial \mathbf{x}} - \frac{\partial f(\mathbf{x}, \mathbf{z}^*_\mu(\mathbf{x}))}{\partial \mathbf{x}} \right)}{f^*_\mu(\mathbf{x}) - f(\mathbf{x}, \mathbf{y}^*_{\mu,\theta,\tau}(\mathbf{x}))}, \quad (17)$$

and $f^*_\mu(\mathbf{x}) = f(\mathbf{x}, \mathbf{z}^*_\mu(\mathbf{x})) + \frac{\mu_1}{2}\|\mathbf{z}^*_\mu(\mathbf{x})\|^2 + \mu_2$.

The decreasing proximity from Eq. (13) to the original BLOs leans at the core of our convergence theory, as the regularization sequence $\{(\mu_k, \theta_k, \tau_k)\}$ is tending to zero. In particular, the solutions returned by solving approximate sub-problems $\min_{\mathbf{x} \in \mathcal{X}} \varphi_k(\mathbf{x})$, where φ_k denotes $\varphi_{\mu_k, \theta_k, \tau_k}$, converge to the true solution of BLOs, as $\{(\mu_k, \theta_k, \tau_k)\}$ vanishes with $\tau_k \ln \mu_{k,2} \to 0$; see Theorem 1 in Section 4. Naturally, to execute this computing process, we need to calculate the gradient $\frac{\partial \varphi_k}{\partial \mathbf{x}}$, thereby solving each approximate sub-problem efficiently.

We next illustrate the calculation of $\frac{\partial \varphi_k}{\partial \mathbf{x}}$ at $\mathbf{x}_l$ as a guide for implementation. Indeed, for a user-friendly purpose, this procedure is divided into three steps, according to Proposition 1. First, we calculate $\mathbf{z}^*_{\mu_k}(\mathbf{x}_l)$ by solving the regularized LL problem Eq. (12), which can be easily done by gradient descent as

$$\mathbf{z}^t_{k,l} = \mathbf{z}^{t-1}_{k,l} - s_1 \left(\frac{\partial f(\mathbf{x}_l, \mathbf{z}^{t-1}_{k,l})}{\partial \mathbf{y}} + \mu_{k,1} \mathbf{z}^{t-1}_{k,l} \right), \quad (18)$$

where $s_1 > 0$ is an appropriately chosen step size. To calculate $\mathbf{y}^*_{\mu_k, \theta_k, \tau_k}(\mathbf{x}_l)$ returned by solving Eq. (13), the gradient descent scheme reads as

$$\mathbf{y}^t_{k,l} = \mathbf{y}^{t-1}_{k,l}$$
$$- s_2 \left(\frac{\partial F(\mathbf{x}_l, \mathbf{y}^{t-1}_{k,l})}{\partial \mathbf{y}} + \theta_k \mathbf{y}^{t-1}_{k,l} + \frac{\tau_k \frac{\partial f(\mathbf{x}_l, \mathbf{y}^{t-1}_{k,l})}{\partial \mathbf{y}}}{f^{T_\mathbf{z}}_{k,l} - f(\mathbf{x}_l, \mathbf{y}^{t-1}_{k,l})} \right), \quad (19)$$

where $s_2 > 0$ is an appropriately chosen step size. In particular, $f^{T_\mathbf{z}}_{k,l} = f(\mathbf{x}_l, \mathbf{z}^{T_\mathbf{z}}_{k,l}) + \frac{\mu_{k,1}}{2}\|\mathbf{z}^{T_\mathbf{z}}_{k,l}\|^2 + \mu_{k,2}$, where

$\mathbf{z}^{T_\mathbf{z}}_{k,l}$ represents the output returned from $T_\mathbf{z}$-step gradient descent in Eq. (18).

Following Eqs. (16) and (17), we eventually obtain $\frac{\partial \varphi_k(\mathbf{x}_l)}{\partial \mathbf{x}}$ by

$$\frac{\partial \varphi_k(\mathbf{x}_l)}{\partial \mathbf{x}} \approx \frac{\partial F(\mathbf{x}_l, \mathbf{y}^{T_\mathbf{y}}_{k,l})}{\partial \mathbf{x}} + G_{k,l}, \quad (20)$$

with

$$G_{k,l} = \frac{\tau_k \left(\frac{\partial f(\mathbf{x}_l, \mathbf{y}^{T_\mathbf{y}}_{k,l})}{\partial \mathbf{x}} - \frac{\partial f(\mathbf{x}_l, \mathbf{z}^{T_\mathbf{z}}_{k,l})}{\partial \mathbf{x}} \right)}{f^{T_\mathbf{z}}_{k,l} - f(\mathbf{x}_l, \mathbf{y}^{T_\mathbf{y}}_{k,l})}, \quad (21)$$

where $\mathbf{y}^{T_\mathbf{y}}_{k,l}$ represents the output returned from $T_\mathbf{y}$-step gradient descent in Eq. (19), and $\mathbf{z}^{T_\mathbf{z}}_{k,l}, f^{T_\mathbf{z}}_{k,l}$ are stored in the last step.

