Penalty Method for Inversion-Free Deep Bilevel Optimization

Akshay Mehra AMEHRA@TULANE.EDU
Jihun Hamm JHAMM3@TULANE.EDU
Tulane University, New Orleans, LA, USA

Editors: Vineeth N Balasubramanian and Ivor Tsang

Abstract

Solving a bilevel optimization problem is at the core of several machine learning problems such as hyperparameter tuning, data denoising, meta- and few-shot learning, and training-data poisoning. Different from simultaneous or multi-objective optimization, the steepest descent direction for minimizing the upper-level cost in a bilevel problem requires the inverse of the Hessian of the lower-level cost. In this work, we propose a novel algorithm for solving bilevel optimization problems based on the classical penalty function approach. Our method avoids computing the Hessian inverse and can handle constrained bilevel problems easily. We prove the convergence of the method under mild conditions and show that the exact hypergradient is obtained asymptotically. Our method's simplicity and small space and time complexities enable us to effectively solve large-scale bilevel problems involving deep neural networks. We present results on data denoising, few-shot learning, and training-data poisoning problems in a large-scale setting. Our results show that our approach outperforms or is comparable to previously proposed methods based on automatic differentiation and approximate inversion in terms of accuracy, run-time, and convergence speed.

Keywords: Bilevel optimization; data denoising; few-shot learning; data poisoning.

1. Introduction

Bilevel optimization problems appear in the fields of study involving a competition between two parties or two objectives. Particularly, a bilevel problem arises if one party makes its choice first affecting the optimal choice for the second party, known as the Stackelberg model [von Stackelberg (2010)]. The general form of a bilevel optimization problem is as follows

$$\min_{u \in \mathcal{U}} \; f(u, v^*) \quad \text{s.t.} \;\; v^* = \arg\min_{v \in \mathcal{V}(u)} \; g(u, v) \tag{1}$$

The 'upper-level' problem $\min_{u\in\mathcal{U}} f(u, v^*)$ is a usual minimization problem except that v^* is constrained to be the solution to the 'lower-level' problem $\min_{v\in\mathcal{V}(u)} g(u, v)$ which is in turn dependent on u (see [Bard (2013)] for a review of bilevel optimization). In this work, we propose and analyze a new algorithm for solving bilevel problems based on the classical penalty function approach. We demonstrate the effectiveness of our algorithm on several important machine learning applications including gradient-based hyperparameter tuning [Domke (2012); Maclaurin et al. (2015); Luketina et al. (2016); Pedregosa (2016); Franceschi et al. (2017, 2018); Lorraine et al. (2020)], data denoising by importance learning [Liu and Tao (2016); Yu et al. (2017); Ren et al. (2018); Shu et al. (2019)], meta/few-shot learning [Ravi and Larochelle (2017); Santoro et al. (2016); Vinyals et al. (2016); Franceschi et al. (2017); Mishra et al. (2017); Snell et al. (2017); Franceschi et al. (2018); Rajeswaran et al.

(2019)], and training-data poisoning [Mei and Zhu (2015); Muñoz-González et al. (2017); Koh and Liang (2017); Shafahi et al. (2018)].

Gradient-based hyperparameter tuning. Hyperparameter tuning is essential for any learning problem and grid search is a popular method when domain of the hyperparameters is a discrete set or a range. However, when losses are differentiable functions of the hyperparameter(s), a continuous bilevel optimization problem can help find the optimal hyperparameters. Let u and w be hyperparameter(s) and parameter(s) for a class of learning algorithms, $h(x; u, w)$ be the hypothesis, $L_{\text{val}}(u, w) = \frac{1}{N_{\text{val}}} \sum_{(x_i,y_i)\in\mathcal{D}_{\text{val}}} l(h(x_i; u, w), y_i)$ and $L_{\text{train}}(u, w) = \frac{1}{N_{\text{train}}} \sum_{(x_i,y_i)\in\mathcal{D}_{\text{train}}} l(h(x_i; u, w), y_i)$ be the loss on validation and training sets, respectively. Then the best hyperparameter(s) u is the solution to

$$\min_u \; L_{\text{val}}(u, w^*) \;\; \text{s.t.} \;\; w^* = \arg\min_w \; L_{\text{train}}(u, w). \tag{2}$$

