EFFICIENT BI-LEVEL OPTIMIZATION FOR NON-SMOOTH OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Bi-level optimization plays a key role in a lot of machine learning applications. However, existing state-of-the-art bi-level optimization methods are limited to smooth or some specific non-smooth lower-level problems. Even worse, most of them depend on approximating hypergradients to update upper-level variable which is the inherent reason for non-efficiency. Currently, achieving a generalized and efficient optimization algorithm for bi-level problems with a non-smooth, even non-Lipschitz continuous lower-level objective is still an open question to the best of our knowledge. To address these challenging problems, in this paper, we propose a new bi-level optimization algorithm based on the smoothing and penalty techniques. Specifically, we first produce a sequence of smoothed lower-level objectives with an exponential decay smoothing parameter for the non-smooth lower-level problem. Then, we transform the smoothed bi-level optimization to an unconstrained penalty problem by replacing the smoothed sub-problem with its first-order necessary conditions. Finally, we update the upper and lower-level variables alternately with doubly stochastic gradients of the unconstrained penalty problem. Importantly, we provide the theoretical analysis to show that our method can converge to a stationary point of original non-smooth bi-level problem if the lower-level problem is convex, and we give the necessary condition of the original problem if the lower-level problem is nonconvex. We compare our method with existing state-of-the-art bi-level optimization methods in three tasks, and all the experimental results demonstrate that our method is superior to the others in terms of accuracy and efficiency.

1 INTRODUCTION

Bi-level optimization (BO) Bard (2013); Colson et al. (2007) plays a central role in various machine learning applications including hyper-parameter optimization Pedregosa (2016); Bergstra et al. (2011); Bertsekas (1976), meta-learning Feurer et al. (2015); Franceschi et al. (2018); Rajeswaran et al. (2019), reinforcement learning Hong et al. (2020); Konda & Tsitsiklis (2000). It involves a competition between two parties or two objectives, and if one party makes its choice first it will affect the optimal choice for the second party. Several approaches (such as Bayesian optimization Klein et al. (2017), random search Bergstra & Bengio (2012), evolution strategy Sinha et al. (2017), gradient-based methods Pedregosa (2016); Maclaurin et al. (2015); Swersky et al. (2014)) have bee proposed to solve BO problems. Among them, gradient-based methods have become the mainstream to solve the large scale BO problems where the size of upper-level variables is tremendous.

Existing gradient-based algorithms can be roughly divided into two categories, i.e., the bi-level and single-level approaches. For the first one, the key idea is to approximate the gradient of the upper-level objective w.r.t upper-level variables, called hypergradient, which can be obtained through implicit differentiation methods Pedregosa (2016); Rajeswaran et al. (2019) or explicit differentiation methods based on chain rule Maclaurin et al. (2015); Domke (2012); Franceschi et al. (2017); Swersky et al. (2014). Specially, the explicit differentiation Franceschi et al. (2017) includes reverse and forward modes which are shorted as RMD and FMD in this paper. Note that both explicit differentiation methods and implicit differentiation methods need intermediate steps, e.g. solving a linear subproblem or using the reverse/forward modes, to approximate the hypergradient. What's worse, they all assume to obtain the solution of the lower-level problem in a fixed iteration for each given upper-level variables to approximate the hypergradient which is impractical. For the single-level approach, the

Table 1: Representative gradient-based bi-level optimization methods. (Here we summarize whether they need to approximate the solutions of the lower-level objective or intermediate steps to approximating the hypergradient, respectively.)

Method	Reference	Problem	Method type	Approximate solutions	Intermediate steps
FMD	Franceschi et al. (2017)	Smooth	Bi-level	Yes	Yes
RMD	Franceschi et al. (2017)	Smooth	Bi-level	Yes	Yes
Approx	Pedregosa (2016)	Smooth	Bi-level	Yes	Yes
Penalty	Mehra & Hamm (2019)	Smooth	Single-level	Yes	Yes
FBBGL	Frecon et al. (2018)	Group LASSO	Bi-level	Yes	Yes
SparseHO	Bertrand et al. (2020)	LASSO-type	Bi-level	Yes	Yes
SMNBP	Okuno & Takeda (2020)	p-norm	Single-level	No	No
SPNBO	Ours	Generalized	Single-level	No	No

key idea is providing a proxy single-level reduction problem, and then deriving the gradient update for the single-level problem instead of the original bi-level problem. For example, Mehra *et al.*, Mehra & Hamm (2019) transformed the original BO problem into a single-level problem by the penalty method and then calculated the gradients for lower- and upper-level variables respectively to update the solution for the single-level reduction problem. We summarize these representative methods in Table 1.

However, most of the existing BO methods are limited to smooth problems as shown in Table 1. In many real-world applications, such as image restoration Chen et al.; Nikolova et al. (2008), variable selection Fan & Li (2001); Huang et al. (2008); Zhang et al. (2010) and signal processing Bruckstein et al. (2009), the lower-level objective may have a complicated non-smooth, perhaps non-Lipschitz term Bian & Chen (2017). To solve this issue, Bertrand *et al*. Bertrand et al. (2020) searched the regularization parameters for LASSO-type problems by approximating the hypergradient from the soft thresholding function Donoho (1995); Bredies & Lorenz (2008); Beck & Teboulle (2009). Frecon *et al.* Frecon et al. (2018) proposed a primal-dual FMD-based method, called FBBGLasso, to search the group structures of group-LASSO problems. In each iteration of updating lower-level variables, it needs an additional loop to calculate the dual variables which are used to approximate the hypergradient and solve Fenchel conjugate of the upper-level objective. Okuno *et al*. Okuno & Takeda (2020) used the smoothing method and sequential quadratic programming (SQP) method Wright & Nocedal (1999) to search the regularization parameter related to q-norm $(0 < q \leq 1)$. To the best of our knowledge, achieving a generalized and scalable optimization algorithm for BO problems with a non-smooth, even non-Lipschitz continuous lower-level objective is still an open question.

To address this challenging problem, in this paper, we propose a new algorithm, called SPNBO, based on smoothing Nesterov (2005); Chen et al. (2013) and penalty Wright & Nocedal (1999) techniques to solve large-scale non-smooth bi-level problems. Specifically, we first use the smoothing technique to approximate the original non-smooth, perhaps non-Lipschitz lower-level problem and generate a sequence of smoothed bi-level problems. Then, single-level constrained problems are obtained by replacing the smoothed lower-level objective with its first-order necessary condition. For each given batch of samples, instead of calculating full gradients for the heavily constrained sub-problem, we randomly sample a constraint and a data sample to obtain a doubly stochastic gradient of the augmented Lagrange function and then update the upper and lower variables alternately. We give new stationary conditions of the single-level constraint problem which are also the stationary conditions of the original problem if the lower-level problem is convex or the necessary conditions of the bilevel problem if the lower-level problem is nonconvex and prove our method can converge to the points satisfying these conditions. We also compare our method with several state-of-the-art bi-level optimization methods in three tasks, and all the experimental results demonstrate that our method is superior to the others in terms of accuracy and efficiency.

Contributions. We summarize the main contributions of this paper as follows:

1. We propose a generalized algorithm to solve non-smooth bi-level problems. Instead of calculating the hypergradients which normally involves intermediate steps or the training of lower-level problem, we utilize the plain doubly stochastic gradients to update the solution which fundamentally improves the efficiency and scalability of our method.

