A Trust-Region Interior-Point Method for Bilelvel Optimization

Anonymous Author(s)
Affiliation
Address
email

Abstract

Bilevel optimization (BLO) is popular in designing learning tasks in hyperparameter tuning, meta learning, reinforcement learning and adversarial learning. Most existing methods for solving BLO are gradient based methods, which often need to solve a lower level problem approximately to obtain an approximate hyper-gradient. In this paper, we propose the first second-order interior-point method (IPM) based on value function approach for solving BLO, i.e., the Bilevel Trust-Region Interior-Point Method (BTRIPM). As the value function reformulation is nonconvex, we adopt the trust-region method to solve the log-barrier subproblem. Like IPMs in nonlinear optimization, the BTRIPM admits empirical rapid convergence. We theoretically prove convergence and rate of convergence of the proposed method under mild conditions that are widely used in nonlinear IPM or BLO community. Experiments on a toy example and hyperparameter tuning with real-world datasets demonstrate the efficiency and accuracy of the proposed method over existing first-order methods.

1 Introduction

Bilevel Optimization (BLO) refers to a type of Optimization problems with hierarchical structures. It is widely applied in practical machine learning models [5, 9, 12], such as hyper-parameter optimization [19, 15], meta learning [20, 8], reinforcement learning [27], adversarial learning [3, 2, 25]. Generally, a BLO takes the following formulation

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

where $F:\mathcal{X}\times\mathbb{R}^n\to\mathbb{R}$ is called the Upper-Level (UL) objective and $f:\mathbb{R}^m\times\mathbb{R}^n\to\mathbb{R}$ is the Lower-Level (LL) objective. In the literature [19, 7], the solution set of the LL problem $\operatorname{argmin}_{\mathbf{y}} f(\mathbf{x},\mathbf{y})$ is often required to be a singleton, denoted as $\{\mathbf{y}^*(\mathbf{x})\}$, in order that (1) can be reformulated as a single-level optimization problem, i.e. $\min\limits_{\mathbf{x}\in\mathcal{X}}\phi(\mathbf{x}) = F(\mathbf{x},\mathbf{y}^*(\mathbf{x}))$. We call $\partial_{\mathbf{x}}\phi$ and $\partial^2_{\mathbf{xx}}\phi$ the hyper-gradient and hyper-Hessian of (1), respectively.

1.1 Related Work

Generally, BLO is intrinsically NP hard [1], where the difficulty lies in dealing with the special constraint. In the literature, there are various approaches for solving BLO. To find an approximate hyper-gradient, Explicit Gradient-Based Methods (EGBMs) [7, 8] use dynamics on iterative algorithms to solve the LL problem. In this framework, Reverse Hyper-Gradient (RHG) and Forward Hyper-Gradient (FHG) methods identify the hyper-gradient by forward and reverse computation iterations, respectively. To ease the computation, Shaban et al. [22] develops a technique that truncates the back-propagation process to reduce the computation. Besides, Implicit Gradient-Based Methods (IGBMs) [19, 20, 14] are also prevalent for BLO. Using the first-order optimality condition for LL problem and the chain rule, IGBMs solve a linear system to calculate the hyper-gradient. Even approximately solving the linear system by the Conjugate Gradient (CG) method or the Neumann method as widely used in the literature [19, 14], IGBMs demand huge computation. To avoid the expensive Hessian vector products, Liu et al. [13] recently proposes an algorithm termed Bilevel Value-Function-based Interior-point Method (BVFIM). By approximating the constraint with a series of inequalities based on value functions of the LL constraint, the BVFIM uses the log-penalty function to combine the UL and LL objectives. More specifically, in each step, the BVFIM first approximately minimizes the LL problem w.r.t. $\mathbf{y}$ at current $(\bar{\mathbf{x}}_k, \bar{\mathbf{y}}_k)$ to obtain a value function representation of the LL problem, then minimizes, w.r.t. $\mathbf{y}$, the objective function penalized by the log barrier of the value function inequality to achieve a smooth approximation of $\phi(\mathbf{x})$, and finally applies gradient descent to update $\mathbf{x}$ using this smooth approximation. Numerical experiments in Liu et al. [13] show BVFIM outperforms existing methods.

However, to the best of our knowledge, there are almost no second-order algorithms considered in the BLO literature or in the machine learning society, though some works in BLO [17, 29] analyzed the second-order optimality conditions. One reason may be that it is very difficult to estimate the hyper-Hessian $\partial^2_{\mathbf{xx}}\phi(\mathbf{x})$. As is well known, second-order methods enjoy rapid convergence in general nonlinear optimization. This motivates us to design the algorithm in this paper.

1.2 Our Contributions

In this paper, we propose a Bilevel Trust-Region Interior-Point Method (BTRIPM), which is the first value function based second-order method for BLO. We approximate the LL constraint $\mathbf{y} \in \operatorname{argmin}_{\mathbf{y}} f(\mathbf{x}, \mathbf{y})$ by an inequality associated with its value function following value function approaches in the literature [13, 28]. We penalize a relaxation of this value function inequality with the log-barrier penalty and all other inequalities in $\mathcal{X}$ as in the interior-point method (IPM) literature [6]. Unlike Liu et al. [13], we minimize a sequence of penalized problems regarding both $\mathbf{x}$ and $\mathbf{y}$ as decision variables. As the log-barrier problems are possibly nonconvex, we distinguish our algorithm from convex IPMs by applying a trust-region method to solve the log-barrier problems.

The Hessian vector product in our BTRIPM can be computed in a cost dominated by solving a linear system in dimension with n equations. The cost can further be reduced and fast convergence can still be guaranteed in practice if we use the upper level Hessian to approximate the true Hessian when the dimension of LL problem is high. As a second-order method, our algorithm converges faster than first-order methods in the literature [7, 8, 22, 19, 20, 14, 19, 14, 13]. Our BTRIPM needs often several tens of outer iterations to obtain a solution with higher precision and the computational time is less than the-state-of-the-art methods in the experiments.

Moreover, we theoretically prove that the proposed algorithm converges to a strict local optimal solution under mild assumptions. Our proof technique is totally different from the existing first-order methods for BLO. Our technique successfully addresses the well known difficulty in analyzing the value function approach that many usual constraint qualification (CQ) such as the nonsmooth Mangasarian Fromovitz constraint qualification fail at each feasible point (see, e.g., [28]). Specifically, we show that the sequence generated by our algorithm converges to a KKT point of a relaxation of the value function reformulation of (1) that the linear independence constraint qualification (LICQ) holds under minor conditions, and show that the KKT point is a strict local minimum of (1) under some additional minor conditions.