Data denoising by importance learning. Most learning algorithms assume that the training set is an i.i.d. sample from the same distribution as the test set. However, if train and test distributions are not identical or if the training set is corrupted by noise or modified by adversaries, this assumption is violated. In such cases, changing the importance of each training example, before training, can reduce the discrepancy between the two distributions. For example, importance of the examples from the same distribution can be up-weighted in comparison to other examples. Determining the importance of each training example can be formulated as a bilevel problem. Let u be the vector of non-negative importance values $u = [u_1, \cdots, u_N]^T$ where N is the number of training examples, w be the parameter(s) of a classifier $h(x; w)$. Assuming access to a small set of validation data, from the same distribution as test data, $L_{\text{val}}(u, w) = \frac{1}{N_{\text{val}}} \sum_{(x_i,y_i)\in\mathcal{D}_{\text{val}}} l(h(x_i; u, w), y_i)$ be the loss on validation set and $L_{\text{w\_train}}(u, w) := \frac{1}{\sum_i u_i} \sum_{(x_i,y_i)\in\mathcal{D}_{\text{train}}} u_i l(h(x_i; w), y_i)$ be the weighted training error. The problem of learning the importance of each training example is as follows

$$\min_u \; L_{\text{val}}(u, w^*) \;\; \text{s.t.} \;\; w^* = \arg\min_w \; L_{\text{w\_train}}(u, w). \tag{3}$$

Meta-learning. This problem involves learning a prior on the hypothesis classes (a.k.a. inductive bias) for a given set of tasks. Few-shot learning is an example of meta-learning, where a learner is trained on several related tasks, during the meta-training phase, so that it generalizes well on unseen (but related) tasks during the meta-testing phase. An effective approach to this problem is to learn a common representation for various tasks and train task specific classifiers over this representation. Let T be the map that takes raw features to a common representation $T : \mathcal{X} \to \mathbb{R}^d$ for all tasks and h_i be the classifier for the i-th task, $i \in \{1, \cdots, M\}$ where M is the total number of tasks for training. The goal is to learn both the representation map $T(\cdot\,; u)$ parameterized by u and the set of classifiers $\{h_1, \cdots, h_M\}$ parameterized by $w = \{w_1, \cdots, w_M\}$. Let $L_{\text{val}}(u, w_i) := \frac{1}{N_{\text{val}}} \sum_{(x_i,y_i)\in\mathcal{D}_{\text{val}}} l(h_i(T(x_i; u); w_i), y_i)$ be the validation loss of task i and $L_{\text{train}}(u, w_i)$ be the training loss defined similarly. Then the bilevel problem for few-shot learning is as follows

$$\min_u \; \sum_i L_{\text{val}}(u, w_i^*) \;\; \text{s.t.} \;\; w_i^* = \arg\min_{w_i} \; L_{\text{train}}(u, w_i), \;\; i = 1, \cdots, M. \tag{4}$$

At test time the common representation $T(\cdot\,; u)$ is kept fixed and the classifiers $h_i^{'}$ for the new tasks are trained i.e. $\min_{w_i^{'}} \; L_{\text{test}}(u, w_i^{'}) \;\; i = 1, \cdots, N$ where N is the total number of tasks for testing.

Training-data poisoning. This problem refers to the setting in which an adversary can modify the training data so that the model trained on the altered data performs poorly/differently compared to one trained on the unaltered data. Attacker adds one or more 'poisoned' examples $u = \{u_1, \cdots, u_M\}$ to the original training data $X = \{x_1, \cdots, x_N\}$ i.e., $X' = X \bigcup u$ with arbitrary labels. Additionally, to evade detection, an attacker can generate poisoned images starting from existing clean images (called base images) with a bound on the maximum perturbation allowed. Let $L_{\text{poison}}(u, w) := \frac{1}{N} \sum_{(x_i, y_i) \in X' \times Y'} l(h(x_i; u, w), y_i)$ be the loss on the poisoned training data, ϵ be the bound on the maximum perturbation allowed for poisoned points and the validation set consists of target images that an attacker wants the model to misclassify. Then the problem of generating poisoning data is as follows

$$\min_{u} \; L_{\text{val}}(u, w^*) \text{ s.t. } \; w^* = \arg\min_{w} \; L_{\text{poison}}(u, w) \text{ and } \|x^i_{\text{base}} - u^i\|_2 < \epsilon \text{ for } i = 1, ..., M. \quad (5)$$

Challenges of deep bilevel optimization. General bilevel optimization problems cannot be solved using simultaneous optimization of the upper- and lower-level cost and are in fact, shown to be NP-hard even in cases with linear upper-level and quadratic lower-level functions [Bard (1991)]. To make the analyses tractable many previous works require assumptions such as convexity of the lower-level objective and lower-level solution set being a singleton[1]. Moreover, for solving bilevel problems involving deep neural networks, with millions of variables, only first-order methods such as gradient descent are feasible. However, the steepest descent direction (Hypergradient in Sec. 2.1) using the first-order methods for bilevel problems requires the computation of the inverse Hessian–gradient product. Since direct inversion of the Hessian is impractical even for moderate-sized problems, previous approaches approximate the hypergradient using either forward/reverse-mode differentiation [Maclaurin et al. (2015); Franceschi et al. (2017); Shaban et al. (2018)] or by approximately solving a linear system [Domke (2012); Pedregosa (2016); Rajeswaran et al. (2019); Lorraine et al. (2020)]. However, some of these approaches have high space and time complexities which can be problematic, especially for deep learning settings.