Algorithm 1 Our Solution Strategy for Eq. (13)

Require: $(\mu_k, \theta_k, \tau_k)$, step size $\alpha, s_1, s_2 > 0$
1: **for** $l = 0 \to L$ **do**
2:     Calculate $\mathbf{z}^*_{\mu_k}(\mathbf{x}_l)$ by $T_\mathbf{z}$-step gradient descent in Eq. (18).
3:     Calculate $\mathbf{y}^*_{\mu_k, \theta_k, \tau_k}(\mathbf{x}_l)$ by $T_\mathbf{y}$-step gradient descent in Eq. (19).
4:     Calculate an approximation of $\frac{\partial \varphi_k(\mathbf{x}_l)}{\partial \mathbf{x}}$ by Eq. (20), denoted by g_l.
5:     $\mathbf{x}_{l+1} = \mathbf{x}_l - \alpha g_l$.
6: **end for**

The algorithm to solve approximate sub-problems Eq. (13) is outlined in Algorithm 1. Then, as $\{(\mu_k, \theta_k, \tau_k)\}$ vanishes, the solution of Algorithm 1 converges to the true solution of Eq. (10).

4. Theoretical Investigations

In this section, we will give the convergence analysis the proposed method. Please notice that all the proofs of our theoretical results are stated in the Supplemental Material.

We first recall an equivalent definition of epiconvergence given in (Bonnans & Shapiro, 2013)[page 41].

Definition 1. $\varphi_k \xrightarrow{e} \varphi$ *iff for all $\mathbf{x} \in \mathbb{R}^m$ the following two conditions hold:*

1. *For any sequence $\{\mathbf{x}_k\}$ converging to $\mathbf{x}$,*

$$\liminf_{k \to \infty} \varphi_k(\mathbf{x}_k) \geq \varphi(\mathbf{x}). \quad (22)$$

2. *There is a sequence $\{\mathbf{x}_k\}$ converging to $\mathbf{x}$ such that*

$$\limsup_{k \to \infty} \varphi_k(\mathbf{x}_k) \leq \varphi(\mathbf{x}). \quad (23)$$

For a given function $f(\mathbf{x}, \mathbf{y})$, we state the property that it is level-bounded in $\mathbf{y}$ locally uniformly in $\mathbf{x} \in \mathcal{X}$ in the following definition.

Definition 2. *Given a function $f(\mathbf{x}, \mathbf{y}) : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$. If for a point $\bar{\mathbf{x}} \in \mathcal{X} \subseteq \mathbb{R}^m$, for any $c \in \mathbb{R}$, there exists $\delta > 0$ along with a bounded set $\mathcal{B} \in \mathbb{R}^m$, such that*

$$\{\mathbf{y} \in \mathbb{R}^n | f(\mathbf{x}, \mathbf{y}) \leq c\} \subseteq \mathcal{B}, \ \ \forall \mathbf{x} \in \mathcal{B}_\delta(\bar{\mathbf{x}}) \cap \mathcal{X}, \quad (24)$$

then we call $f(\mathbf{x}, \mathbf{y})$ is level-bounded in $\mathbf{y}$ locally uniformly in $\bar{\mathbf{x}} \in \mathcal{X}$. If the above property holds for each $\bar{\mathbf{x}} \in \mathcal{X}$, we further call $f(\mathbf{x}, \mathbf{y})$ is level-bounded in $\mathbf{y}$ locally uniformly in $\mathbf{x} \in \mathcal{X}$.

We will give the convergence result of the proposed algorithm under the following standing assumptions:

Assumption 1. *We take the following as our blanket assumption*

1. *$\mathcal{S}(\mathbf{x})$ is nonempty for $\mathbf{x} \in \mathcal{X}$.*
2. *Both $F(\mathbf{x}, \mathbf{y})$ and $f(\mathbf{x}, \mathbf{y})$ are jointly continuous and continuously differentiable.*
3. *Either $F(\mathbf{x}, \mathbf{y})$ or $f(\mathbf{x}, \mathbf{y})$ is level-bounded in $\mathbf{y}$ locally uniformly in $\mathbf{x} \in \mathcal{X}$.*

To prove the convergence result, we show that $\varphi_k(\mathbf{x}) + \delta_\mathcal{X}(\mathbf{x}) \xrightarrow{e} \varphi(\mathbf{x}) + \delta_\mathcal{X}(\mathbf{x})$, where $\delta_\mathcal{X}(\mathbf{x})$ denotes the indicator function of set $\mathcal{X}$, i.e., $\delta_\mathcal{X}(\mathbf{x}) = 0$ if $\mathbf{x} \in \mathcal{X}$ and $\delta_\mathcal{X}(\mathbf{x}) = +\infty$ if $\mathbf{x} \notin \mathcal{X}$. To do this, we need to verify two conditions given in Definition 1.