2. We give the stationary conditions of the single-level constraint problem which are also the stationary conditions of the original problem if the lower-level problem is convex or the necessary conditions of the bilevel problem if the lower-level problem is nonconvex and prove that our proposed method can converge to a stationary point. To the best of our knowledge, this is the first theoretically guaranteed method for the bi-level problem with a non-smooth, perhaps non-Lipschitz, lower-level objective.

2 PRELIMINARIES

2.1 FORMULATION OF NON-SMOOTH BI-LEVEL OPTIMIZATION PROBLEM

In this paper, we consider the following generalized non-smooth bi-level optimization problem:

$$\min_{\boldsymbol{\lambda}} \ f(\boldsymbol{w}^*, \boldsymbol{\lambda}) \ \ s.t. \ \boldsymbol{w}^* \in \arg\min_{\boldsymbol{w}} g(\boldsymbol{w}, \bar{\boldsymbol{\lambda}}) + \exp(\lambda_1)\varphi(h(\boldsymbol{w})), \tag{1}$$

where $\boldsymbol{\lambda} := [\lambda_1, \lambda_2, \cdots, \lambda_m]^T \in \mathbb{R}^m$, $\bar{\boldsymbol{\lambda}} := [\lambda_2, \cdots, \lambda_m]^T$ and $\boldsymbol{w} \in \mathbb{R}^d$. $h(\cdot) : \mathbb{R}^d \mapsto \mathbb{R}^n$ is continuous, non-convex, non-smooth, and perhaps non-Lipschitz continuous at some points. $\varphi(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$ is twice continuously differentiable. $f : \mathbb{R}^d \times \mathbb{R}^m \mapsto \mathbb{R}$ and $g : \mathbb{R}^d \times \mathbb{R}^m \mapsto \mathbb{R}$ are twice continuously differentiable regarding to both $\boldsymbol{w}$ and $\boldsymbol{\lambda}$. Suppose that $h(\boldsymbol{w})$ can be represented as $h(\boldsymbol{w}) := (h_1(\boldsymbol{D}_1^T\boldsymbol{w}), h_2(\boldsymbol{D}_2^T\boldsymbol{w}), \cdots, h_n(\boldsymbol{D}_n^T\boldsymbol{w}))$, where $\boldsymbol{D}_i \in \mathbb{R}^{d\times r}$ and $h_i : \mathbb{R}^d \mapsto \mathbb{R}$ $(i = 1, 2, \cdots, n)$ is continuous, but not necessarily Lipschitz continuous Bian & Chen (2017).

2.2 EXAMPLES OF NON-SMOOTH LOWER-LEVEL PROBLEMS

The non-smooth lower-level problems in problem (1) widely exist in machine learning since we usually introduce a non-smooth function to utilize some kind of prior structural information Auslender (1997). Let $\{\boldsymbol{x}_i, y_i\}_{i=1}^{N_{tr}}$ denotes a training set, where $\boldsymbol{x}_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$, and N_{tr} is the size of training samples. We give three examples of the non-smooth, perhaps non-Lipschitz lower-level objectives as follows.

1. **Group LASSO:** The objective function of group LASSO Meier et al. (2008); Simon et al. (2013); Scardapane et al. (2017) is formulated as $\min_{\boldsymbol{w}} \exp(\hat{\lambda}) \sum_{g=1}^{G} \|\boldsymbol{w}_{\mathcal{G}_g}\|_2 + \frac{1}{\sum_{i=1} \exp(\lambda_i)} \sum_{i=1}^{N_{tr}} \lambda_i \left(y_i - \boldsymbol{x}_i^T\boldsymbol{w}\right)^2$, where λ_i denotes the weight for each sample and $\hat{\lambda}$ denotes the regularization parameter. $\boldsymbol{w}_{\mathcal{G}_g}$ denotes the parameters belonging to a group $\mathcal{G}_g$, and the term $\sum_{g=1}^{G} \|\boldsymbol{w}_{\mathcal{G}_g}\|_2$ enforces sparsity at the group level.
2. **p-norm optimization problem:** The objective function of regression problem with p-norm Gentile (2003) is formulated as $\min_{\boldsymbol{w}} \exp(\hat{\lambda})\|\boldsymbol{w}\|_p^p + \frac{1}{N_{tr}} \sum_{i=1}^{N_{tr}} \left(y_i - \boldsymbol{x}_i^T\boldsymbol{w}\right)^2$, where $\hat{\lambda}$ denotes the regularization parameter. $\|\boldsymbol{w}\|_p = \left(\sum_{w_i \in \boldsymbol{w}} |w_i|^p\right)^{1/p}$ $(0 < p \leq 1)$ is introduced to achieve a desirable robustness. Note this objective is non-Lipschitz continuous Bian & Chen (2017).
3. **OSCAR** Bondell & Reich (2008): The objective function of OSCAR is $\min_{\boldsymbol{w}} \frac{1}{2} \sum_{i=1}^{N_{tr}} \left(y_i - \boldsymbol{x}_i^T\boldsymbol{w}\right)^2 + \exp(\lambda_1)\|\boldsymbol{w}\|_1 + \exp(\lambda_2) \sum_{j<k} \max\{|\boldsymbol{w}_j|, |\boldsymbol{w}_k|\}$, where λ_1 and λ_2 denote regularization parameters. $\|\boldsymbol{w}\|_1$ and $\sum_{j<k} \max\{|\boldsymbol{w}_j|, |\boldsymbol{w}_k|\}$ are used to achieve capturing the feature groups adaptively.

3 SMOOTHING AND PENALTY METHOD FOR NON-SMOOTH BI-LEVEL PROBLEM

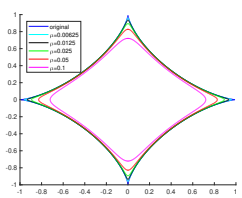
We first introduce the smoothing technique, then give our single-level reduction problem by utilizing the smoothing and penalty methods, finally propose our doubly stochastic gradient algorithm.

3.1 SMOOTHING TECHNIQUE

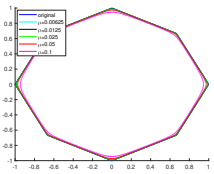
To tackle the non-smooth bi-level problem (1), we use the smoothing function Nesterov (2005); Chen et al. (2013); Bian & Chen (2017) (please see Definition 1) to approximate the original non-smooth objective.

Definition 1 *Let $\psi : \mathbb{R}^n \mapsto \mathbb{R}$ be a continuous nonsmooth function. We call $\tilde{\psi} : \mathbb{R}^n \times [0, +\infty] \mapsto \mathbb{R}$ a smoothing function of ψ, if $\tilde{\psi}(\cdot, \mu)$ is continuously differentiable for any fixed $\mu > 0$ and $\lim_{\hat{\boldsymbol{z}}\mapsto\boldsymbol{z}, \mu\to 0} \tilde{\psi}(\hat{\boldsymbol{z}}, \mu) = \psi(\boldsymbol{z})$ holds for any $\boldsymbol{z} \in \mathbb{R}^n$.*

According to Definition 1, the original non-smooth problem could be approximated by its smoothing function. If the smoothing parameter μ approaches 0, the smoothing function is asymptotically equal to the original non-smooth problem. Thus, we could optimize the smoothed proxy problem, instead of the original non-smooth bi-level problem.