Contributions. We propose an algorithm (Alg. 1) based on the classical penalty function approach for solving large-scale bilevel optimization problems. We prove convergence of the method (Theorem 2) and present its complexity analysis showing that it has linear time and constant space complexity (Table 1). This makes our approach superior to forward/reverse-mode differentiation and similar to the approximate inversion-based methods. The small space and time complexities of the method make it an effective solver for large-scale bilevel problems involving deep neural networks as shown by our experiments on data denoising, few-shot learning, and training-data poisoning problems. In addition to being able to solve constrained problems, our method performs competitively to the state-of-the-art methods on simpler problems (with convex lower-level cost) and significantly outperforms other methods on complex problems (with non-convex lower-level cost), in terms of accuracy (Sec. 3), convergence speed (Sec. 3.5) and run-time (Appendix D.3).

The rest of the paper is organized as follows. We present and analyze the main algorithm in Sec. 2, present experiments in Sec. 3, and conclude in Sec. 4. Proofs, experimental settings, and additional results are presented in the supplementary material. The scripts used to generate the results in this paper are available at https://github.com/jihunhamm/bilevel-penalty.

1. Recently, Liu et al. (2021) presented an analysis of bilevel problems without requiring these assumptions.

2. Inversion-Free Penalty Method

We assume the upper- and lower-level costs f and g are twice continuously differentiable and the upper-level constraint function h is continuously differentiable in both u and v. Let $\nabla_u f$ and $\nabla_v f$ denote the gradient vectors, $\nabla^2_{uv} f$ denote the Jacobian matrix $\left[\frac{\partial^2 f}{\partial u_i \partial v_j}\right]$, and $\nabla^2_{vv} f$ denote the Hessian matrix $\left[\frac{\partial^2 f}{\partial v_i \partial v_j}\right]$. Following previous works, we also assume the lower-level solution $v^*(u) := \arg\min_v \; g(u,v)$ is unique for all u and $\nabla^2_{vv} g$ is invertible everywhere.

2.1. Background

A bilevel problem is a constrained optimization problem with the lower-level optimality $v^*(u) = \arg\min_v g(u,v)$ being a constraint. Additionally, it can also have other constraints in the upper- and lower-level problems[2].

$$\min_u \;\; f(u, v^*), \;\; \text{s.t.} \;\; h(u, v^*) \leq 0 \;\; \text{and} \;\; v^*(u) = \arg\min_v g(u,v). \tag{6}$$

Inequality constraints $h(u,v) \leq 0$ can be converted into equality constraints, e.g., $h(u,v) + s^2 = 0$ by using a slack variables s, resulting in the following problem

$$\min_u \;\; f(u, v^*), \;\; \text{s.t.} \;\; h(u, v^*) = 0 \;\; \text{and} \;\; v^*(u) = \arg\min_v g(u,v). \tag{7}$$

The assumption about the uniqueness of the lower-level solution for each u allows us to convert the bilevel problem in Eq. (7) into the following single-level constrained problem:

$$\min_{u,v} \;\; f(u,v), \;\; \text{s.t.} \;\; h(u,v) = 0 \;\; \text{and} \;\; \nabla_v g = 0. \tag{8}$$

For general bilevel problems, Eq. (7) and Eq. (8) are not the same [Dempe and Dutta (2012)]. But, for simpler problems where the lower-level cost g is convex in v for each u, the lower-level solution is unique for each u and the upper-level problem does not have any additional constraints, the problems in Eq. (8) and Eq. (7) are equivalent.

Hypergradient for bilevel optimization. For such simpler problems, we can use the gradient-based approaches on the single-level problem in Eq. (8) to compute the total derivative $\frac{df}{du}(u, v^*(u))$, also known as the hypergradient. By the chain rule, we have $\frac{df}{du} = \nabla_u f + \frac{dv}{du} \cdot \nabla_v f$ at $(u, v^*(u))$. Even if $v^*(u)$ cannot be found explicitly, we can still compute $\frac{dv}{du}$ using the implicit function theorem. As $\nabla_v g = 0$ at $v = v^*(u)$ and $\nabla^2_{vv} g$ is invertible, we get $du \cdot \nabla^2_{uv} g + dv \cdot \nabla^2_{vv} g = 0$, and $\frac{dv}{du} = -\nabla^2_{uv} g (\nabla^2_{vv} g)^{-1}$. Thus the hypergradient is

$$\frac{df}{du} = \nabla_u f - \nabla^2_{uv} g (\nabla^2_{vv} g)^{-1} \nabla_v f \;\; \text{at} \;\; (u, v^*(u)). \tag{9}$$

Existing approaches [Domke (2012); Maclaurin et al. (2015); Pedregosa (2016); Franceschi et al. (2017); Shaban et al. (2018); Rajeswaran et al. (2019); Lorraine et al. (2020)] can be viewed as implicit methods of approximating the hypergradient, with distinct efficiency and iteration complexity trade-offs [Grazzi et al. (2020)].