First, to show the condition 1 in Definition 1, we introduce the value function $\psi_\mu(\mathbf{x})$ of the relaxed problem of Eq. (11):

$$\psi_\mu(\mathbf{x}) = \min_{\mathbf{y} \in \mathbb{R}^n} F(\mathbf{x}, \mathbf{y}), \ \ \text{s.t.} \ -1 \leq f(\mathbf{x}, \mathbf{y}) - f_\mu^*(\mathbf{x}) \leq 0.$$
$$(25)$$

And we propose the following two lemmas.

Lemma 1. *Let $\{\mu_k\}$ be a positive sequence such that $\mu_k \to 0$. Then for any sequence $\{\mathbf{x}_k\}$ converging to $\bar{\mathbf{x}}$,*

$$\limsup_{k \to \infty} f_{\mu_k}^*(\mathbf{x}_k) \leq f^*(\bar{\mathbf{x}}).$$

Lemma 2. *Given $\mathbf{x} \in \mathcal{X}$, suppose either $F(\mathbf{x}, \mathbf{y})$ or $f(\mathbf{x}, \mathbf{y})$ is level-bounded in $\mathbf{y}$ locally uniformly in $\mathbf{x}$. Let $\{\mu_k\}$ be a positive sequence such that $\mu_k \to 0$, and then for any sequence $\{\mathbf{x}_k\}$ converging to $\mathbf{x}$,*

$$\liminf_{k \to \infty} \psi_{\mu_k}(\mathbf{x}_k) \geq \varphi(\mathbf{x}). \quad (26)$$

Next, we verify condition 2 in Definition 1.

Lemma 3. *Let $\{(\mu_k, \theta_k, \tau_k)\}$ be a positive sequence such that $(\mu_k, \theta_k, \tau_k) \to 0$ and $\tau_k \ln \mu_{k,2} \to 0$. Then for any $\mathbf{x} \in \mathcal{X}$,*

$$\limsup_{k \to \infty} \varphi_k(\mathbf{x}) \leq \varphi(\mathbf{x}).$$

Now, by combining the results obtained above, we can obtain the desired epiconvergence result.

Proposition 2. *Suppose either $F(\mathbf{x}, \mathbf{y})$ or $f(\mathbf{x}, \mathbf{y})$ is level-bounded in $\mathbf{y}$ locally uniformly in $\mathbf{x} \in \mathcal{X}$. Let $\{(\mu_k, \theta_k, \tau_k)\}$ be a positive sequence such that $(\mu_k, \theta_k, \tau_k) \to 0$ and $\tau_k \ln \mu_{k,2} \to 0$, and then*

$$\varphi_k(\mathbf{x}) + \delta_\mathcal{X}(\mathbf{x}) \xrightarrow{e} \varphi(\mathbf{x}) + \delta_\mathcal{X}(\mathbf{x}). \quad (27)$$

In summary, we can get the following convergence result:

Theorem 1. *Suppose either $F(\mathbf{x}, \mathbf{y})$ or $f(\mathbf{x}, \mathbf{y})$ is level-bounded in $\mathbf{y}$ locally uniformly in $\mathbf{x} \in \mathcal{X}$. Let $\{(\mu_k, \theta_k, \tau_k)\}$ be a positive sequence such that $(\mu_k, \theta_k, \tau_k) \to 0$ and $\tau_k \ln \mu_{k,2} \to 0$. Then*

1. *We have the following inequality*

$$\limsup_{k \to \infty} \left(\inf_{\mathbf{x} \in \mathcal{X}} \varphi_k(\mathbf{x}) \right) \leq \inf_{\mathbf{x} \in \mathcal{X}} \varphi(\mathbf{x}). \quad (28)$$

2. *If $\mathbf{x}_\ell \in \operatorname{argmin}_{\mathbf{x} \in \mathcal{X}} \varphi_\ell(\mathbf{x})$, for some sequence $\{\ell\} \subset \mathbb{N}$ and $\mathbf{x}_\ell$ converges to $\tilde{\mathbf{x}}$, then $\tilde{\mathbf{x}} \in \operatorname{argmin}_{\mathbf{x} \in \mathcal{X}} \varphi(\mathbf{x})$ and*

$$\lim_{\ell \to \infty} \left(\inf_{\mathbf{x} \in \mathcal{X}} \varphi_\ell(\mathbf{x}) \right) = \inf_{\mathbf{x} \in \mathcal{X}} \varphi(\mathbf{x}). \quad (29)$$