In summary, our contributions are as follows:

- We propose the BTRIPM, the first second-order interior-point method for BLO. The BTRIPM first uses value function approach to rewrite the LL problem as an inequality constraint, and then applies trust-region methods to minimize a sequence of log-barrier problems.
- We are the first to prove the local minimizers of a relaxed value function reformulation of BLO converge to the local minimizer of the original problem. Based on this, we prove that our algorithm converges to a strict local minimizer of (1) under mild conditions.
- Our experiments show that the BTRIPM is faster and more accurate than existing methods in the literature on both toy example and real-world datasets.

Notation. In this paper we consider $\mathcal{X} = \{\mathbf{x} : \mathbf{c}(\mathbf{x}) \geq 0\}$ where $\mathbf{c} : \mathbb{R}^m \to \mathbb{R}^\kappa$, κ is the number of the inequality constraints.[1] Through the paper, we assume the UL and LL functions F and f, and $c_i, i \in [\kappa]$ are twice continuously differentiable functions in its domain. We define derivatives as follows: for a scaler function $h_1(\mathbf{x})$,

$$\partial h_1(\mathbf{x}) \triangleq \partial_{\mathbf{x}} h_1(\mathbf{x}) = \frac{\partial h_1(\mathbf{x})}{\partial \mathbf{x}} = \left(\frac{\partial h_1(\mathbf{x})}{\partial x_1} \cdots \frac{\partial h_1(\mathbf{x})}{\partial x_m} \right)^T \in \mathbb{R}^m,$$

and for a vector function $\mathbf{h}(\mathbf{x}) \in \mathbb{R}^n$,

$$\partial_{\mathbf{x}} \mathbf{h}(\mathbf{x}) = \frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial h_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial h_1(\mathbf{x})}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_n(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial h_n(\mathbf{x})}{\partial x_m} \end{pmatrix} \in \mathbb{R}^{n \times m}.$$

Following this, we further define

$$\partial_{\mathbf{y}} \partial_{\mathbf{x}} f(\mathbf{x}, \mathbf{y}) \triangleq \partial_{\mathbf{y}} (\partial_{\mathbf{x}} f(\mathbf{x}, \mathbf{y})) \in \mathbb{R}^{m \times n}.$$

For notational convenience, for $c_i(\mathbf{x})$ as a function of $\mathbf{x}$, we always use the convention $\partial c_i(\mathbf{x}) = (\partial_{\mathbf{x}} c_i(\mathbf{x})^T, \mathbf{0}^T)^T \in \mathbb{R}^{m+n}$, and if we want to specify the gradient of c_i w.r.t. $\mathbf{x}$, we use $\partial_{\mathbf{x}} c_i(\mathbf{x})$. Given $\mathbf{x} \in \mathcal{X}$, we always use $\mathbf{z}^*(\mathbf{x})$ to denote an element of $\operatorname{argmin}_{\mathbf{y}} f(\mathbf{x}, \mathbf{y})$, which is a vector function of $\mathbf{x}$ and not unique if $\operatorname{argmin}_{\mathbf{y}} f(\mathbf{x}, \mathbf{y})$ is not a singleton. Several definitions, technical lemmas and all the proofs defer to the appendix.

2 The Bilevel Trust-region Interior-point Method

In this section, we provide a new algorithm that incorporates the second-order information using nonconvex IPM framework.

To begin with, we reformulate the LL constraint into an inequality constraint based on the value function approach following [13, 28]. Specifically, (1) is equivalent to

$$\min_{\mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathbb{R}^n} F(\mathbf{x}, \mathbf{y}), \text{ s.t. } f(\mathbf{x}, \mathbf{y}) \leq f^*(\mathbf{x}), \tag{2}$$

where $f^*(\mathbf{x}) = \min_{\mathbf{y}} f(\mathbf{x}, \mathbf{y})$. However, the constraint is not friendly to IPMs as we always have $f^*(\mathbf{x}) \leq f(\mathbf{x}, \mathbf{y})$ and thus there is no interior point. To avoid this, we introduce a parameter $\mu > 0$ to relax the inequality constraint and obtain

$$\min_{\mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathbb{R}^n} F(\mathbf{x}, \mathbf{y}), \text{ s.t. } f(\mathbf{x}, \mathbf{y}) \leq f^*(\mathbf{x}) + \mu. \tag{3}$$

As $\mathcal{X} = \{\mathbf{x} : c(\mathbf{x}) \geq 0\}$, (3) is then equivalent to

$$\begin{array}{ll} \min_{\mathbf{x} \in \mathbb{R}^m, \mathbf{y} \in \mathbb{R}^n} & F(\mathbf{x}, \mathbf{y}) \\ \text{s.t.} & f^*_\mu(\mathbf{x}) - f(\mathbf{x}, \mathbf{y}) \geq 0, \quad c_i(\mathbf{x}) \geq 0, i \in [\kappa], \end{array} \tag{4}$$

where $f^*_\mu(\mathbf{x}) \triangleq f^*(\mathbf{x}) + \mu$. Using the log-barrier penalty as in IPMs, we obtain the following penalized problem

$$\min_{\mathbf{x}, \mathbf{y}} g(\mathbf{x}, \mathbf{y}), \tag{5}$$

where

$$g(\mathbf{x}, \mathbf{y}) \triangleq \quad F(\mathbf{x}, \mathbf{y}) - \tau \ln(f^*_\mu(\mathbf{x}) - f(\mathbf{x}, \mathbf{y})) - \tau \textstyle\sum_{i=1}^{\kappa} \ln(c_i(\mathbf{x})).$$

To achieve a fast convergence in solving the unconstrained problem (5), in contrast to Liu et al. [13] we consider a second-order algorithm rather than use gradient descent. Moreover, as (5) is nonconvex, we cannot use damped Newton's method like usual IPMs. Instead, we apply the trust-region method that ensures monotonic decrease of objective value to solve (5) [4].

In the following we explore the formula of first- and second-order derivatives of $g(\mathbf{x}, \mathbf{y})$.

[1]We remark our algorithm allows additional affine constraints. For notational simplicity in the convergence analysis in Section 3, we only consider inequality constraints in this paper.