2. Problems with lower-level constraints are less common in machine learning and are left for future work.

2.2. Penalty function approach

The penalty function method is a well-known approach for solving constrained optimization problems as in Eq. (8) [Bertsekas (1997)]. It has been previously applied to solve bilevel problems described in Eq. (6) under strict assumptions and only high-level descriptions of the algorithm were presented [Aiyoshi and Shimizu (1984); Ishizuka and Aiyoshi (1992)]. The corresponding penalty function that is minimized to solve Eq. (8) has the form $\tilde{f}(u, v; \gamma) := f(u, v) + \frac{\gamma_k}{2}(\Psi(\|h(u, v)\|) + \|\nabla_v g(u, v)\|^2)$ which is the sum of the original cost f and the penalty terms for constraint satisfaction and first-order stationarity of the lower-level problem. The function Ψ is an interior or exterior penalty function depending on whether $h(u, v)$ is an equality or inequality constraint. The following result for obtaining a solution to the inequality constrained problem in Eq. (6) by solving Eq. (10) is known.

$$(\hat{u}_k, \hat{v}_k) = \arg\min_{u,v} \tilde{f}(u, v; \gamma_k) = \arg\min_{u,v} f(u, v) + \frac{\gamma_k}{2}(\Psi(\|h(u, v)\|) + \|\nabla_v g(u, v)\|^2). \quad (10)$$

Theorem 1 (Simplified Theorem 8.3.1 [Bard (2013)])**.** *Assume f and g are convex in v for any fixed u, Let $\{\gamma_k\}$ be any positive ($\gamma_k > 0$) and divergent ($\gamma_k \to \infty$) sequence. If $\{(\hat{u}_k, \hat{v}_k)\}$ is the corresponding sequence of the* **optimal** *solutions to Eq. (10), then the sequence $\{(\hat{u}_k, \hat{v}_k)\}$ has limit points any one of which is a solution to Eq. (6).*

2.3. Our algorithm

Theorem 1 presents a strong result, however, it is not very practical, especially for bilevel problems involving deep neural networks due to the following reasons. Firstly, the minimizer $(\hat{u}_k, \hat{v}_k)$ for Eq. (10) cannot be computed exactly for each γ_k and it is not computationally possible to increase $\gamma_k \to \infty$. Secondly, the upper- and lower-level costs f and g may not be convex in v for any u. To overcome some of these limitations and guarantee convergence in deep learning settings, we allow ϵ_k-optimal (instead of exact) solution to Eq. (10) at each k. Our Theorem 2 below (for equality constrained problems Eq. (7)), shows that the solution found by allowing ϵ_k-optimal solution to Eq. (10) converges to a KKT point of Eq. (8), assuming that linear independence constraint qualification (LICQ) is satisfied at the optimum (i.e., linear independence of the gradients of the constraints, h and $\nabla_v g$). We handle inequality constraints using slack variables and use $\Psi(\|\cdot\|) = \|\cdot\|^2$. Using the penalty function approach, we propose an algorithm for solving large-scale bilevel problems in Alg. 1.

Theorem 2. *Suppose $\{\epsilon_k\}$ is a positive ($\epsilon_k > 0$) and convergent ($\epsilon_k \to 0$) sequence, $\{\gamma_k\}$ is a positive ($\gamma_k > 0$), non-decreasing ($\gamma_1 \leq \gamma_2 \leq \cdots$), and divergent ($\gamma_k \to \infty$) sequence. Let $\{(u_k, v_k)\}$ be the sequence of approximate solutions to Eq. (10) with tolerance $(\nabla_u \tilde{f}(u_k, v_k))^2 + (\nabla_v \tilde{f}(u_k, v_k))^2 \leq \epsilon_k^2$ for all $k = 0, 1, \cdots$ and LICQ is satisfied at the optimum. Then any limit point of $\{(u_k, v_k)\}$ satisfies the KKT conditions of the problem in Eq. (8).*

The proof for Theorem 2 is based on the standard proof for penalty function methods (See [Nocedal and Wright (2006)]) and is presented in the Appendix A. Alg. 1 describes our method where we minimize the penalty function in Eq. (10), alternatively over v and u. This greatly reduces the complexity of solving a bilevel problem since in each iteration we only approximately solve a single-level problem over the penalty function $\tilde{f}$ with guaranteed convergence to KKT point of Eq. (8). Moreover, for unconstrained problems ($h \equiv 0$),

Algorithm 1 Our algorithm for solving bilevel problems (Penalty).