5. Complexity Analysis

In this part, we compare the time and space complexity of Algorithms 1 in computing the direction for updating variable $\mathbf{x}$ with EGBMs (i.e.,FHG and RHG) and IGBMs (i.e.,CG and Neumann). Our complexity analysis is based on the basic step. We suppose the gradients of F, f, Hessian-vector product $\frac{\partial^2 f}{\partial \mathbf{y}^2} \mathbf{q}$ and Jacobian-vector product $\frac{\partial^2 f}{\partial \mathbf{x} \partial \mathbf{y}} \mathbf{q}$ can be evaluated in time $c = c(m, n)$ for any vector $\mathbf{q} \in \mathbb{R}^n$. For all existing methods, we assume the optimal solution of the LL problem is calculated by a T-step gradient descent, and the transition function in FHG and RHG, also the Φ in Eq. (5), is gradient descent. In practice, it has been shown in (Rajeswaran et al., 2019) that the time and space cost for computing $\frac{\partial^2 f}{\partial \mathbf{y}^2} \mathbf{q}$ and $\frac{\partial^2 f}{\partial \mathbf{x} \partial \mathbf{y}} \mathbf{q}$ via AD is no more than a (universal) constant (e.g., usually 2-5 times) order over the cost for computing gradient of F and f.

EGBMs. As discussed in (Franceschi et al., 2017; Shaban et al., 2019), FHG for forward calculation of Hessian-matrix product can be evaluated in time $O(cmT)$ and space $O(mn)$, and RHG for reverse calculation of Hessian- and Jacobian-vector products can be evaluated in time $O(cT)$ and space $O(m + nT)$. The time complexity for EGBMs to calculate the UL gradient is proportional to the number of iterations T of the LL problem, and thus EGBMs take a large amount of time to ensure convergence.

Table 2. Complexity analysis of various bi-level methods on unconstrained problems in calculating $\frac{\partial \varphi(\mathbf{x})}{\partial \mathbf{x}}$. We show the key update steps. $\mathbf{p} \in \mathbb{R}^n$ and $\mathbf{q} \in \mathbb{R}^m$ are intermediate variables and $\mathbf{Z} \in \mathbb{R}^{m \times n}$ is intermediate matrix. $\mathbf{I}$ is the identity matrix. FHG and RHG calculate Eq. (6) by forward or reverse AD. CG calculates Hessian inverse by solving linear equation, while Neumann approximates Hessian inverse by calculating the limit of the sequence. Please see (Franceschi et al., 2017; Pedregosa, 2016; Lorraine et al., 2020) for more calculation details. Note that our method avoid calculating the Hessian- and Jacobian-vector products.

Method	$G(\mathbf{x})$	Time	Space
FHG	$\mathbf{Z}_T^\top \frac{\partial F(\mathbf{x}, \mathbf{y}_T)}{\partial \mathbf{y}}$ with $\mathbf{Z}_t = \frac{\partial^2 f}{\partial \mathbf{y}^2} \mathbf{Z}_{t-1} + \frac{\partial^2 f}{\partial \mathbf{y} \partial \mathbf{x}}$	$O(cmT)$	$O(mn)$
RHG	$\mathbf{q}_{-1}$ with $\mathbf{q}_{t-1} = \mathbf{q}_t + \left(\frac{\partial^2 f}{\partial \mathbf{x} \partial \mathbf{y}} \right)^\top \mathbf{p}_t, \mathbf{p}_{t-1} = \left(\frac{\partial^2 f}{\partial \mathbf{y}^2} \right)^\top \mathbf{p}_t$	$O(cT)$	$O(m + nT)$
CG	$- \left(\frac{\partial^2 f(\mathbf{x}, \mathbf{y}_T)}{\partial \mathbf{y} \partial \mathbf{x}} \right)^\top \mathbf{q}$ with $\frac{\partial^2 f}{\partial \mathbf{y}^2} \mathbf{q} = \frac{\partial F}{\partial \mathbf{y}}$	$O(c(T + J))$	$O(m + n)$
Neumann	$- \left(\frac{\partial^2 f(\mathbf{x}, \mathbf{y}_T)}{\partial \mathbf{y} \partial \mathbf{x}} \right)^\top \mathbf{q} \frac{\partial F}{\partial \mathbf{y}}$ with $\mathbf{q} = \sum_{j=0}^{J} \left(\mathbf{I} - \frac{\partial^2 f}{\partial \mathbf{y}^2} \right)^j$	$O(c(T + J))$	$O(m + n)$
BVFIM	$\frac{\tau_k}{f_{k,l}^{T_{\mathbf{z}}} - f(\mathbf{x}_l, \mathbf{y}_{k,l}^{T_{\mathbf{y}}})} \left(\frac{\partial f(\mathbf{x}_l, \mathbf{y}_{k,l}^{T_{\mathbf{y}}})}{\partial \mathbf{x}} - \frac{\partial f(\mathbf{x}_l, \mathbf{z}_{k,l}^{T_{\mathbf{z}}})}{\partial \mathbf{x}} \right)$	$O(c(T_{\mathbf{z}} + T_{\mathbf{y}}))$	$O(m + n)$

IGBMs. After implementing a T-step gradient descent for the LL problem, IGBMs approximate matrix inverse by solving J steps of a linear system. As each step contains a Hessian-vector product, this part required $O(cJ)$ time and $O(m + n)$ space. So IGBMs run in time $O(c(T + J))$ and space $O(m + n)$. The iteration number J always relies on the properties of the Hessian-matrix, and thus in some cases, J can be much larger than T.