Table 1: The details of the second-order derivative of $\varphi(x, y)$

Notation	Expression
$\partial^2_{\mathbf{xx}} g(\mathbf{x}, \mathbf{y})$	$\partial^2_{\mathbf{xx}} F(\mathbf{x}, \mathbf{y}) + \tau \frac{\partial^2_{\mathbf{xx}} f(\mathbf{x},\mathbf{y}) - \partial^2_{\mathbf{xx}} f^*_\mu(\mathbf{x})}{f^*_\mu(\mathbf{x}) - f(\mathbf{x},\mathbf{y})} + \tau \frac{\left[\partial_\mathbf{x} f(\mathbf{x},\mathbf{y}) - \partial_\mathbf{x} f^*_\mu(\mathbf{x})\right]\left[\partial_\mathbf{x} f(\mathbf{x},\mathbf{y}) - \partial_\mathbf{x} f^*_\mu(\mathbf{x})\right]^T}{\left(f^*_\mu(\mathbf{x}) - f(\mathbf{x},\mathbf{y})\right)^2}$ $+\tau \sum_{i=1}^{\kappa} \left[\frac{\partial_\mathbf{x} c_i(\mathbf{x}) \partial_\mathbf{x} c_i(\mathbf{x})^T}{c_i(\mathbf{x})^2} - \frac{\partial^2_{\mathbf{xx}} c_i(\mathbf{x})}{c_i(\mathbf{x})}\right]$
$\partial^2_{\mathbf{yy}} g(\mathbf{x}, \mathbf{y})$	$\partial^2_{\mathbf{yy}} F(\mathbf{x}, \mathbf{y}) + \tau \frac{\partial^2_{\mathbf{yy}} f(\mathbf{x},\mathbf{y})}{f^*_\mu(\mathbf{x}) - f(\mathbf{x},\mathbf{y})} + \tau \frac{[\partial_\mathbf{y} f(\mathbf{x},\mathbf{y})][\partial_\mathbf{y} f(\mathbf{x},\mathbf{y})]^T}{\left(f^*_\mu(\mathbf{x}) - f(\mathbf{x},\mathbf{y})\right)^2}$
$\partial_\mathbf{x} \partial_\mathbf{y} g(\mathbf{x}, \mathbf{y})$	$\partial_\mathbf{x} \partial_\mathbf{y} F(\mathbf{x}, \mathbf{y}) + \tau \frac{\partial_\mathbf{x} \partial_\mathbf{y} f(\mathbf{x},\mathbf{y})}{f^*_\mu(\mathbf{x}) - f(\mathbf{x},\mathbf{y})} + \tau \frac{\partial_\mathbf{y} f(\mathbf{x},\mathbf{y})\left[\partial_\mathbf{x} f(\mathbf{x},\mathbf{y}) - \partial_\mathbf{x} f^*_\mu(\mathbf{x})\right]^T}{\left(f^*_\mu(\mathbf{x}) - f(\mathbf{x},\mathbf{y})\right)^2}$
$\partial^2_{\mathbf{xx}} f^*_\mu(\mathbf{x})$	$\partial^2_{\mathbf{xx}} f(\mathbf{x}, \mathbf{z}^*(\mathbf{x})) - \partial_\mathbf{y} \partial_\mathbf{x} f(\mathbf{x}, \mathbf{z}^*(\mathbf{x}))(\partial^2_{\mathbf{yy}} f(\mathbf{x}, \mathbf{z}^*(\mathbf{x})))^{-1} \partial_\mathbf{x} \partial_\mathbf{y} f(\mathbf{x}, \mathbf{z}^*(\mathbf{x}))$
$\partial_\mathbf{x} f^*_\mu(\mathbf{x})$	$\partial_\mathbf{x} f\left(\mathbf{x}, \mathbf{z}^*(\mathbf{x})\right)$

Proposition 2.1. *Suppose $\mu, \tau > 0$. If $\mathbf{z}^*(\mathbf{x})$ is a differentiable function in a neighborhood of $\mathbf{x}$, then $g(\mathbf{x}, \mathbf{y})$ is differentiable and*

$$\partial g(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} \partial_\mathbf{x} g(\mathbf{x}, \mathbf{y}) \\ \partial_\mathbf{y} g(\mathbf{x}, \mathbf{y}) \end{pmatrix},$$

where

$$\partial_\mathbf{x} g(\mathbf{x}, \mathbf{y}) = \partial_\mathbf{x} F(\mathbf{x}, \mathbf{y}) - \tau \frac{\partial_\mathbf{x} f(\mathbf{x}, \mathbf{z}^*(\mathbf{x})) - \partial_\mathbf{x} f(\mathbf{x}, \mathbf{y})}{f^*_\mu(\mathbf{x}) - f(\mathbf{x}, \mathbf{y})} - \tau \sum_{i=1}^{\kappa} \frac{\partial_\mathbf{x} c_i(\mathbf{x})}{c_i(\mathbf{x})}$$

and

$$\partial_\mathbf{y} g(\mathbf{x}, \mathbf{y}) = \partial_\mathbf{y} F(\mathbf{x}, \mathbf{y}) + \tau \frac{\partial_\mathbf{y} f(\mathbf{x}, \mathbf{y})}{f^*_\mu(\mathbf{x}) - f(\mathbf{x}, \mathbf{y})}.$$

Proposition 2.2. *Suppose $\mu, \tau > 0$. If $\mathbf{z}^*(\mathbf{x})$ is a differentiable function in a neighborhood of $\mathbf{x}$ and $\partial^2_{\mathbf{yy}} f(\mathbf{x}, \mathbf{z}^*(\mathbf{x}))$ is invertible, then $g(\mathbf{x}, \mathbf{y})$ is twice differentiable and*

$$\partial^2 g(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} \partial^2_{\mathbf{xx}} g(\mathbf{x}, \mathbf{y}) & \partial_\mathbf{y} \partial_\mathbf{x} g(\mathbf{x}, \mathbf{y}) \\ \partial_\mathbf{x} \partial_\mathbf{y} g(\mathbf{x}, \mathbf{y}) & \partial^2_{\mathbf{yy}} g(\mathbf{x}, \mathbf{y}) \end{pmatrix},$$

where the details of $\partial^2_{\mathbf{xx}} g, \partial_\mathbf{x} \partial_\mathbf{y} g, \partial^2_{\mathbf{yy}} g$ are listed in the Table 1.[2]

We summarise our method in Algorithm 1, which using barrier methods sequentially solves

$$\begin{array}{ll} \min_{\mathbf{x} \in \mathbb{R}^m, \mathbf{y} \in \mathbb{R}^n} & F(\mathbf{x}, \mathbf{y}) \\ \text{s.t.} & f^*_{\mu_k}(\mathbf{x}) - f(\mathbf{x}, \mathbf{y}) \geq 0, \quad c_i(\mathbf{x}) \geq 0, i \in [\kappa]. \end{array} \tag{6}$$

In each iteration, for $\tau = \tau_{k,l}$ and $\mu = \mu_k$, we parameterize g by k and l as follows

$$g_{k,l}(\mathbf{x}, \mathbf{y}) \triangleq F(\mathbf{x}, \mathbf{y}) - \tau_{k,l} \ln(f^*_{\mu_k}(\mathbf{x}) - f(\mathbf{x}, \mathbf{y})) - \tau_{k,l} \sum_{i=1}^{\kappa} \ln(c_i(\mathbf{x})).$$

Thus each iteration of our algorithm minimizes the log-barrier function

$$\min_{\mathbf{x}, \mathbf{y}} g_{k,l}(\mathbf{x}, \mathbf{y}). \tag{7}$$

Now we apply the trust-region algorithm to solve problem (7). The trust-region subproblem at the tth step, denoted by $\mathbf{s}_t = (\mathbf{x}_t, \mathbf{y}_t)$, is as follows

$$\min_{\|\mathbf{d}\| \leq \Delta_t} m_{k,l}(\mathbf{d}) = g_{k,l}(\mathbf{s}_t) + \nabla g_{k,l}(\mathbf{s}_t)^T \mathbf{d} + \frac{1}{2} \mathbf{d}^T B_t \mathbf{d}. \tag{8}$$

Here B_t needs not to be the exact Hessian of $g_{k,l}$, and the convergence of the trust-region method can still be guaranteed [4]. In practice, we use the Steihaug-Toint truncated CG method [23, 24] to solve

[2] As the Hessian is symmetric and F and f are twice continuously differentiable, we always have $\partial_\mathbf{x} \partial_\mathbf{y} g = \partial_\mathbf{y} \partial_\mathbf{x} g^T$.