Input: $K, T, \{\sigma_k\}, \{\rho_{k,t}\}, \gamma_0, \epsilon_0, c_\gamma(=1.1), c_\epsilon(=0.9)$
Output: (u_K, v_T)
Initialize u_0, v_0 randomly
Begin
  for $k = 0, \cdots, K$-1 **do**
    while $\|\nabla_u \tilde{f}\|^2 + \|\nabla_v \tilde{f}\|^2 > \epsilon_k^2$ **do**
      for $t = 0, \cdots, T$-1 **do**
        $v_{t+1} \leftarrow v_t - \rho_{k,t} \nabla_v \tilde{f}$ (from Eq. (10))
      end for
      $u_{k+1} \leftarrow u_k - \sigma_k \nabla_u \tilde{f}$ (from Eq. (10))
    end while
    $\gamma_{k+1} \leftarrow c_\gamma \gamma_k,\ \epsilon_{k+1} \leftarrow c_\epsilon \epsilon_k$
  end for

Lemma 3 below shows that the approximate gradient direction $\nabla_u \tilde{f}$, computed from Alg. 1 becomes the exact hypergradient Eq. (9) asymptotically.

Lemma 3. *Assume $h \equiv 0$. Given u, let $\hat{v}$ be $\hat{v} := \arg\min_v \tilde{f}(u, v; \gamma)$ from Eq. (10). Then, $\nabla_u \tilde{f}(u, \hat{v}; \gamma) = \frac{df}{du}(u, \hat{v})$ as in Eq. (9).*

Thus if we find the minimizer $\hat{v}$ of the penalty function for given u and γ, Alg. 1 computes the exact hypergradient for unconstrained problems (Eq. (9)) at $(u, \hat{v})$. A similar theoretical analysis of the penalty method for non-bilevel problems was recently presented in [Shi and Hong (2020)] using a different stopping condition and a weaker constraint quantification condition. Although we have a similar theoretical result, [Shi and Hong (2020)] shows results on small-scale experiments on non-bilevel problems whereas we demonstrate our Alg. 1 on large-scale bilevel problems appearing in machine learning involving deep neural networks.

Comparison of Penalty with other algorithms. Previous algorithms for solving bilevel optimization problems rely on computing an approximation to the hypergradient using forward/reverse-mode differentiation (FMD/RMD) or approximately solving a linear system (ApproxGrad) (See Appendix B for a summary). Unlike these methods, Penalty does not require an explicit computation of the hypergradient in each iteration and obtains the exact hypergradient asymptotically, leading to better run-times for the experiments (Sec. 3.5 and Appendix D.3). Secondly, Penalty provides an easy way to incorporate upper-level constraints for various problems, unlike other methods which have to rely on projection to satisfy the constraints. Although projection is a reasonable method for convex constraints but for non-convex constraints, where computing the projection is intractable, Penalty has a significant computational advantage. Thirdly, for problems with multiple lower-level solutions, Penalty converges to the optimistic case solution without modification (Appendix C.3) whereas convergence of other methods is unknown because the hypergradient is dependent on the choice of the particular minimizer of the lower-level cost. Lastly, for unconstrained problems, we show the trade-offs of the different methods for computing the hypergradient in Table 1 and see that as T (total number of v-updates per one hypergradient computation) increases, FMD and RMD become impractical due to $O(cUT)$ time complexity

Table 1: Complexity analysis of various bilevel methods (Appendix B) on unconstrained problems. U is the size of u, V is the size of v, T is the number of v-updates per one hypergradient computation. P, p and q are variables of size $U \times V$, $U \times 1$, $V \times 1$ used to compute the hypergradient. We assume gradient of f, g, Hessian-vector product $\nabla^2_{vv} g$ and Jacobian-vector product $\nabla^2_{uv} g$ can be computed in time $c = c(U, V)$. (We use gradient descent as the process for FMD and RMD.)