BVFIM. In our algorithm, $T_{\mathbf{z}}$ steps of gradient descent in Eq. (18) for the LL problem solution $\mathbf{z}^{T_{\mathbf{z}}}$ requires time $O(cT_{\mathbf{z}})$ and space $O(n)$. Then $T_{\mathbf{y}}$ gradient descent steps in Eq. (19) are used calculate $\mathbf{y}^{T_{\mathbf{y}}}$, which requires time $O(cT_{\mathbf{y}})$ and space $O(n)$. Lastly, the direction can be obtained according to the formula given in Eq. (20) by several computations of the gradient $\frac{\partial f}{\partial \mathbf{x}}$ and $\frac{\partial F}{\partial \mathbf{x}}$ without any intermediate update, which requires time $O(c)$ and space $O(m + n)$, so BVFIM runs in time $O(c(T_{\mathbf{z}} + T_{\mathbf{y}}))$ and space $O(m + n)$, respectively. Note that although BVFIM has similar complexity to existing methods, it does not need any computation of Hessian- or Jacobian-vector products and all the complexity comes from calculating the gradients of F and f. And because calculating the gradient is much faster than calculating Hessian- and Jacobian-vector products (even by AD) when the dimension of LL variable is large, the proposed BVFIM is more efficient than existing methods when the LL problem is high-dimensional, for example, in applications with a large-scale network. We will illustrate this advantage in the next section.

6. Experimental Results

In this section, we present numerical simulations with the proposed methods and demonstrate application in hyper-parameter optimization for non-convex LL problems.

6.1. Toy Examples

In this section, our goal is to use a simple numerical sample to verify the superiority of BVFIM over existing methods in the case of non-convex LL problems. We use the following numerical examples [1]

$$\min_{x \in \mathbb{R}, y \in \mathbb{R}} \|x - a\|^2 + \|y - a\|^2, \text{ s.t. } y \in \operatorname*{argmin}_{y \in \mathbb{R}} \sin(x + y),$$
(30)

where a is an adjustable parameter which will be set as $a = 0$ and 2 in following experiments. By simple calculation, we know that the optimal solution of Eq. (30) is $x_{a=0}^* = y_{a=0}^* = -\pi/4$, $F_{a=0}^* = \pi^2/8$ and $x_{a=2}^* = y_{a=2}^* = 3\pi/4$, $F_{a=3}^* = 3\pi^2/16 - 9\pi/2 + 9$. This example satisfies all the assumptions of BVFIM, but not LLS and LLC assumptions in (Pedregosa, 2016; Rajeswaran et al., 2019; Lorraine et al., 2020), which makes it be a good example to validate the advantages of BVFIM.

In Figure 1, we show the comparison of BVFIM with the explicit method RHG and the implicit method CG for solving different bi-level optimization problems at different initial points. It can be observed that RHG and CG converge to the optimal point only when the initial point is near the optimal point, but different initial points only slightly affect the convergence of BVFIM. From the second column in Figure 1, we can see that the effectiveness of the BVFIM method in solving non-convex problems comes at the expense of slowing down the convergence rate of the LL problem f.

To further illustrate the differences between the BVFIM methods and existing methods on the LL problem, we plot the convergence behavior of the LL iteration (i.e., $\|y_t(x_0) - y^*\|$ and $\|y_t(x_{500}) - \mathcal{S}(x_{500})\|$) with given UL