Algorithm 1 Bilevel Trust-region Interior-point Method

```
1: Input: parameters $\eta > 1$, $\tau = \tau_0$, $\mu_0 > 0$, and $u > 1$
2: for $k = 1$ to $K$ do
3:    $\mu = \mu_0/u^k$;
4:    for $l = 1$ to $T$ do
5:       $\tau = \tau/\eta$;
6:       compute $\mathbf{z}^*(\mathbf{x}) = \mathrm{argmin}_y f(\mathbf{x}, \mathbf{y})$;
7:       derive (approximate) Hessian $B_k$ for $g_{k,l}(\mathbf{x}, \mathbf{y})$;
8:       solve log-barrier minimization problem (7) to obtain $(\mathbf{x}_{k,l}, \mathbf{y}_{k,l})$;
9:    end for
10: end for
```

the trust-region subproblem, where each step only involves Hessian vector products. This allows us to solve large scale machine learning applications. More details on the complexity analysis for the subproblem see Section 4. We will show in the next section that when μ_k and $\tau_{k,l}$ both approach 0, any limiting point of the sequence generated by Algorithm 1 is a local optimal solution of the original BLO (1) under mild assumptions.

As a closing remark for this section, we point out several main differences of our method with the BVFIM in Liu et al. [13]. First, we simultaneously update $\mathbf{x}$ and $\mathbf{y}$, while the BVFIM only updates $\mathbf{x}$. Second, our method is a second-order method, while the BVFIM is a first-order method. Third, only one minimization for $\min_{\mathbf{y}} f(\mathbf{x}_k, \mathbf{y})$ is required in each iteration to compute the gradient and Hessian vector products at $(\mathbf{x}, \mathbf{y})$ in BTRIPM, while the BVFIM needs to solve an additional minimization problem

$$\varphi_{\mu,\tau}(\mathbf{x}) = \min_{\mathbf{y}} F(\mathbf{x}, \mathbf{y}) - \tau \ln(f^*_\mu(\mathbf{x}) - f(\mathbf{x}, \mathbf{y}))$$

to compute the hyper-gradient $\partial_{\mathbf{x}} \phi(\mathbf{x})$. Forth, our method can handle general nonlinear constraint $\mathbf{c}(\mathbf{x}) \geq 0$, while the BVFIM can only handle simply constraints with easy projection if we extend their method using projected gradient methods w.r.t. $\mathbf{x}$. Moreover, as we will discuss in the next section, we prove that our algorithm converges to a local minimum if each inner sequence (iterates generated by the trust-region method for solving the log-barrier problem (7)) converges to a *local* minimum of each subproblem, while the BVFIM is only guaranteed to converge under a very strong condition that the inner sequence converges to a *global* minimum of each subproblem. Our proof technique is totally different from [13]

3 Theoretical Investigations

In this section, we mainly give theoretical convergence results of the BTRIPM. Before that, let us recall some notation from nonlinear optimization ([18]). Consider a general constrained optimization problem

$$\begin{array}{ll} \min_{\mathbf{x}\in\mathbb{R}^n} & \theta(\mathbf{x}) \\ \text{s.t.} & \mathbf{a}(\mathbf{x}) \geq \mathbf{0},\ \mathbf{b}(\mathbf{x}) = \mathbf{0}, \end{array} \tag{9}$$

where $\theta : \mathbb{R}^n \to \mathbb{R}$, $\mathbf{a} : \mathbb{R}^n \to \mathbb{R}^p$ and $\mathbf{b} : \mathbb{R}^n \to \mathbb{R}^l$ are twice continuously differentiable functions in its domain. Denote the Lagrange multipliers of $\mathbf{a}(\mathbf{x}) \geq 0$ and $\mathbf{b}(\mathbf{x}) = 0$ by $\boldsymbol{\lambda}^a$ and $\boldsymbol{\lambda}^b$, respectively. We say $(\boldsymbol{\lambda}^a, \boldsymbol{\lambda}^b)$ is an acceptable Lagrange multiplier vector at $\bar{\mathbf{x}}$ if $(\bar{\mathbf{x}}; \boldsymbol{\lambda}^a, \boldsymbol{\lambda}^b)$ satisfies the KKT conditions. Let $\mathcal{I}$ be the active set of the inequality constraints, i.e., $\mathcal{I} \triangleq \{i : \mathbf{a}_i(\mathbf{x}) = 0,\ i \in [p]\}$. We say that the linear independence constraint qualification (LICQ) holds at the feasible point $\bar{\mathbf{x}}$ if $\{\partial a_i(\bar{\mathbf{x}}), \partial b_j(\bar{\mathbf{x}}) : i \in \mathcal{I},\ j \in [l]\}$ are linear independent. For problem (9), we say strict complementarity holds at the KKT point $\bar{\mathbf{x}}$ if there exists an acceptable Lagrange multiplier vector $(\boldsymbol{\lambda}^a, \boldsymbol{\lambda}^b)$ such that $\bar{\lambda}^a_i > 0$ for all $i \in \mathcal{I}$. Let $L(\mathbf{x}; \boldsymbol{\lambda}^a, \boldsymbol{\lambda}^b)$ denote the Lagrangian function of (9). Define the critical cone

$$\mathcal{C}(\mathbf{x}, \boldsymbol{\lambda}^a) \triangleq \left\{ \mathbf{s} : \begin{array}{l} \partial\mathbf{b}(\mathbf{x})\mathbf{s} = \mathbf{0},\ \partial a_i(\mathbf{x})^T\mathbf{s} = 0 \,\text{for all}\, i \in \mathcal{I} \,\text{with}\, \lambda^a_i > 0, \\ \text{and}\ \partial a_i(\mathbf{x})^T\mathbf{s} \geq 0 \,\text{for all}\, i \in \mathcal{I} \,\text{with}\, \lambda^a_i = 0 \end{array} \right\}.$$

Let $(\bar{\mathbf{x}}_k, \bar{\mathbf{y}}_k)$ and $(\mathbf{x}^*_{k,l}, \mathbf{y}^*_{k,l})$ be local minimizers of (6) and (7), respectively. Define $c_0(\mathbf{x}, \mathbf{y}) \triangleq f(\mathbf{x}, \mathbf{y}) - f^*(\mathbf{x})$. We make the following assumption before presenting our main results.