Method	v-update	Intermediate updates	Time	Space
FMD	$v \leftarrow v - \rho \nabla_v g$	$P \leftarrow P(I - \rho \nabla^2_{vv} g) - \rho \nabla^2_{uv} g$	$O(cUT)$	$O(UV)$
RMD	$v \leftarrow v - \rho \nabla_v g$	$p \leftarrow p - \rho \nabla^2_{uv} g \cdot q$ $q \leftarrow q - \rho \nabla^2_{vv} g \cdot q$	$\boldsymbol{O(cT)}$	$O(U + VT)$
ApproxGrad	$v \leftarrow v - \rho \nabla_v g$	$q \leftarrow q - \rho \nabla^2_{vv} g [\nabla^2_{vv} g \cdot q - \nabla_v f]$	$\boldsymbol{O(cT)}$	$\boldsymbol{O(U+V)}$
Penalty	$v \leftarrow v - \rho [\nabla_v f + \gamma \nabla^2_{vv} g \nabla_v g]$	Not required	$\boldsymbol{O(cT)}$	$\boldsymbol{O(U+V)}$

and $O(U + VT)$ space complexity, respectively, whereas ApproxGrad and Penalty, have the same linear time complexity and constant space complexity. Recently, Lorraine et al. (2020) proposed an implicit gradient-based method similar to ApproxGrad where the Hessian inverse term in the hypergradient is approximated using the terms of the Neumann series. Another recent method, BVFIM, Liu et al. (2021), solves the bilevel problem by using a regularized value function of the lower-level problem in the upper-level objective and then solves a sequence of unconstrained problems. The time and space complexity of both these methods are the same as those of Penalty. Since the complexity analysis does not show the quality of the hypergradient approximation of these methods, we extensively compare Penalty against existing bilevel solvers (Sec. 3) and show its superiority on synthetic and real problems. We present a detailed comparison between ApproxGrad and Penalty since they have the same complexities and show that Penalty has better performance, run-time, and convergence speed (Fig. 4 and Fig. 5 in Appendix D).

Improvements. Some of the assumptions made for analysis such as unique lower-level solution may not hold in practice. Here, we discuss some techniques to address these issues and improve Alg. 1. The first problem is non-convexity of the lower-level cost g, which leads to the problem that a local minimum of $\|\nabla_v g\|$ can be either minima, maxima, or a saddle point of g. To address this we modify the v-update in Eq. (10) by adding a 'regularization' term $\lambda_k g$ to the cost. The minimization over v becomes $\min_v(\tilde{f} + \lambda_k g)$. This affects the optimization in the beginning; but as $\lambda_k \to 0$ the final solution remains unaffected with or without regularization. The second problem is that the tolerance $\nabla_{(u,v)} \tilde{f}(u_k, v_k; \gamma_k) \leq \epsilon_k$ may not be satisfied in a limited time and the optimization may terminate before γ_k becomes large enough. The method of multipliers and augmented Lagrangian [Bertsekas (1976)] can help the penalty method to find a solution with a finite γ_k. Thus we add the term $\nabla_v g^T \nu$ to the penalty function in Eq. (10) to get $\min_{u,v}(\tilde{f} + \nabla_v g^T \nu)$ and use the method of multipliers to update ν. In summary, we use the following update rules. $u_{k+1} \leftarrow u_k - \rho \nabla_u (\tilde{f} + \nabla_v g^T \nu_k)$, $v_{k+1} \leftarrow v_k - \sigma \nabla_v (\tilde{f} + \nabla_v g^T \nu_k + \lambda_k g)$, $\nu_{k+1} \leftarrow \nu_k + \gamma_k \nabla_v g$. Practically, the improvement due to these changes is moderate and problem-dependent (see Appendix C for details).

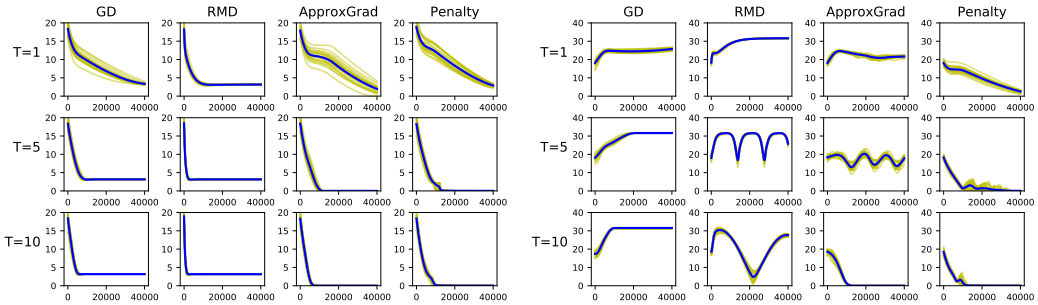
3. Experiments

In this section, we evaluate the performance of the proposed method (Penalty) on machine learning problems discussed earlier. Since previous bilevel methods in machine learning dealt only with unconstrained problems, we evaluate Penalty on unconstrained problems in Sec. 3.1-Sec. 3.3 and show its effectiveness in solving constrained problems in Sec. 3.4.