(a) $\psi_2(\boldsymbol{w}) = 1$, where $\psi_2(\boldsymbol{w})$ is p-norm regularization term and $p = 0.6$.



(b) $\psi_3(\boldsymbol{w}) = 1$, where $\psi_3(\boldsymbol{w})$ is a combination of l_1-norm and pair-wise l_∞-norm used in OSCAR.

Figure 1: Two examples of smoothing functions.

Based on the definition, we can give the smoothing functions of the non-smooth terms in section 2.2. For example, the smoothing function of $\psi_1(\boldsymbol{w}) = \sum_{g=1}^{G} \|\boldsymbol{w}_{\mathcal{G}_g}\|_2$ is $\tilde{\psi}_1(\boldsymbol{w}, \mu) = \sum_{g=1}^{G} \sqrt{\|\boldsymbol{w}_{\mathcal{G}_g}\|_2^2 + \mu^2}$, the smoothing function of $\psi_2(\boldsymbol{w}) = \|\boldsymbol{w}\|_p$ is $\tilde{\psi}_2(\boldsymbol{w}, \mu) = \sum_{w_i \in \boldsymbol{w}} (w_i^2 + \mu^2)^{p/2}$ and the smoothing function of $\psi_3(\boldsymbol{w}) = \lambda_1 \|\boldsymbol{w}\|_1 + \lambda_2 \sum_{j<k} \max\{|\boldsymbol{w}_j|, |\boldsymbol{w}_k|\}$ is $\tilde{\psi}_3(\boldsymbol{w}, \mu) = \lambda_1 \sum_{w_i \in \boldsymbol{w}} \sqrt{w_i^2 + \mu^2} + \frac{\lambda_2}{2} \sum_{j<k} (|w_j| + |w_k| + \sqrt{(|w_j| - |w_k|)^2 + \mu^2})$. Assume $p = 0.6$, $\boldsymbol{w} \in \mathbb{R}^2$ and $\lambda_1 = \lambda_2 = 1$. Then we illustrate $\psi_2(\boldsymbol{w})$ and $\psi_3(\boldsymbol{w})$ together with their smoothing functions using different smoothing parameters in Figure 1.

3.2 Penalty Method for Smoothed Bi-level Optimization

The smoothing function of $h(\boldsymbol{w})$ can be defined as $\tilde{h}(\boldsymbol{w}, \mu) := (\tilde{h}_1(\boldsymbol{D}_1^T \boldsymbol{w}, \mu), \tilde{h}_2(\boldsymbol{D}_2^T \boldsymbol{w}, \mu), \cdots, \tilde{h}_n(\boldsymbol{D}_n^T \boldsymbol{w}, \mu))$, where $\tilde{h}_i : \mathbb{R}^d \times [0, +\infty] \mapsto \mathbb{R}$ is the smoothing function of h_i. Then, for each given μ^k, we can get the smoothed bi-level sub-problem as follows, $\min_{\boldsymbol{\lambda}} f(\boldsymbol{w}^*, \boldsymbol{\lambda})$ $s.t.$ $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} g(\boldsymbol{w}, \bar{\boldsymbol{\lambda}}) + \exp(\lambda_1)\varphi(\tilde{h}(\boldsymbol{w}, \mu^k))$. Further, we can replace the smoothed lower-level objective with its first-order necessary condition and derive the following single-level problem: $\min_{\boldsymbol{w}, \boldsymbol{\lambda}} f(\boldsymbol{w}, \boldsymbol{\lambda})$ $s.t.$ $c^{\mu^k}(\boldsymbol{w}, \boldsymbol{\lambda}) := \nabla_{\boldsymbol{w}} g(\boldsymbol{w}, \bar{\boldsymbol{\lambda}}) + \exp(\lambda_1) \nabla_{\boldsymbol{w}} \varphi(\tilde{h}(\boldsymbol{w}, \mu^k)) = \boldsymbol{0}$, where $\nabla_{\boldsymbol{w}} \varphi(\tilde{h}(\boldsymbol{w}, \mu)) = \varphi'(z)_{z=h(\boldsymbol{w},\mu)} \nabla_{\boldsymbol{w}} h(\boldsymbol{w}, \mu)$.

As mentioned in Wright & Nocedal (1999), the penalty method can be used to solve the above sub-problem. Because the original non-smooth bi-level problem involves a sequence of sub-problems, simply using the penalty method on each sub-problem would be time-consuming. Besides, solving the quadratic penalty function needs the penalty parameter to be large enough, which makes it impossible to get a solution in a limited time.

To solve each sub-problem for each given μ^k in a limited time, we calculate the ϵ_k-optimal solution, instead of the exact solution, of the following augmented Lagrange function with a penalty parameter $\beta^k > 0$,

$$\min_{\boldsymbol{w}, \boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\alpha}, \beta^k, \mu^k) = f(\boldsymbol{w}, \boldsymbol{\lambda}) + \Psi(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\alpha}, \beta^k, \mu^k), \tag{2}$$

where $\Psi(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\alpha}, \beta^k, \mu^k) = \frac{1}{d} \sum_{i=1}^{d} (\alpha_i c_i^{\mu^k}(\boldsymbol{w}, \boldsymbol{\lambda}) + \frac{\beta^k}{2} c_i^{\mu^k}(\boldsymbol{w}, \boldsymbol{\lambda})^2)$, $\boldsymbol{\alpha} \in \mathbb{R}^d$ denotes the Lagrangian multiplier, α_i and $c_i^{\mu^k}(\boldsymbol{w}, \boldsymbol{\lambda})$ denote the i-th elements of $\boldsymbol{\alpha}$ and $c^{\mu^k}(\boldsymbol{w}, \boldsymbol{\lambda})$, respectively.

Once the following tolerance conditions are satisfied (which means the ϵ_k-stationary point is found),

$$\|\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\alpha}, \beta^k, \mu^k)\|_2^2 + \|\nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\alpha}, \beta^k, \mu^k)\|_2^2 \leq \epsilon_{1,k}^2, \tag{3}$$

$$|\Psi(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\alpha}, \beta^k, \mu^k)| \leq \epsilon_{2,k}^2, \tag{4}$$

$$\|c^{\mu^k}(\boldsymbol{w}, \boldsymbol{\lambda})\|_2^2 \leq \epsilon_{3,k}^2, \tag{5}$$

we update the Lagrangian multiplier $\boldsymbol{\alpha}^{k+1} = \boldsymbol{\alpha}^k + \beta^k c^{\mu^k}$, enlarge the penalty parameter β^k, and decrease the smooth parameter μ^k and the tolerance $\epsilon_{i,k}$ $(i = 1, \cdots, 3)$. Here, $\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\alpha}, \beta^k, \mu^k)$ and $\nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\alpha}, \beta^k, \mu^k)$ denote the full gradients of the augmented Lagrange function.

3.3 Doubly Stochastic Gradient Method

In many real-world applications, we may need to deal with high dimensional data or use complex models (such as deep models), which will leads to a large number of constraints when replace

the lower-level with its first order conditions. This makes directly solving the problem 2 is time-consuming.