Assumption 3.1. Assume the following conditions hold. (1) The sequence $(\bar{\mathbf{x}}_k, \bar{\mathbf{y}}_k)$ is bounded and, by passing to a subsequence if necessary, $(\bar{\mathbf{x}}_k, \bar{\mathbf{y}}_k) \to (\bar{\mathbf{x}}, \bar{\mathbf{y}})$ as $k \to \infty$. (2) $\partial^2_{\mathbf{yy}} f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \succ 0$. (3) $\mathbf{z}^*(\mathbf{x})$ is continuous in a neighborhood of $\bar{\mathbf{x}}$ with $\mathbf{z}^*(\bar{\mathbf{x}}) = \bar{\mathbf{y}}$. (4) The vectors $\{\partial_{\mathbf{x}} c_i(\bar{\mathbf{x}})\}_{i \in \bar{\mathcal{A}}}$ are linearly independent, where $\bar{\mathcal{A}} = \{i : c_i(\bar{\mathbf{x}}) = 0, i = 1, 2, \cdots n\}$

Similar assumptions except (3) are widely used in the IPM literature [6]. Indeed, (3) holds if f is level-bounded in $\mathbf{y}$, locally uniformly in $\mathbf{x} \in \mathcal{X}$ and $\operatorname{argmin}_{\mathbf{y}} f(\mathbf{x}, \mathbf{y})$ is a singleton for all $\mathbf{x} \in \mathcal{X}$ (see Lemma 3 in Liu et al. [11]). More discussion on this assumption see Appendix H. We also point out that (4) usually holds in machine learning scenarios, e.g., when $\mathcal{X} = \mathbb{R}^m$ or $\mathcal{X}$ is a box constraint, the linear independence of $\{\partial_{\mathbf{x}} c_i(\bar{\mathbf{x}})\}_{i \in \bar{\mathcal{A}}}$ holds trivially.

Note that $f(\mathbf{x}, \mathbf{y}) \le f^*(\mathbf{x})$ implies $\partial_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) = 0$ due to the first-order condition, and, moreover, no CQ holds for (2) ([28]), which means that the KKT conditions may not be necessary optimality conditions of (2). This motivates us to introduce the following auxiliary problem

$$\begin{array}{ll} \min_{\mathbf{x} \in \mathbb{R}^m, \mathbf{y} \in \mathbb{R}^n} & F(\mathbf{x}, \mathbf{y}) \\ \text{s.t.} & c_i(\mathbf{x}) \ge 0, i \in [\kappa], \quad \partial_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) = \mathbf{0}. \end{array} \tag{10}$$

Note also that the feasible region of (10) is larger than that of (2) and (10) and (2) have the same objective functions. Under Assumption 3.1, we can show that LICQ holds for (10) (see Lemma E.6), and thus the KKT conditions are necessary optimality conditions for (10). Then using the relationship of the KKT conditions of (10) and (6), we show that the KKT conditions of (10) holds at any limiting point of the local minimum sequence $\{(\bar{\mathbf{x}}_k, \bar{\mathbf{y}}_k)\}$ of (6).

Theorem 3.2. *Let Assumption 3.1 hold. Then for problem* (6) *and sufficiently large k, the KKT conditions hold at $(\bar{\mathbf{x}}_k, \bar{\mathbf{y}}_k)$ with an acceptable Lagrange multiplier vector $(\lambda_0^k, \boldsymbol{\lambda}^k)$. Moreover, letting $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ be any limiting point of the sequence $\{(\bar{\mathbf{x}}_k, \bar{\mathbf{y}}_k)\}$, the KKT conditions hold at $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ with an acceptable Lagrange multiplier vector $(\boldsymbol{\gamma}, \boldsymbol{\nu}) = (\lim_{k\to\infty} \boldsymbol{\lambda}^k, \lim_{k\to\infty} \lambda_0^k(\bar{\mathbf{x}}_k, \mathbf{z}^*(\bar{\mathbf{x}}_k)))$ for problem* (10)*, where $\boldsymbol{\gamma}$ and $\boldsymbol{\nu}$ are the Lagrange multipliers of $\mathbf{c}(\mathbf{x}) \ge 0$ and $\partial_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) = 0$, respectively.*

We further obtain convergence rate results for the BTRIPM in the following theorem.

Theorem 3.3. *Let the conditions of Theorem 3.2 hold and $\bar{L}(\bar{\mathbf{x}}, \bar{\mathbf{y}}; \boldsymbol{\gamma}, \boldsymbol{\nu})$ be the Lagrangian function for* (10)*, then there exist Lagrangian multipliers $\boldsymbol{\gamma}_k$ and $\boldsymbol{\nu}_k$ such that*

$$\|\partial \bar{L}(\bar{\mathbf{x}}_k, \bar{\mathbf{y}}_k; \boldsymbol{\gamma}_k, \boldsymbol{\nu}_k)\| = O(u^{-\frac{k}{2}}). \tag{11}$$

Recalling the definitions of the second-order necessary conditions (SONCs) and second-order sufficient conditions (SOSCs) in optimization literature, which are also stated in Definitions A.4 and A.5 for completeness, we next show the relationship between the KKT solutions of (10) and local minimizers of (1).

Theorem 3.4. *Assume that Assumption 3.1 holds, strict complementarity holds at $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ for* (10)*, and f is third-order continuously differentiable in a neighborhood of $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$. Let $\bar{L}(\bar{\mathbf{x}}, \bar{\mathbf{y}}; \boldsymbol{\gamma}, \boldsymbol{\nu})$ be the Lagrangian function for* (10) *with the same $(\boldsymbol{\gamma}, \boldsymbol{\nu})$ in Theorem 3.2. Then we have the following conclusions.*

1. *The SONCs hold at $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ for problem* (10)*. That is, $\forall \mathbf{s} \in \mathcal{C}((\bar{\mathbf{x}}, \bar{\mathbf{y}}), \boldsymbol{\gamma})$, we have*

$$\mathbf{s}^T \bar{L}(\bar{\mathbf{x}}, \bar{\mathbf{y}}; \boldsymbol{\gamma}, \boldsymbol{\nu}) \mathbf{s} \ge \liminf_{k\to\infty} w_k \|\mathbf{s}\|^2 \ge 0,$$

for some sequence $\{w_k\}$ satisfying $w_k \ge 0 \ \forall k > 0$, where $\mathcal{C}((\bar{\mathbf{x}}, \bar{\mathbf{y}}), \boldsymbol{\gamma})$ denotes the critical cone at $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$.

2. *If $\limsup_{k\to\infty} w_k > 0$, then the SOSCs hold at $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ for problem* (10) *and $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ is a strict local optimal solution for* (1).