3.1. Synthetic problems

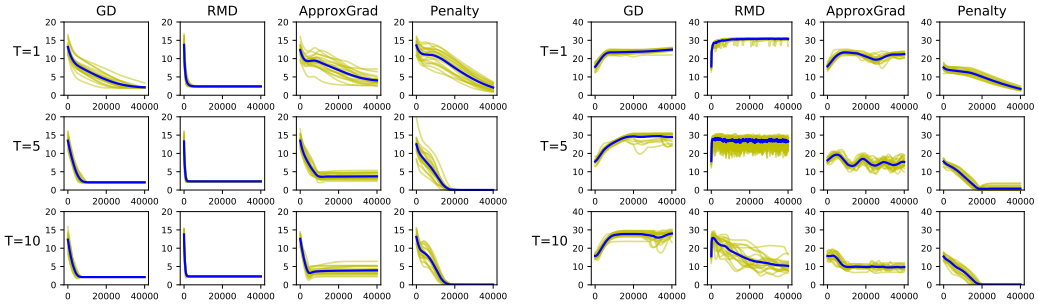
We compare Penalty against gradient descent (GD), reverse-mode differentiation (RMD), and approximate hypergradient method (ApproxGrad) on synthetic examples. We omit the comparison with forward-mode differentiation (FMD) because of its impractical time complexity for larger problems (Table 1). GD refers to the alternating minimization: $u \leftarrow u - \rho\nabla_u f$, $v \leftarrow v - \sigma\nabla_v g$. For RMD, we used the version with gradient descent as the lower-level process. For ApproxGrad experiments, we used Adam for all updates including solving the linear system. We found that Adam performs similar to the conjugate gradient method for solving the linear system with enough iterations. Since Adam uses the GPU effectively we used it for all experiments with ApproxGrad with a large number of iterations and added regularization to improve the ill-conditioning when solving the linear systems. Using simple quadratic surfaces for f and g, we compare all the algorithms by observing their convergence as a function of the number of upper-level iterations for a different number of lower-level updates (T). We measure the convergence using the Euclidean distance of the current iterate (u, v) from the closest optimal solution (u^*, v^*). Since the synthetic examples are not learning problems, we can only measure the distance of the iterates to an optimal solution ($\|(u, v) - (u^*, v^*)\|_2^2$). Fig. 1 shows the performance of two 10-dimensional examples described in the caption (see Appendix E.1). As one would expect, increasing the number T of v-updates makes all the algorithms, except GD, better since doing more lower-level iterations makes the hypergradient estimation more accurate but it increases the run time of the methods. However, even for these examples, only Penalty and ApproxGrad converge to the optimal solution and GD and RMD converge to non-solution points. A large T (eg. 100) makes RMD converge too but for large-scale experiments, it's impractical to have such a large T. From Fig. 1(b), we see that Penalty converges even with T=1 while ApproxGrad requires at least T=10 and RMD needs an even higher T to converge. This translates to smaller run-time for our method since run-time is directly proportional to T (Table 1).

In Fig. 2 we show examples with ill-conditioned or singular Hessian for the lower-level problem ($\nabla^2_{vv} g$). The ill-conditioning poses difficulty for the methods since the implicit function theorem requires the invertibility of the Hessian at the solution point. Fig. 2 shows that only Penalty converges to the true solution even though we added regularization ($\nabla^2_{vv} g + \lambda I$) while solving the linear system in ApproxGrad. Additionally, we report the wall clock times for different methods on the four examples tested here in Table 4 in the Appendix. We can see that as we increase the number of lower-level iterations all methods get slower but Penalty is still faster than both RMD and ApproxGrad. Although GD is the fastest but as seen in Fig. 1 and 2, GD does not converge to optima since not all bilevel solutions are saddle points. Moreover, for bilevel problems such as in Eq. (3) and Eq. (5) the upper-level costs are not directly dependent on the upper-level variable, thus the comparison with GD is omitted for large-scale bilevel problems.



(a) (b)

Figure 1: Convergence of GD, RMD, ApproxGrad, and Penalty vs upper-level epochs (x-axis) on synthetic problems: $f(u,v) = \|u\|^2 + \|v\|^2, g(u,v) = \|1\text{-}u\text{-}v\|^2$ (a) and $f(u,v) = \|v\|^2$ - $\|u-v\|^2, g(u,v) = \|u\text{-}v\|^2$ (b). Mean curve (blue) is superimposed on 20 independent trials (yellow).