Algorithm 1 Smoothing and Penalty Method for Non-smooth Bi-level Optimization (SPNBO)

Input: $K, T, \eta_{\boldsymbol{w}}^1, \eta_{\boldsymbol{\lambda}}^1, \boldsymbol{\alpha}^1, \mu^1 = 0.001, \beta^1 = 1$ and $\epsilon_{i,1} = 0.01$ for $i = 1, \cdots, 3$.
Output: $\boldsymbol{w}^{t+1,k}$ and $\boldsymbol{\lambda}^{k+1}$.

for $k = 1, ..., K$ **do**
  for $t = 1, \cdots, T$ **do**
    Randomly sample a upper-level data instance.
    Randomly sample a constraint.
    Calculate the doubly stochastic gradient $\hat{\nabla}_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\alpha}, \beta^k, \mu^k)$.
    Update $\boldsymbol{w}$ using $\boldsymbol{w}^{t+1,k} = \boldsymbol{w}^{t,k} - \eta_{\boldsymbol{w}}^t \hat{\nabla}_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\alpha}, \beta^k, \mu^k)$.
  end for
  Randomly sample a validation training data instance .
  Randomly sample a constraint.
  Calculate the doubly stochastic gradient $\hat{\nabla}_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\alpha}, \beta^k, \mu^k)$.
  Update $\boldsymbol{\lambda}$ using $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k - \eta_{\boldsymbol{\lambda}}^t \hat{\nabla}_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\alpha}, \beta^k, \mu^k)$.
  if satisfying the tolerance conditions (3)-(5) **then**
    $\boldsymbol{\alpha}^{k+1} = \boldsymbol{\alpha}^k + \beta^k c^{\mu_k}(\boldsymbol{w}^{t+1,k}, \boldsymbol{\lambda}^{k+1})$.
    $\mu^{k+1} = \mu^k / 2$.
    $\epsilon_{i,k+1} = \epsilon_{i,k}/2$ for $i = 1, \cdots, 3$.
    $\beta^{k+1} = 2\beta^k$.
  end if
end for

To solve this problem, the stochastic manner can be introduced. Specifically, instead of using all the constraints, we randomly sample a constraint $c_i^{\mu^k}(\boldsymbol{w}, \boldsymbol{\lambda})$ and calculate its gradient w.r.t. $\boldsymbol{w}$ and $\boldsymbol{\lambda}$, *i.e.,* $\nabla_{\boldsymbol{w}} c_i^{\mu^k}(\boldsymbol{w}, \boldsymbol{\lambda})$ and $\nabla_{\boldsymbol{\lambda}} c_i^{\mu^k}(\boldsymbol{w}, \boldsymbol{\lambda})$. By using this method, we only need to calculate the gradient of the chosen item in c^{μ^k} instead of calculating the Hessian matrix of the lower-level objective. Let $\hat{\nabla}_{\boldsymbol{w}} \Psi = [\alpha_i + \beta^k c_i^{\mu^k}(\boldsymbol{w}, \boldsymbol{\lambda})] \nabla_{\boldsymbol{w}} c_i^{\mu^k}(\boldsymbol{w}, \boldsymbol{\lambda})$ and $\hat{\nabla}_{\boldsymbol{\lambda}} \Psi = [\alpha_i + \beta^k c_i^{\mu^k}(\boldsymbol{w}, \boldsymbol{\lambda})] \nabla_{\boldsymbol{\lambda}} c_i^{\mu^k}(\boldsymbol{w}, \boldsymbol{\lambda})$. If d is sufficient large, $\hat{\nabla}_{\boldsymbol{w}} \Psi$ and $\hat{\nabla}_{\boldsymbol{\lambda}} \Psi$ can be viewed as the unbiased estimations of the full gradients $\nabla_{\boldsymbol{w}} \Psi$ and $\nabla_{\boldsymbol{\lambda}} \Psi$ respectively.

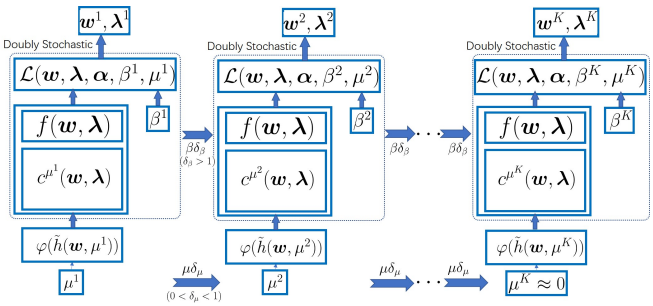


Figure 2: Illustration of proposed method.

In addition, since the upper-level objective f is usually formulated as the expectation on the upper-level data set, we can randomly sample a upper-level data point and calculate the stochastic gradients of f w.r.t. $\boldsymbol{w}$ and $\boldsymbol{\lambda}$, which are denoted as $\hat{\nabla}_{\boldsymbol{w}} f(\boldsymbol{w}, \boldsymbol{\lambda})$ and $\hat{\nabla}_{\boldsymbol{\lambda}} f(\boldsymbol{w}, \boldsymbol{\lambda})$ respectively.

Then, by combining these stochastic gradients together, we can obtain the stochastic gradients of the augmented Lagrange function as $\hat{\nabla}_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\alpha}, \beta^k, \mu^k) = \hat{\nabla}_{\boldsymbol{w}} f(\boldsymbol{w}, \boldsymbol{\lambda}) + \hat{\nabla}_{\boldsymbol{w}} \Psi$ and $\hat{\nabla}_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\alpha}, \beta^k, \mu^k) = \hat{\nabla}_{\boldsymbol{\lambda}} f(\boldsymbol{w}, \boldsymbol{\lambda}) + \hat{\nabla}_{\boldsymbol{w}} \Psi$. Since the stochastic gradient has two sources of randomness, it is called as doubly stochastic gradient in this paper. Following the work of Mehra & Hamm (2019), we update $\boldsymbol{w}$ for fixed T iterations and then update $\boldsymbol{\lambda}$ for a single time using the doubly stochastic gradient as follows, $\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \eta_{\boldsymbol{w}}^t \hat{\nabla}_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}^t, \boldsymbol{\lambda}, \boldsymbol{\alpha}, \beta^k, \mu^k)$ and $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k - \eta_{\boldsymbol{\lambda}}^k \hat{\nabla}_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}^k, \boldsymbol{\alpha}, \beta^k, \mu^k)$, where $\eta_{\boldsymbol{w}}^t$ and $\eta_{\boldsymbol{\lambda}}^t$ denote the step sizes.

The whole algorithm is presented in Algorithm 1. In addition, we give an illustration of our method in Figure 2. Note we can sample a batch of constraints and upper-level data points to update $\boldsymbol{w}$ and $\boldsymbol{\lambda}$. Finding the ϵ_k-optimal solution allows us starting from large tolerance parameters $\epsilon_{1,0}$, $\epsilon_{2,0}$ and $\epsilon_{3,0}$.

4 THEORETICAL ANALYSIS

In this section, we give convergence analysis of our proposed method (the details can be found in the supplement). First, we give several assumptions which are commonly used in the convergence analysis for optimization algorithms Bian & Chen (2017); Clarke (1990).