Following the conditions in Theorem 3.4, we further show the convergence and rate of convergence of BTRIPM to a local optimal solution under suitable choice of T_k.

Theorem 3.5. *Suppose that the assumptions in Theorem 3.4 hold, $\limsup_{k\to\infty} w_k > 0$ and strict complementarity holds at $(\bar{\mathbf{x}}_k, \bar{\mathbf{y}}_k)$ for* (6) *for sufficiently large k. Then if each T_k is chosen sufficiently large, there exists a subsequence of $\{(\mathbf{x}^*_{k,l}, \mathbf{y}^*_{k,l})\}$, denoted by $\{(\mathbf{x}^*_{k',l}, \mathbf{y}^*_{k',l})\}$, such that*

Table 2: Complexity analysis of different algorithms for BLO. We use $\mathbf{p} \in \mathbb{R}^n, \mathbf{q} \in \mathbb{R}^m$ and $\mathbf{d} \in \mathbb{R}^{m+n}$ to denote the intermediate vectors, and Z to denote the intermediate matrix. $J(S)$ denotes the number of CG (truncated CG) steps performed in one outer iteration by CG method (BTRIPM). T_z represents the number of gradient steps to update $\mathbf{y}$ in BVFIM and our method. The analysis for RHG, FHG, CG and BVFIM can be found in [13].

Method	Main Update Steps	Time	Space
RHG	$Z_G^\top \frac{\partial F(\mathbf{x},\mathbf{y}_G)}{\partial \mathbf{y}}$ with $Z_t = \frac{\partial^2 f}{\partial \mathbf{y}^2} Z_{t-1} + \frac{\partial^2 f}{\partial \mathbf{y} \partial \mathbf{x}}$	$O(cG)$	$O(nG)$
FHG	$\mathbf{q}_{-1}$ with $\mathbf{q}_{t-1} = \mathbf{q}_t + \left(\frac{\partial^2 f}{\partial \mathbf{x} \partial \mathbf{y}}\right)^\top \mathbf{p}_t, \mathbf{p}_{t-1} = \left(\frac{\partial^2 f}{\partial \mathbf{y}^2}\right)^\top \mathbf{p}_t$	$O(cmG)$	$O(mn)$
CG	$-\left(\frac{\partial^2 f(\mathbf{x},\mathbf{y}_G)}{\partial \mathbf{y} \partial \mathbf{x}}\right)^\top \mathbf{q}$ with $\frac{\partial^2 f}{\partial \mathbf{y}^2}\mathbf{q} = \frac{\partial F}{\partial \mathbf{y}}$ by CG	$O(c(G+J))$	$O(m+n)$
BVFIM	$\frac{\tau_k}{f_{k,l}^{T_\mathbf{z}} - f\left(\mathbf{x}_l, \mathbf{y}_{k,l}^{T_\mathbf{y}}\right)} \left(\frac{\partial f\left(\mathbf{x}_l, \mathbf{y}_{k,l}^{T_\mathbf{y}}\right)}{\partial \mathbf{x}} - \frac{\partial f\left(\mathbf{x}_l, \mathbf{z}_{k,l}^{T_\mathbf{z}}\right)}{\partial \mathbf{x}}\right)$	$O(c(T_\mathbf{z} + T_\mathbf{y}))$	$O(m+n)$
Ours	$\min\limits_{\Vert\mathbf{d}\Vert \le \Delta_t} \frac{1}{2}\mathbf{d}^T B_t \mathbf{d} + \partial g_{k,l}(\mathbf{x}_t, \mathbf{y}_t)^T \mathbf{d}$ by ST CG	$O(c(T_\mathbf{z} + S))$	$O(m+n)$

$\lim_{k' \to \infty}(\mathbf{x}^*_{k',T_{k'}}, \mathbf{y}^*_{k',T_{k'}})$ *exists and is a strict local optimal solution of* (1). *Furthermore, there exist Lagrangian multipliers* $\boldsymbol{\gamma}_{k'}$ *and* $\boldsymbol{\nu}_{k'}$ *such that*

$$\Vert \partial \bar{L}(\mathbf{x}^*_{k',T_{k'}}, \mathbf{y}^*_{k',T_{k'}}; \boldsymbol{\gamma}_{k'}, \boldsymbol{\nu}_{k'}) \Vert = O(\eta^{-\frac{k'}{2}}) + O(u^{-\frac{k'}{2}}).$$

4 Complexity Analysis

In this section, we analyze the main computational cost of our algorithm and provide a comparison with existing ones. This is based on the following widely used assumptions. (i) The existing methods search the optimal solution of the LL problems by G-step gradient descent. Gradient descent also serves as the transition function of RHG and FHG [13]. (ii) The function or gradient evaluation of F or f, and Hessian-vector product associated with $\frac{\partial^2 f}{\partial \mathbf{y}^2}$ and Jacobian vector product associated with $\frac{\partial^2 f}{\partial \mathbf{x} \partial \mathbf{y}}$ are supposed to be calculated in time $c = c(m,n)$ for all $\mathbf{p} \in \mathbb{R}^n$ [20, 13].

Now let us consider the complexity of solving our trust-region model (8) using truncated CG. From Table 1, we can see that calculating the exact Hessian vector product associated with $\partial^2 g$ involves solving a linear system[3], which is computationally expensive when the dimension of $\mathbf{y}$ is high. This inspires us to use an approximation $B_k(\mathbf{s}_t)$ of the true Hessian, which can still guarantee the convergence of the trust-region method [4]. Specifically, we let $B_k(\mathbf{s}_t) = \partial^2 F(\mathbf{x}_t, \mathbf{y}_t)$, then the computation of Hessian vector product $B_k\mathbf{p}$ is $\partial^2 F(\mathbf{x}_t, \mathbf{y}_t)\mathbf{p}$ for any vector $\mathbf{p}$. By our second assumption in this section, its computation is in the same order with gradient or function evaluation of F and f.

The main cost in solving the subproblem is the gradient descent steps for $\min_\mathbf{y} f(\mathbf{x}_t, \mathbf{y})$ and the Hessian vector products $B_k(\mathbf{s}_t)\mathbf{p}$ in CG steps. Denote by S the number of CG steps taken to entirely solve a trust-region subproblem, and by T_z the number of steps to update $\mathbf{y}$ in $\min_\mathbf{y} f(\mathbf{x}_t, \mathbf{y})$. Then the cost of one outer iteration of our algorithm is $O(c(S + T_z))$. Since we only need to restore $\mathbf{z}^*(\mathbf{x}_t),\ \mathbf{s}_t,\ B_k(\mathbf{s}_t)\mathbf{p}$ and the gradients, the consumed space is $O(m+n)$. We point out that $B_k(\mathbf{s}_t)\mathbf{p}$ can be directly computed using the structure of BLO in machine learning applications without explicitly formulating the (approximate) Hessian [14]. We summarise the comparison of our algorithm with the existing methods in Table 2.