---

[1]Since variables are one-dimensional real numbers in numerical experiments, we do not use bold letters which represent vectors.
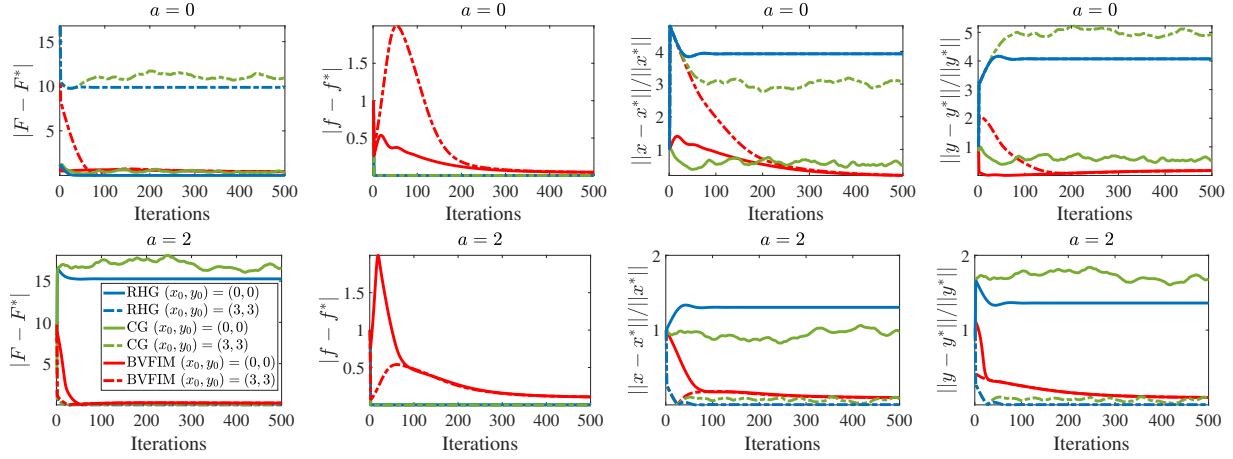
Figure 1. Comparison of the existing explicit method (RHG) and implicit method (CG) with BVFIM on different metrics and different initial points. We set $a = 0$ in the first row and $a = 2$ in the second row. The solid line represents $(x_0, y_0) = (0, 0)$, and the dotted line represents $(x_0, y_0) = (3, 3)$. The legend is only plotted in the first subfigure of the second line.
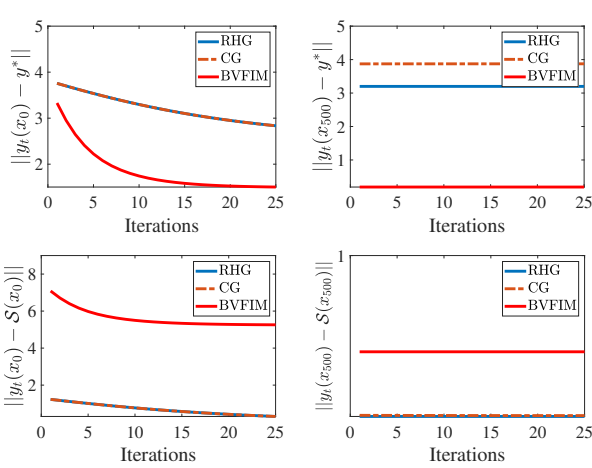

Figure 2. LL iterative curves with explicit and implicit methods (RHG and CG) in x_0 and x_{500}. $\mathcal{S}(x)$ is the solution set of the LL problem with given x, and y^* is the global optimal solution. We set $a = 0$ and $(x_0, y_0) = (3, 3)$.
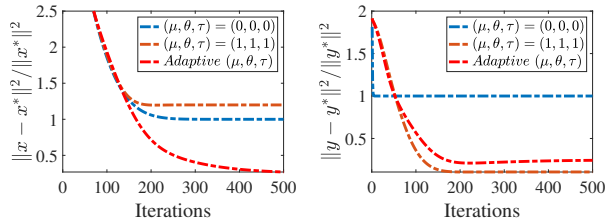

Figure 3. Convergence curves under different $(\mu_k, \theta_k, \tau_k)$ settings. The legend is only plotted in the second subfigure. We set $a = 0$ and $(x_0, y_0) = (3, 3)$. The adaptive (μ, θ, τ) means to follow the default setting which can be found in Supplementary Materials.
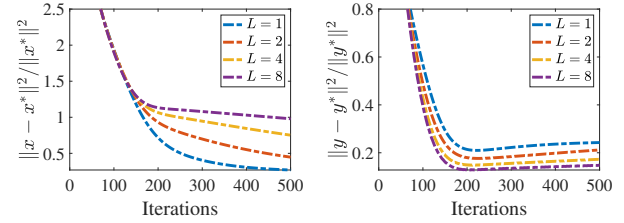

Figure 4. Comparison of convergence rates under different L settings. We set $a = 0$ and $(x_0, y_0) = (3, 3)$. The iterations are the number of gradient descents of x, which is $K \times L$.

variable x in Figure 2. Note that x_0 and x_{500} refer to the x at the 0th and 500th iteration of the UL problem, respectively. From the first row in Figure 2, it can be observed that RHG and CG converge quickly to the solution set of the LL problem, while BVFIM does not because $y_t(x)$ in BVFIM is solving problem Eq. (13). However, the second row of Figure 2 shows that BVFIM can converge to the global optimal solution while RHG and CG cannot.

Figure 3 shows the convergence behavior of BVFIM with different regularization parameter settings. It can be seen that by setting the regularization parameter to 0, our method degenerates to optimize UL and LL problems separately without regard to their relevance, thus it is difficult to obtain an appropriate feasible solution. Choosing a fixed nonzero regularization parameter can improve convergence on y and achieve the same convergence behavior on x as adaptive (μ, θ, τ) at the early stage. However, due to the influence of the regular term, it does not converge to the optimal solution on x. When the regularization term takes a decreasing value, the best results are obtained.