Assumption 1 *Assume that $\mathcal{W} := \mathbb{R}^d$ can be expressed by $\mathcal{W} = \mathcal{W}_1 \cap \mathcal{W}_2$ with a nonempty close convex set $\mathcal{W}_1$ and $\mathcal{W}_2 := \{\boldsymbol{w} : \boldsymbol{A}\boldsymbol{w} \leq \boldsymbol{b}\}$ where $\mathrm{int}(\mathcal{W}_1) \cap \mathcal{W}_2 \neq \emptyset$, $\boldsymbol{A} \in \mathbb{R}^{t\times d}$, $\boldsymbol{b} \in \mathbb{R}^t$ and $\mathrm{int}(\cdot)$ denote the interior set.*

Assumption 2 *The smoothing function $\tilde{h}(\boldsymbol{w}, \mu)$ is twice continuously differentiable on $\boldsymbol{w}$.*

Assumption 3 *g and f are both Lipschitz continuous.*

Based on Assumptions 1-3, we given the definition of the stationary point of the generalized non-smooth bi-level optimization problem (1) as follows.

Definition 2 *$(\boldsymbol{w}^*, \boldsymbol{\lambda}^*)$ is said to be a stationary point of problem (1), if it satisfies the following conditions for all $\boldsymbol{v}_1 \in \mathcal{T}_{\mathcal{W}}(\boldsymbol{w}^*) \cap \mathcal{V}_{\boldsymbol{w}^*}$, $\boldsymbol{v}_2 \in \mathcal{T}_{\mathcal{W}}(\boldsymbol{w}^*)$ and $\boldsymbol{v}_3 \in \mathcal{T}_{\mathcal{U}}(\boldsymbol{\lambda}^*)$, where $\mathcal{U} = \mathbb{R}^m$ and the lower-level problem is convex,*

$$\nabla_{\boldsymbol{w}} f(\boldsymbol{w}^*, \boldsymbol{\lambda}^*)^T \boldsymbol{v}_2 - \left(\boldsymbol{v}_2^T \nabla^2_{\boldsymbol{w}\boldsymbol{w}} g(\boldsymbol{w}^*, \bar{\boldsymbol{\lambda}}^*)\boldsymbol{v}_1 + \exp(\lambda_1^*)\phi^{\circ\circ}(\boldsymbol{w}^*; \boldsymbol{v}_1, \boldsymbol{v}_2; \mathcal{W})\right)\xi^* \geq 0 \tag{6}$$

$$\nabla_{\boldsymbol{\lambda}} f(\boldsymbol{w}^*, \boldsymbol{\lambda}^*)^T \boldsymbol{v}_3 - \left(\bar{\boldsymbol{v}}_3 \nabla^2_{\boldsymbol{w}\bar{\boldsymbol{\lambda}}} g(\boldsymbol{w}^*, \bar{\boldsymbol{\lambda}}^*)\boldsymbol{v}_1 + v_3^1 \exp(\lambda_1^*)\phi^{\circ}(\boldsymbol{w}^*; \boldsymbol{v}_1; \mathcal{W})\right)\xi^* \geq 0 \tag{7}$$

$$\nabla_{\boldsymbol{w}} g(\boldsymbol{w}^*, \bar{\boldsymbol{\lambda}}^*)^T \boldsymbol{v}_1 + \exp(\lambda_1^*)\phi^{\circ}(\boldsymbol{w}^*; \boldsymbol{v}_1; \mathcal{W}) \geq 0 \tag{8}$$

where $\xi^* \geq 0$, $\boldsymbol{v}_3 = [v_3^1, \bar{\boldsymbol{v}}_3^T]^T$, $\phi^{\circ\circ}(\boldsymbol{w}^*; \boldsymbol{v}_1, \boldsymbol{v}_2; \mathcal{W}) = \limsup_{\substack{\boldsymbol{w} \mapsto \boldsymbol{w}^*, \boldsymbol{w} \in \mathcal{W} \\ s \downarrow 0, \boldsymbol{w} + s\boldsymbol{v}_2 \in \mathcal{W}}} \frac{\phi^{\circ}(\boldsymbol{w} + \boldsymbol{v}_2 s; \boldsymbol{v}_1; \mathcal{W}) - \phi^{\circ}(\boldsymbol{w}; \boldsymbol{v}_1; \mathcal{W})}{s}$). *and* $\phi^{\circ}(\boldsymbol{w}; \boldsymbol{v}_1; \mathcal{W}) = \limsup_{\substack{\boldsymbol{w}' \mapsto \boldsymbol{w}, \boldsymbol{w}' \in \mathcal{W} \\ t \downarrow 0, \boldsymbol{w}' + t\boldsymbol{v} \in \mathcal{W}}} \frac{\varphi(h(\boldsymbol{w}' + t\boldsymbol{v})) - \varphi(h(\boldsymbol{w}'))}{t}$ *denotes the Clarke generalized directional derivative of $\varphi(h(\boldsymbol{w}))$ at point $\boldsymbol{w}$. Note if the lower-level problem is nonconvex, conditions 6-8 is the necessary conditions of the original nonsmooth, perhaps non-Lipschitz bilevel problem.*

Assume Assumptions 1-3 hold, our proposed method has the following convergence result.

Theorem 1 *Suppose $\{\epsilon_{i,k}\}_{k=1}^{\infty}$ $(i = 1, 2, 3)$ are positive and convergent $(\lim_{k\to\infty} \epsilon_{i,k} = 0)$ sequences, $\{\mu^k\}_{k=1}^{\infty}$ is a positive and convergent $(\lim_{k\to\infty} \mu^k = 0)$ sequence, and β^k is increasing and divergent $(\beta^1 < \beta^2 < \cdots)$. Then any limit point of the sequence points generated by SPNBO satisfies the conditions (6)-(8).*

Remark 1 *Theorem 1 shows that with the increasing of the penalty parameter and decreasing of the smoothing parameter and tolerance parameters, our method can finally converge to a stationary point of the original non-smooth bi-level problem if the lower-level objective is convex. If the lower-level problem is nonconvex, the solutions satisfy the necessary conditions of the original problem.*

5 EXPERIMENTS

In this section, we conduct experiments to demonstrate the superiority of our method in terms of accuracy and efficiency in three applications.

5.1 APPLICATIONS

We give a brief introduction of the three applications (*i.e.*, data re-weight, training data poisoning and meta-learning) used in our experiments.

Data re-weight: In many real-world applications, the training set and testing set may have different distributions. To reduce the discrepancy between the two distributions, each data point will be given an additional importance weight, which is called data re-weight. In this application, we search the weight λ_i of each training data and the group sparse regularization parameters Scardapane et al. (2017) $\hat{\lambda}$ and $\check{\lambda}$ for deep neural networks (DNNs). It can be formulated

as $\min_{\boldsymbol{\lambda}} 1/N_{val} \sum_{i=1}^{N_{val}} l(\theta(\boldsymbol{x}_i; \boldsymbol{w}), \boldsymbol{y}_i)$ $s.t.$ $\boldsymbol{w}^* \in \arg\min_{\boldsymbol{w}} 1/N_{tr} \sum_{i=1}^{N_{tr}} \exp(\lambda_i) l(\theta(\boldsymbol{x}_i), \boldsymbol{y}_i) + \exp(\hat{\lambda}) \|\boldsymbol{w}\|_1^1 + \exp(\check{\lambda}) \sum_{\mathcal{G}_i} \|\boldsymbol{w}_{\mathcal{G}_i}\|_2$, where N_{tr} and N_{val} denote the sizes of training set and validation set respectively, $\theta(\cdot; \boldsymbol{w})$ denotes the DNN parameterized by $\boldsymbol{w}$, $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}$ denotes the data instance, $\mathcal{G}_i$ denotes the group index and l denotes the loss function. Besides, in DNNs, model parameters are grouped by layers.