Although the cost in a single outer iteration of BTRIPM may not be less than the existing algorithms, BTRIPM needs far fewer outer iterations than any existing algorithms. As a second-order algorithm, at most tens of outer iterations are needed for BTRIPM to obtain a highly accurate solution (see next section). In contrast, the existing first-order algorithms usually demand hundreds of outer iterations

[3]The linear system comes from the Hessian vector product associated with $\partial^2_{\mathbf{x}\mathbf{x}} f^*_\mu(\mathbf{x})$. More specifically, for any $\mathbf{w}$, computing $\left[\partial^2_{\mathbf{y}\mathbf{y}} f\left(\mathbf{x}, \mathbf{z}^*(\mathbf{x})\right)\right]^{-1} \left[\partial_\mathbf{x} \partial_\mathbf{y} f\left(\mathbf{x}, \mathbf{z}^*(\mathbf{x})\right)\right]\mathbf{w}$ involves solving a linear system $\left[\partial^2_{\mathbf{y}\mathbf{y}} f\left(\mathbf{x}, \mathbf{z}^*(\mathbf{x})\right)\right]^{-1}\mathbf{u}$ for $\mathbf{u} = \left[\partial_\mathbf{x} \partial_\mathbf{y} f\left(\mathbf{x}, \mathbf{z}^*(\mathbf{x})\right)\right]\mathbf{w}$.

to reach a relative low accuracy. Therefore, in many cases of interest, BTRIPM is more efficient than the existing methods, as will be illustrated in the next section.

5 Experimental Results

In this section, we conduct experiments to compare the BTRIPM with existing algorithms for BLO. All experiments are implemented using MATLAB R2021b on a PC running Windows 10 Intel(R) Xeon(R) E5-2650 v4 CPU (2.2GHz) and 64GB RAM. Please refer to Appendix F for more experimental results.

5.1 Toy Example

To illustrate the superiority of our algorithm, we test our method on a toy example from Liu et al. [13]

$$\min_{x\in\mathbb{R},y\in\mathbb{R}} x^2+y^2, \quad \text{s.t. } y\in \operatorname{argmin}_{y\in\mathbb{R}} \sin(x+y). \tag{12}$$

We underline that the solution set of the LL problem is not a singleton, which prevents most of the existing methods finding the global optimal solution precisely when the initial point is not close to it. We use gradient descent to identify the optimal solution of LL objective $f(x,y)=\sin(x+y)$ in both of BVFIM and our algorithm. Since this problem is simple, we use the exact Hessian as B_k in the trust-region subproblem.

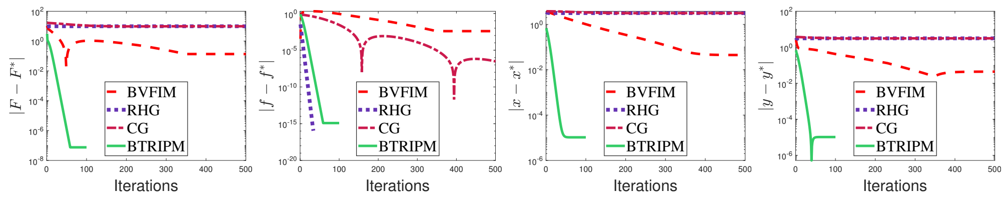

Figure 1: Comparison of BVFIM, RHG, CG with BTRIPM in the toy example (12) with the initial point (x_0,y_0) as $(3,3)$.

We report comparison results in Figure 1. It shows that BTRIPM enjoys high precision even though the LL problem is nonconvex. RHG and CG fails to find the optimal point though they solve the LL problem well and BVFIM suffers from low accuracy. Furthermore, BTRIPM only requires tens of outer iterations to converge while the existing first-order algorithms needs hundreds.

5.2 Data Hyper-cleaning

To further demonstrate the accuracy and efficiency of our algorithm, we test experiments on MNIST and FasionMNIST [26]. Following the previous experiments in Franceschi et al. [7], we randomly choose 5000 images as the training set, 5000 images as the validation set and 10000 images as the test set. We corrupt 2500 labels in the training set by replacing them with different labels randomly. For the data hyper-cleaning model, we use a single fully-connected layer as the network. Intrinsically, it is a linear model with a linear classifier $\mathbf{y}\in\mathbb{R}^{10}\times\mathbb{R}^{7850}$. After applying a softmax regression, we define the weighted cross entropy loss of the output vector as the LL objective function as the experiment in Liu et al. [13], i.e.,

$$f(\mathbf{x},\mathbf{y})=\sum\nolimits_{(\mathbf{u}_i,v_i)\in\mathcal{D}_{\text{tr}}}\sigma(x_i)\,\text{CE}\left(\mathbf{y},\mathbf{u}_i,v_i\right).$$

Here $(\mathbf{u}_i,v_i)$ are the sample data, and the input of the sigmoid functions $\mathbf{x}\in\mathbb{R}^{|\mathcal{D}_{\text{tr}}|}$ is regarded as the UL variable, which directly determines the weights. For the UL objective, the original cross entropy loss is used, only depending on the classifier $\mathbf{y}$, i.e.,

$$F(\mathbf{x},\mathbf{y})=\sum\nolimits_{(\mathbf{u}_i,v_i)\in\mathcal{D}_{\text{val}}}\text{CE}\left(\mathbf{y},\mathbf{u}_i,v_i\right).$$

In data hyper-cleaning model, the weights of the corrupted samples tend to become lower in the training process. Then the corrupted ones will be selected out. In our experiment we choose 0 as the threshold of x_i and calculate the prevalent F1 score of each method. Moreover, we predict the labels of the validation set by the maximal index of the output vector, thus obtaining the accuracy rate of each method. Then we compare our algorithm with the existing methods using these indexes.

Since BVFIM is obviously stronger than EGBMs and IGBMS in hyper-parameter optimization experiments (see [13] and also evidences from previous subsection), we only compare with BVFIM. We test the two algorithms with two different update rules for parameter $(\mu_k, \tau_{k,l})$. We always set $T = 1$ for the inner iterations and then $\tau_{k,1} = \tau_0/\eta^k$. For BVFIM1 and BTRIPM1, we set $\mu_k = \mu_0/u^k$. For BVFIM2 and BTRIPM2, we update parameters $\mu_k = f(\mathbf{x}_k, \mathbf{y}_k)$ as in [13].

Table 3 shows the accuracy, F1 scores and total wall-clock time of the two methods on two different datasets, where accuracy denotes proportion of labels predicted correctly in the test set and F1 score denotes the harmonic mean of the precision and recall. It can be seen that BTRIPM achieves the most competitive results on two datasets. Furthermore, BTRIPM is faster than BVFIM in both μ updating rules, which indicates robustness of BTRIPM in parameter updating.