Figure 4 shows the effect of different iteration number L

Table 3. Comparison of the results of existing methods for solving data hyper-cleaning tasks on MNIST, FashionMNIST and CIFAR10. The F1 score denotes the harmonic mean of the precision and recall.

Method	MNIST			FashionMNIST			CIFAR10		
	Acc.	F1 score	Time(s)	Acc.	F1 score	Time(s)	Acc.	F1 score	Time/s
RHG	87.90	89.36	0.4131	81.91	87.12	0.4589	34.95	68.27	1.3374
TRHG	88.57	89.77	0.2623	81.85	86.76	0.2840	35.42	68.06	0.8409
CG	89.19	85.96	0.1799	83.15	85.13	0.2041	34.16	69.10	0.4796
Neumann	87.54	89.58	0.1723	81.37	87.28	0.1958	33.45	68.87	0.4694
BDA	87.15	90.38	0.6694	79.97	88.24	0.8571	36.41	67.33	1.4869
BVFIM	**90.41**	**91.19**	**0.1480**	**84.31**	**88.35**	**0.1612**	**38.19**	**69.55**	**0.4092**

on convergence behavior. We can find that smaller L can accelerate the convergence of variable x, while larger L can improve the convergence of variable y. These preliminary results indicate that in situations where optimal solution x^* is needed, BVFIM with small L (e.g. $L = 1$) is a better choice.

6.2. Hyper-parameter Optimization

In this section, we use a special case of hyper-parameter optimization, called data hyper-cleaning, to evaluate the performance of our method when the LL problem is non-convex. Assuming that some of the labels in our datasets are contaminated, the goal of data hyper-cleaning is to reduce the impact of incorrect samples by adding hyper-parameters to the contaminated samples.

In the experiment, we set $\mathbf{y} \in \mathbb{R}^{10 \times 301} \times \mathbb{R}^{300 \times d}$ as the parameters of 2-layer linear network classifier where d is the data dimension and $\mathbf{x} \in \mathbb{R}^{|\mathcal{D}_{\text{tr}}|}$ as the weight of each sample in the training set. Therefore, the LL problem is to learn a classifier $\mathbf{y}$ on cross-entropy training loss weighted with given $\mathbf{x}$

$$f(\mathbf{x}, \mathbf{y}) = \sum_{(u_i, v_i) \in \mathcal{D}_{\text{tr}}} \sigma(\mathbf{x}_i) \mathtt{CE}(\mathbf{y}, u_i, v_i), \quad (31)$$

where (u_i, v_i) are the training examples, $\sigma(\mathbf{x})$ denotes the sigmoid function on $\mathbf{x}$ to constrain the weights in the range $[0, 1]$ and $\mathtt{CE}$ is the cross entropy loss. The UL problem is to find a weight $\mathbf{x}$ which can reduce the cross-entropy loss of $\mathbf{y}$ on a cleanly labeled validation set

$$F(\mathbf{x}, \mathbf{y}) = \sum_{(u_i, v_i) \in \mathcal{D}_{\text{val}}} \mathtt{CE}(\mathbf{y}, u_i, v_i). \quad (32)$$

Our experiment is based on MNIST, FashionMNIST (Xiao et al., 2017) and CIFAR10 datasets. For each dataset, we randomly select 5000 samples as the training set $\mathcal{D}_{\text{tr}}$, 5000 samples as the validation set $\mathcal{D}_{\text{val}}$, and 10000 samples as the test set $\mathcal{D}_{\text{test}}$. Then, we contaminated half of the labels in $\mathcal{D}_{\text{tr}}$.

Table 3 shows the results and calculation time of the compared methods and BVFIM on three different datasets. It can
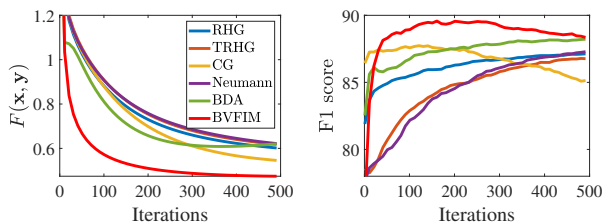

Figure 5. $F(\mathbf{x}, \mathbf{y})$ and F1 score between existing methods and BVFIM. The curves are based on the FashionMNIST experiment in Table 3. The legend is only plotted in the first subfigure.

be seen that BVFIM achieves the most competitive results on all three datasets. Furthermore, BVFIM is faster than EGBMs and IGBMs, and this advantage is more evident in CIFAR10 datasets with larger LL parameters, consistent with the complexity analysis given above. The UL objective and F1 scores of BVFIM and compared methods on the FashionMNIST dataset are plotted in Figure 5.