Training data poisoning: Assume we have pure training data $\{\boldsymbol{x}_i\}_{i=1}^{N_{tr}}$ with several poisoned points $\{\boldsymbol{\lambda}_j\}_{j=1}^{N_{poi}}$ assigned arbitrary labels. In this task, we search the poisoned data which can hurt the performance of the model trained from the clean data. This problem can be formulated as $\min_{\boldsymbol{\lambda}} -1/N_{vl} \cdot \sum_{i=1}^{N_{val}} l(\theta(\boldsymbol{x}_i; \boldsymbol{w}), \boldsymbol{y}_i)$ $s.t.$ $\boldsymbol{w}^* \in \arg\min_{\boldsymbol{w}} \min_{\boldsymbol{w}} 1/N \cdot \sum_{\boldsymbol{x}_i \in \mathcal{D}} l(\theta(\boldsymbol{x}_i; \boldsymbol{w}), \boldsymbol{y}_i) + \|\boldsymbol{w}\|_p^p$, where $N = N_{tr} + N_{poi}$ and $\mathcal{D}$ denotes the dataset containing all the clean training data and poisoned data. Besides, we add a p-norm $(0 < p \leq 1)$ regularization term in the lower-level problem to ensure that we can get a sparse model.

Meta-learning: Meta-learning Ravi & Larochelle (2016); Snell et al. (2017); Sung et al. (2018); Santoro et al. (2016) trains a model on several related tasks and then generalizes to unseen tasks with just a few examples. We can learn a common representation for various tasks and then train the task specific layers. It can be formulated as the non-smooth bi-level problem, $\min_{\boldsymbol{\lambda}} 1/N_{val} \cdot \sum_{i=1}^{N_{val}} l(\theta_i(M(\boldsymbol{x}_i, \boldsymbol{\lambda}); \boldsymbol{w}_i^*), \boldsymbol{y}_i)$ $s.t.$ $\boldsymbol{w}_i^* \in \arg\min_{\boldsymbol{w}_i} 1/N_{tr} \cdot \sum_{i=1}^{N_{tr}} l(\theta_i(M(\boldsymbol{x}_i, \boldsymbol{\lambda}); \boldsymbol{w}_i^*), \boldsymbol{y}_i) + \|\boldsymbol{w}_i\|_p^p$, where $M(\cdot, \boldsymbol{\lambda})$ is the deep map for all tasks parameterized by $\boldsymbol{\lambda}$, θ_i denotes ith task's classifier parameterized by $\boldsymbol{w}_i$ and $0 < p \leq 1$. Besides, a p-norm $(0 < p \leq 1)$ is added on the parameters of each classifier to get sparse classifiers.

5.2 Experimental Setup

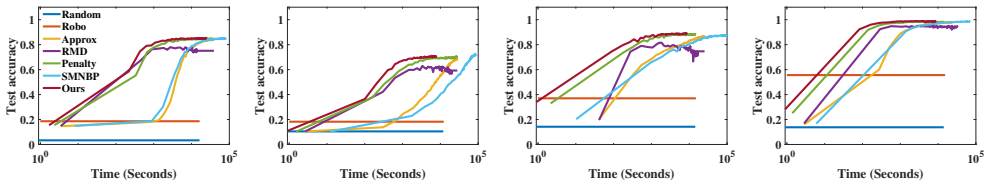
We summarize the baseline methods used in our experiments as follows.

1. **Random search** Bergstra & Bengio (2012). It randomly samples upper-level parameters from the given domain and then evaluate the performance of the corresponding lower-level parameters.
2. **Robo**. The robust Bayesian optimization method proposed in Klein et al. (2017). We use the code from https://github.com/automl/RoBO as the implementation.
3. **Penalty**. The method proposed in Mehra & Hamm (2019). It formulates the bi-level optimization problem as a one-level optimization problem, and then uses the gradient method to solve the new problem.
4. **Approx**. The method proposed in Pedregosa (2016). It solves an additional linear problem to find the hypergradient to update the hyper-parameters.
5. **RMD**. The reverse method proposed in Franceschi et al. (2017). An additional loop is used to approximate the hypergradient.
6. **SMNBP**. The method proposed in Okuno & Takeda (2020). It uses the smoothing method to produce a sequence of smoothing lower-level functions and replaces them with the necessary condition. Then the SQP method is used to solve each single level problem.

Table 2: Datasets used in the experiments.

Datasets	Features	Samples	Classes
SVHN	$32 \times 32 \times 3$	73257	10
Cifar10	$84 \times 84 \times 3$	581012	10
Mnist	$28 \times 28 \times 1$	60000	10
Fashion	$28 \times 28 \times 1$	60000	10
Miniimagenet	$84 \times 84 \times 3$	60000	100
Omnglot	$28 \times 28 \times 1$	81150	1623

We implement random search, SMNBP, Penalty, Approx, RMD, and our method in Python. For random search and Robo, each lower-level variable is chosen from $[e^{-10}, e^{10}]$. Besides, we solve the lower-level problem for 20 epochs by using the sub-gradient method for given upper-level variables. After searching 100 times, we get best upper-level variables and use them to re-solve the lower-level problem and evaluate the performance. For Penalty, Approx and RMD, a smoothing function with parameter $\mu = 1e^{-3}$ is used to approximate the lower-level objective, such that these methods can be used for non-smooth bi-level problems. In SMNBP, for each given batch data, we solve the constrained problem using the SQP method. We initialize the step size of updating $\boldsymbol{w}$ from the set $\{0.01, 0.001, 0.0001\}$. The step-size of updating $\boldsymbol{\lambda}$ is fixed at 1. We fix the inner iteration number of T in Penalty, Approx, RMD and our SPNBO at 20. For all these methods, we fix the data batch size at 128 in the former two applications. For all applications, we use a DNN model which has 6 convolution-maxpooling-relu layers and 4

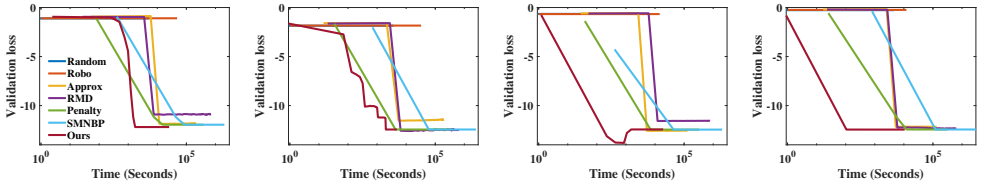


(a) Svhn (b) Cifar10 (c) Fashion (d) Mnist

Figure 3: Test accuracy versus training time of all the methods in data re-weight.

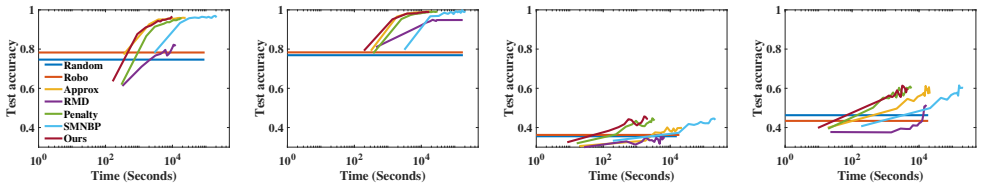
Table 3: Test accuracy (%) of all the methods in data re-weight.