Table 3: Comparison of the results of BTRIPM and BVFIM.

Method	MNIST			FashionMNIST		
	Acc.	F1 score	Time(s)	Acc.	F1 score	Time(s)
BVFIM1	0.8829	0.8438	354.2613	0.8402	0.8651	356.1338
BVFIM2	0.9046	0.9186	359.7937	0.8452	0.8923	358.3472
BTRIPM1	0.8981	**0.9520**	243.5601	0.8428	**0.9179**	268.8219
BTRIPM2	**0.9055**	0.9487	**241.5207**	**0.8484**	0.9080	**262.8357**

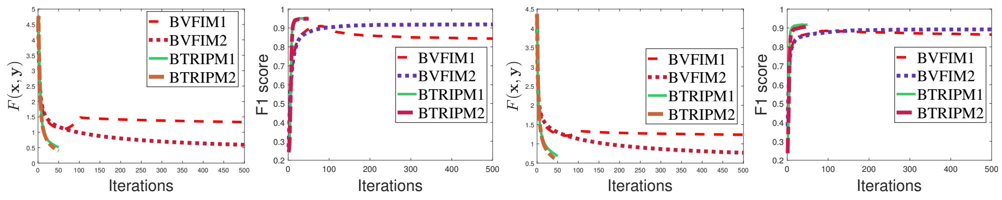

Figure 2: Comparison of the results of BTRIPM and BVFIM for solving data hyper-cleaning tasks. The left two are on MNIST and the right two are on FashionMNIST.

For the same experiment, we also report the UL objective, accuracy and F1 scores of BTRIPM and BVFIM on two datasets in Figure 5.2. As in the last two experiments, BTRIPM converges in tens of outer iterations in both parameter update rules. Meanwhile, BTRIPM achieves higher accuracy and F1 scores.

6 Conclusion and Discussion

In this paper, we propose the first value function based second-order interior-point algorithm for BLO, BTRIPM. We give comprehensive convergence analysis under suitable assumptions that are wildly used in the IPM and BLO literature. Computational analysis and numerical experiments have shown the applicability and superiority of our algorithm over existing first-order algorithms. As future work, we would like to apply our algorithm for more practical BLO applications to see if we can use the structure of different models to reduce the truncated CG cost in the trust-region method for solving log-barrier problems.

References

[1] Ben-Ayed, O. and Blair, C. E. Computational difficulties of bilevel linear programming. *Operations Research*, 38(3):556–560, 1990.

[2] Bishop, N., Tran-Thanh, L., and Gerding, E. Optimal learning from verified training data. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9520–9529, 2020.

[3] Brückner, M. and Scheffer, T. Stackelberg games for adversarial prediction problems. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 547–555, 2011.

[4] Conn, A. R., Gould, N. I., and Toint, P. L. *Trust region methods*. SIAM, 2000.

[5] Dempe, S. Bilevel optimization: theory, algorithms, applications and a bibliography. In *Bilevel Optimization*, pp. 581–672. Springer, 2020.

[6] Forsgren, A., Gill, P. E., and Wright, M. H. Interior methods for nonlinear optimization. *SIAM Review*, 44(4):525–597, 2002.

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

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

[9] Ji, K., Yang, J., and Liang, Y. Bilevel optimization: Convergence analysis and enhanced design. In *International Conference on Machine Learning*, pp. 4882–4892. PMLR, 2021.

[10] Johnson, B. A. and Iizuka, K. Integrating openstreetmap crowdsourced data and landsat time-series imagery for rapid land use/land cover (lulc) mapping: Case study of the laguna de bay area of the philippines. *Applied Geography*, 67:140–149, 2016.

[11] Liu, R., Mu, P., Yuan, X., Zeng, S., and Zhang, J. A generic first-order algorithmic framework for bi-level programming beyond lower-level singleton. In *International Conference on Machine Learning*, pp. 6305–6315. PMLR, 2020.

[12] 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. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021 doi: 10.1109/TPAMI.2021.3132674.

[13] Liu, R., Liu, X., Yuan, X., Zeng, S., and Zhang, J. A value-function-based interior-point method for non-convex bi-level optimization. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 6882–6892. PMLR, 2021

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

[15] MacKay, M., Vicol, P., Lorraine, J., Duvenaud, D., and Grosse, R. Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions. In *International Conference on Learning Representations (ICLR)*, 2019.

[16] Magnus, J. R. and Neudecker, H. *Matrix differential calculus with applications in statistics and econometrics*. John Wiley & Sons, 2019.

[17] Mehlitz, P. and Zemkoho, A. B. Sufficient optimality conditions in bilevel programming. *Mathematics of Operations Research*, 46(4):1573–1598, 2021.

[18] Nocedal, J. and Wright, S. *Numerical optimization*. Springer Science & Business Media, 2006.

[19] Pedregosa, F. Hyperparameter optimization with approximate gradient. In *International Conference on Machine Learning*, pp. 737–746. PMLR, 2016.

[20] Rajeswaran, A., Finn, C., Kakade, S. M., and Levine, S. Meta-learning with implicit gradients. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019.

[21] Rohra, J. G., Perumal, B., Narayanan, S. J., Thakur, P., and Bhatt, R. B. User localization in an indoor environment using fuzzy hybrid of particle swarm optimization & gravitational search algorithm with neural networks. In *Proceedings of Sixth International Conference on Soft Computing for Problem Solving*, pp. 286–295. Springer, 2017.

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

[23] Steihaug, T. The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20(3):626–637, 1983.

[24] Toint, P. Towards an efficient sparsity exploiting newton method for minimization. In *Sparse matrices and their uses*, pp. 57–88. Academic press, 1981.

[25] Wang, J., Chen, H., Jiang, R., Li, X., and Li, Z. Fast algorithms for stackelberg prediction game with least squares loss. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 10708–10716. PMLR.

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

[27] 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 *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 8353–8365, 2019.

[28] Ye, J. J., Yuan, X., Zeng, S., and Zhang, J. Difference of convex algorithms for bilevel programs with applications in hyperparameter selection. *arXiv preprint arXiv:2102.09006*, 2021.

[29] Ye, J. J. and Zhu, D. Optimality conditions for bilevel programming problems. *Optimization*, 33:9–27, 1995.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Please refer to Section 1 for detailed description corresponding to the abstract.
   (b) Did you describe the limitations of your work? [No]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Please refer to Section 3 in the main body and Section D in the supplementary materials.
   (b) Did you include complete proofs of all theoretical results? [Yes] We put most of the detailed proofs in the supplementary materials.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Instruction for the code, datasets are provided in Section 5 and the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Information about the dataset and hyperparameters for numerical experiments can be found in Section 5 and the supplemental material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A] Some experiments e.g. problem 12 is deterministic.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Please refer to Section 5 for detail information.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] All the experiments in this paper are based on public data.

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]