7. Conclusions

In this paper, we present a new bi-level optimization algorithm BVFIM. By penalizing the regularized LL problem into the UL problem, we approximate the BLO problem by a series of unconstrained smooth SLO sub-problems. We prove the convergence without requiring any convexity assumption on either the UL or the LL objectives. Extensive evaluations show the superiority of BVFIM for different applications.

Acknowledgements

References

Bergstra, J. and Bengio, Y. Random search for hyper-parameter optimization. *JMLR*, 2012.

Bonnans, J. F. and Shapiro, A. *Perturbation analysis of optimization problems*. Springer Science & Business Media, 2013.

Borges, P., Sagastizábal, C., and Solodov, M. A regularized smoothing method for fully parameterized convex problems with applications to convex and nonconvex two-stage stochastic programming. *Mathematical Programming*, 2020.

Chen, X., Xie, L., Wu, J., and Tian, Q. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *ICCV*, 2019.

Dempe, S. *Bilevel optimization: theory, algorithms and applications*. TU Bergakademie Freiberg, Fakultät für Mathematik und Informatik, 2018.

Franceschi, L., Donini, M., Frasconi, P., and Pontil, M. Forward and reverse gradient-based hyperparameter optimization. In *ICML*, 2017.

Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., and Pontil, M. Bilevel programming for hyperparameter optimization and meta-learning. In *ICML*, 2018.

Grazzi, R., Franceschi, L., Pontil, M., and Salzo, S. On the iteration complexity of hypergradient computation. In *ICML*, 2020.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, 2011.

Jeroslow, R. G. The polynomial hierarchy and a simple model for competitive analysis. *Mathematical programming*, 1985.

Ji, K. and Liang, Y. Lower bounds and accelerated algorithms for bilevel optimization. *arXiv preprint arXiv:2102.03926v2*, 2021.

Ji, K., Yang, J., and Liang, Y. Bilevel optimization: Nonasymptotic analysis and faster algorithms. *arXiv preprint arXiv:2010.07962v2*, 2020.

Liang, H., Zhang, S., Sun, J., He, X., Huang, W., Zhuang, K., and Li, Z. Darts+: Improved differentiable architecture search with early stopping. *arXiv preprint arXiv:1909.06035*, 2019.

Liu, H., Simonyan, K., and Yang, Y. DARTS: differentiable architecture search. In *ICLR*, 2019a.

Liu, R., Cheng, S., He, Y., Fan, X., Lin, Z., and Luo, Z. On the convergence of learning-based iterative methods for nonconvex inverse problems. *IEEE TPAMI*, 2019b.

Liu, R., Li, Z., Zhang, Y., Fan, X., and Luo, Z. Bi-level probabilistic feature learning for deformable image registration. In *IJCAI*, 2020a.

Liu, R., Liu, J., Jiang, Z., Fan, X., and Luo, Z. A bilevel integrated model with data-driven layer ensemble for multi-modality image fusion. *IEEE TIP*, 2020b.

Liu, R., Mu, P., Chen, J., Fan, X., and Luo, Z. Investigating task-driven latent feasibility for nonconvex image modeling. *IEEE TIP*, 2020c.

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 *ICML*, 2020d.

Liu, R., Gao, J., Zhang, J., Meng, D., and Lin, Z. Investigating bi-level optimization for learning and vision from a unified perspective: A survey and beyond. *arXiv preprint arXiv:2101.11517*, 2021.

Lorraine, J., Vicol, P., and Duvenaud, D. Optimizing millions of hyperparameters by implicit differentiation. In *AISTATS*, 2020.

Mackay, M., Vicol, P., Lorraine, J., Duvenaud, D., and Grosse, R. Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions. In *ICLR*, 2018.

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

Outrata, J. V. On the numerical solution of a class of stackelberg problems. *ZOR-Methods and Models of Operations Research*, 1990.

Pedregosa, F. Hyperparameter optimization with approximate gradient. In *ICML*, 2016.

Pfau, D. and Vinyals, O. Connecting generative adversarial networks and actor-critic methods. *arXiv preprint arXiv:1610.01945*, 2016.

Rajeswaran, A., Finn, C., Kakade, S. M., and Levine, S. Meta-learning with implicit gradients. In *NeurIPS*, 2019.

Shaban, A., Cheng, C., Hatch, N., and Boots, B. Truncated back-propagation for bilevel optimization. In *AISTATS*, 2019.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Yang, Z., Chen, Y., Hong, M., and Wang, Z. Provably global convergence of actor-critic: A case for linear quadratic regulator with ergodic cost. In *NeurIPS*, 2019.

Ye, J. J. and Zhu, D. L. Optimality conditions for bilevel programming problems. *Optimization*, 1995.

Zügner, D. and Günnemann, S. Adversarial attacks on graph neural networks via meta learning. In *ICLR*, 2019.