Data	Random	Robo	Approx	RMD	Penalty	SMNBP	Ours
Svhn	3.32 ± 0.43	18.61 ± 0.16	84.06 ± 0.15	75.48 ± 0.92	84.26 ± 0.54	84.92 ± 0.24	**85.21** ± 0.59
Cifar10	10.53 ± 0.13	18.52 ± 0.22	68.65 ± 0.52	59.26 ± 1.33	71.08 ± 0.44	71.81 ± 0.63	**72.12** ± 0.56
Fashion	14.12 ± 0.24	36.98 ± 0.67	86.56 ± 0.27	73.60 ± 1.57	88.18 ± 0.13	88.22 ± 0.59	**88.52** ± 0.23
Mnist	13.79 ± 0.32	55.61 ± 0.28	97.85 ± 0.19	94.52 ± 0.38	98.31 ± 0.65	98.49 ± 0.04	**98.53** ± 0.23



(a) Svhn (b) Cifar10 (c) Fashion (d) Mnist

Figure 4: Validation loss versus training time of all the methods in training data poisoning.



(a) Omnglot 5-way 1-shot (b) Omnglot 5-way 5-shot (c) Mini 5-way 1-shot (d) Mini 5-way 1-shot

Figure 5: Test accuracy versus training time of all the methods in meta-learning, where Mini denotes the dataset Miniimagenet.

dense layers. Besides, in meta-learning, the dense layers are viewed as classifiers for specific tasks. In data re-weight and training data poisoning, our method randomly samples 4 layers to calculate the doubly stochastic gradient in each iteration. And we run Penalty, Approxm, RMD, SMNBP and our SPNBO for 50 epochs. In meta-learning, we randomly sample 2 layers of each classifier to calculate the doubly stochastic gradient. And we run Penalty, Approxm, RMD, SMNBP and our SPNBO for 5000 iterations. We fix $p = 0.6$ in training data poisoning and meta-learning. All experiments are carried out 10 times on a PC with four 1080 Ti GPUs.

5.3 Datasets

We summarize the image datasets used in our experiments in Table 2. For the first four datasets, we divide all of them into three parts, *i.e.*, 40% for the training set, 40% for the validation set and 20% the testing set. The last two datasets are used in meta-learning application.

5.4 Results and Discussion

The results of data re-weight are presented in Table 3 and Figure 3. The results of training data poisoning are presented in Table 4 and Figure 4 and the results of meta-learning are presented in Table 5 and Figure 5. From Table 3, Table 4 and Table 5, we can find that our proposed method has the best results in most cases. This because with the decreasing of the smoothing parameter, our

Table 4: Test accuracy (%) of all the methods in training data poisoning (lower is better).

Data	Random	Robo	Approx	RMD	Penalty	SMNBP	Ours
Svhn	50.98 ± 0.22	55.29 ± 0.42	50.79 ± 0.39	50.67 ± 0.27	50.67 ± 0.27	50.62 ± 0.29	**48.85** ± 0.57
Cifar10	83.19 ± 0.15	82.73 ± 0.21	82.91 ± 0.18	83.25 ± 0.11	82.29 ± 0.11	82.57 ± 0.11	**82.22** ± 0.28
Fashion	95.99 ± 0.35	96.08 ± 0.21	96.09 ± 0.07	95.89 ± 0.31	95.87 ± 0.19	96.01 ± 0.22	**95.80** ± 0.20
Mnist	81.15 ± 0.45	79.17 ± 0.72	80.27 ± 0.25	77.63 ± 0.08	77.43 ± 0.54	77.50 ± 0.30	**77.22** ± 0.08

Table 5: Test accuracy (%) of all the methods in meta-learning.

Data	Problem	Random	Robo	Approx	RMD	Penalty	SMNBP	Ours
Miniimagent	5-way 1-shot	35.25 ± 0.11	36.23 ± 0.08	39.75 ± 0.04	37.48 ± 0.96	43.98 ± 0.84	44.28 ± 0.31	**44.66** ± 0.13
Miniimagent	5-way 5-shot	45.52 ± 0.39	46.39 ± 0.25	60.75 ± 0.19	49.91 ± 0.23	60.75 ± 0.14	61.13 ± 0.31	**61.28** ± 0.90
Omnglot	5-way 1-shot	75.12 ± 0.89	78.26 ± 0.43	96.08 ± 0.08	81.79 ± 0.57	96.10 ± 0.21	96.63 ± 0.34	**96.70** ± 0.32
Omnglot	5-way 5-shot	77.28 ± 0.18	78.33 ± 0.26	99.12 ± 0.07	94.81 ± 0.58	99.03 ± 0.35	**99.15** ± 0.46	**99.15** ± 0.60

method can finally get the solution of the original non-smooth bi-level problems. For Approx, RMD, and Penalty, they use a fixed smoothing method and only get approximate solutions, which makes them obtain worse performances. Besides, Robo and random search are used to search a small size of upper-level variables. However, in our experiments, we have more than 10000 upper-level variables which makes Robo and random search cannot get the best results. From Figure 3, Figure 4 and Figure 5, we can find that our method is faster than other methods in most cases. This is because that Approx, RMD need to solve the lower-level objective first and then use the solution to approximate the hypergradient with an intermediate step in each iteration. Penalty needs to use all the constraints in each updating step which is also time-consuming. When we use complex models (*e.g.*, DNNs), all these methods suffer from high time complexity. However, our method uses the doubly stochastic gradient method which makes it scalable to complicated model and does not need intermediate steps to approximate the hypergradient. Besides, SMNBP needs to solve each sub-problem using the SQP method, which is time-consuming. For Robo and random search, they need to solve the lower-level problem for the given values of upper-level variables. This makes it non-efficient for large-scale problems. From all these results, we can conclude that our SPNBO is superior to other methods in terms of accuracy and efficiency.

6 Discussion of Gradient-based Bi-level Optimization Methods

In this section, we give a detailed discussion of several gradient-based methods. We consider the smooth bi-level problem $\min_{\boldsymbol{\lambda}} \ f(\boldsymbol{w}^*, \boldsymbol{\lambda})\ s.t.\ \boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \hat{g}(\boldsymbol{w}, \boldsymbol{\lambda})$, where f and $\hat{g}$ are twice continuously differentiable in both $\boldsymbol{w} \in \mathbb{R}^d$ and $\boldsymbol{\lambda} \in \mathbb{R}^m$. Assume the dynamic system in RMD/FMD is $\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \eta \nabla_{\boldsymbol{w}} \hat{g}(\boldsymbol{w}^t, \boldsymbol{\lambda})$, where $t = 1, \cdots, T$ and η denotes the step size. Then, a constrained problem with T constraints is obtained. For RMD, to calculate the hypergradient, they needs to compute $\boldsymbol{p}^{t-1} = \boldsymbol{p}^t - \eta \nabla^2_{\boldsymbol{w}\boldsymbol{\lambda}} \hat{g}(\boldsymbol{w}^t, \boldsymbol{\lambda}) \cdot \boldsymbol{q}^t$ and $\boldsymbol{q}^{t-1} = (\boldsymbol{I} - \eta \nabla^2_{\boldsymbol{w}\boldsymbol{w}} \hat{g}(\boldsymbol{w}^t, \boldsymbol{\lambda}))\boldsymbol{q}^t$, where $\boldsymbol{q}^t \in \mathbb{R}^d$,$\boldsymbol{p}^t \in \mathbb{R}^m, t = T, T-1, \cdots, 1, \boldsymbol{p}^T = \nabla_{\boldsymbol{\lambda}} f(\boldsymbol{w}^T, \boldsymbol{\lambda})$ and $\boldsymbol{q}^T = \nabla_{\boldsymbol{w}} f(\boldsymbol{w}^T, \boldsymbol{\lambda})$. Finally, hypergradient can be obtained by using $\boldsymbol{p}_0$. For FMD method, when update the lower-level variables, it needs to calculate $\boldsymbol{P}^{t+1} = \boldsymbol{P}^t(\boldsymbol{I} - \eta \nabla^2_{\boldsymbol{w}\boldsymbol{w}} \hat{g}(\boldsymbol{w}^t, \boldsymbol{\lambda})) - \eta \nabla^2_{\boldsymbol{w}\boldsymbol{\lambda}} \hat{g}(\boldsymbol{w}^t, \boldsymbol{\lambda})$ at the same time, where $\boldsymbol{P} \in \mathbb{R}^{m \times d}$ and $\boldsymbol{P}^0$ is a zero matrix. Then, the hypergradient can be obtained by using $\nabla_{\boldsymbol{\lambda}} f + \boldsymbol{P}_T \nabla_{\boldsymbol{w}} f$. The approximate gradient method Pedregosa (2016) approximates the hypergradient by solving the linear problem $\min_{\boldsymbol{q}} \|\nabla^2_{\boldsymbol{w}\boldsymbol{w}} \hat{g} \cdot \boldsymbol{q} - \nabla_{\boldsymbol{w}} f\|_2^2$ with gradient method for T iterations. PenaltyMehra & Hamm (2019) replace the lower-level objective with its necessary condition and does not need to compute the hypergradient. It updates $\boldsymbol{w}$ and $\boldsymbol{\lambda}$ by using the gradient of penalty function of the single-level problem. SMNBP Okuno & Takeda (2020) uses smoothing method to generate a sequence of smoothed problems and transforms them into single -level problems. Then, the SQP method can be used to get the solution for each given smoothing parameter. To avoid the calculation of Hessian matrix of the lower-level objective, the Hessian-vector product and auto-differentiable are used.

7 Conclusion

In this paper, we proposed a new method, SPNBO, to solve the generalized non-smooth bi-level optimization problems by using the smoothing method and the penalty method. We also give the convergence analysis of our proposed method. The experimental results demonstrate the superiority of our method in terms of training time and accuracy.

REFERENCES

Alfred Auslender. How to deal with the unbounded in optimization: Theory and algorithms. *Mathematical Programming*, 79(1):3–18, 1997.

Jonathan F Bard. *Practical bilevel optimization: algorithms and applications*, volume 30. Springer Science & Business Media, 2013.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012.

James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pp. 2546–2554, 2011.

Quentin Bertrand, Quentin Klopfenstein, Mathieu Blondel, Samuel Vaiter, Alexandre Gramfort, and Joseph Salmon. Implicit differentiation of lasso-type models for hyperparameter optimization. In *International Conference on Machine Learning*, pp. 810–821. PMLR, 2020.

Dimitri P Bertsekas. On penalty and multiplier methods for constrained minimization. *SIAM Journal on Control and Optimization*, 14(2):216–235, 1976.

Wei Bian and Xiaojun Chen. Optimality and complexity for constrained optimization problems with nonconvex regularization. *Mathematics of Operations Research*, 42(4):1063–1084, 2017.

Howard D Bondell and Brian J Reich. Simultaneous regression shrinkage, variable selection, and supervised clustering of predictors with oscar. *Biometrics*, 64(1):115–123, 2008.

Kristian Bredies and Dirk A Lorenz. Linear convergence of iterative soft-thresholding. *Journal of Fourier Analysis and Applications*, 14(5-6):813–837, 2008.

Alfred M Bruckstein, David L Donoho, and Michael Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM review*, 51(1):34–81, 2009.

X Chen, M Ng, and C Zhang. Nonconvex l_p regularization and box constrained model for image restoration. *IEEE Trans. Image Process*, 21.

Xiaojun Chen, Lingfeng Niu, and Yaxiang Yuan. Optimality conditions and a smoothing trust region newton method for nonlipschitz optimization. *SIAM Journal on Optimization*, 23(3):1528–1552, 2013.

Frank H Clarke. *Optimization and nonsmooth analysis*. SIAM, 1990.

Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals of operations research*, 153(1):235–256, 2007.

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

David L Donoho. De-noising by soft-thresholding. *IEEE transactions on information theory*, 41(3): 613–627, 1995.

Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360, 2001.

Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Initializing bayesian hyperparameter optimization via meta-learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. Forward and reverse gradient-based hyperparameter optimization. *arXiv preprint arXiv:1703.01785*, 2017.

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

Jordan Frecon, Saverio Salzo, and Massimiliano Pontil. Bilevel learning of the group lasso structure. *Advances in Neural Information Processing Systems*, 31:8301–8311, 2018.

Claudio Gentile. The robustness of the p-norm algorithms. *Machine Learning*, 53(3):265–299, 2003.

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

Jian Huang, Joel L Horowitz, Shuangge Ma, et al. Asymptotic properties of bridge estimators in sparse high-dimensional regression models. *Annals of Statistics*, 36(2):587–613, 2008.

A. Klein, S. Falkner, N. Mansur, and F. Hutter. Robo: A flexible and robust bayesian optimization framework in python. In *NIPS 2017 Bayesian Optimization Workshop*, December 2017.

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

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

Akshay Mehra and Jihun Hamm. Penalty method for inversion-free deep bilevel optimization. *arXiv preprint arXiv:1911.03432*, 2019.

Lukas Meier, Sara Van De Geer, and Peter Bühlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):53–71, 2008.

Yu Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1): 127–152, 2005.

Mila Nikolova, Michael K Ng, Shuqin Zhang, and Wai-Ki Ching. Efficient reconstruction of piecewise constant images using nonsmooth nonconvex minimization. *SIAM journal on Imaging Sciences*, 1 (1):2–25, 2008.

Takayuki Okuno and Akiko Takeda. Bilevel optimization of regularization hyperparameters in machine learning. pp. 169–194, 2020.

Fabian Pedregosa. Hyperparameter optimization with approximate gradient. *arXiv preprint arXiv:1602.02355*, 2016.

Aravind Rajeswaran, Chelsea Finn, Sham Kakade, and Sergey Levine. Meta-learning with implicit gradients. *arXiv preprint arXiv:1909.04630*, 2019.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. One-shot learning with memory-augmented neural networks. *arXiv preprint arXiv:1605.06065*, 2016.

Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81–89, 2017.

Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A sparse-group lasso. *Journal of computational and graphical statistics*, 22(2):231–245, 2013.

Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. Evolutionary algorithm for bilevel optimization using approximations of the lower level optimal solution mapping. *European Journal of Operational Research*, 257(2):395–411, 2017.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pp. 4077–4087, 2017.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1199–1208, 2018.

Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014.

Stephen Wright and Jorge Nocedal. Numerical optimization. *Springer Science*, 35(67-68):7, 1999.

Cun-Hui Zhang et al. Nearly unbiased variable selection under minimax concave penalty. *The Annals of statistics*, 38(2):894–942, 2010.