(a) (b)

Figure 2: Convergence of GD, RMD, ApproxGrad, and Penalty vs upper-level epochs (x-axis) for synthetic problems: $f(u,v)=\|u\|^2 + \|v\|^2, g(u,v)=(1\text{-}u\text{-}v)^T A^T A(1\text{-}u\text{-}v)$ (a) and $f(u,v)=\|v\|^2\text{-}(u\text{-}v)^T A^T A(u\text{-}v), g(u,v)=(u\text{-}v)^T A^T A(u\text{-}v)$ (b), where $A^T A$ is a rank-deficient random matrix. Mean curve (blue) is superimposed on 20 independent trials (yellow).

3.2. Data denoising by importance learning

We evaluate the performance of Penalty on learning a classifier from a dataset with corrupted labels by posing the problem as an importance learning problem (Eq. (3)). The performance of the learned classifier using Penalty, with 20 lower-level updates (T), is evaluated against the following classifiers: **Oracle**: classifier trained on the portion of the training data with clean labels and the validation data, **Val-only**: classifier trained only on the validation data, **Train+Val**: classifier trained on the entire training and validation data, **ApproxGrad**: classifier trained with our implementation of ApproxGrad, with 20 lower-level and 20 linear system updates. We evaluate the performance on MNIST, CIFAR10, and SVHN datasets with the validation set sizes of 1000, 10000, and 1000 points respectively. We used convolutional neural networks

Table 2: Test accuracy (%) of the classifier learned after data denoising using importance learning. (Mean ± s.d. of 5 runs)

Dataset (Noise%)	Oracle	Val-Only	Train+Val	Bilevel Approaches	
				ApproxGrad	Penalty
MNIST (25)	99.3±0.1	94.6±0.3	83.9±1.3	98.11±0.08	**98.89**±0.04
MNIST (50)	99.3±0.1	94.6±0.3	60.8±2.5	97.27±0.15	**97.51**±0.07
CIFAR10 (25)	82.9±1.1	70.3±1.8	79.1±0.8	71.59±0.87	**79.67**±1.01
CIFAR10 (50)	80.7±1.2	70.3±1.8	72.2±1.8	68.08±0.83	**79.03**±1.19
SVHN (25)	91.1±0.5	70.6±1.5	71.6±1.4	80.05±1.37	**88.12**±0.16
SVHN (50)	89.8±0.6	70.6±1.5	47.9±1.3	74.18±1.05	**85.21**±0.34

Table 3: Few-shot classification accuracy (%) on Omniglot and Mini-ImageNet. We report mean±s.d. for Omniglot and 95% confidence intervals for Mini-Imagenet over five trials. Results for learning a common representation using Penalty, ApproxGrad and RMD [Franceschi et al. (2018)] are averaged over 600 randomly sampled tasks from the meta-test set. Results for previous methods using similar models are also reported (MAML [Finn et al. (2017)], iMAML [Rajeswaran et al. (2019)], Prototypical Networks (Proto Net) [Snell et al. (2017)], Relation Networks (Rel Net) [Sung et al. (2018)], SNAIL [Mishra et al. (2017)]).

						Learning a common representation		
	MAML	iMAML	Proto Net	Rel Net	SNAIL	RMD	ApproxGrad	Penalty
Omniglot								
5-way 1-shot	98.7	**99.50**±0.26	98.8	**99.6**±0.2	99.1	**98.6**	97.75±0.06	97.83±0.35
5-way 5-shot	**99.9**	99.74 ± 0.11	99.7	**99.8**±0.1	99.8	**99.5**	**99.51**±0.05	**99.45**±0.05
20-way 1-shot	95.8	96.18 ± 0.36	96.0	**97.6**±0.2	**97.6**	**95.5**	94.69±0.22	94.06±0.17
20-way 5-shot	98.9	99.14 ± 0.10	98.9	99.1±0.1	**99.4**	98.4	**98.46**±0.08	**98.47**±0.08
Mini-Imagenet								
5-way 1-shot	48.70±1.75	49.30 ± 1.88	49.42±0.78	50.44±0.82	**55.71**±0.99	50.54±0.85	43.74±1.75	**53.17**±0.96
5-way 5-shot	63.11±0.92	-	68.20±0.66	65.32±0.82	**68.88**±0.92	64.53±0.68	65.56±0.67	**67.74**±0.71

(architectures described in Appendix E.2) at the lower-level for this task. Table 2 summarizes our results for this problem and shows that Penalty outperforms Val-only, Train+Val, and ApproxGrad by significant margins and in fact performs very close to the Oracle classifier (which is the ideal classifier), even for high noise levels. This demonstrates that Penalty is extremely effective in solving bilevel problems involving several million variables (see Table 7 in Appendix) and shows its effectiveness at handling non-convex problems. Along with the improvement in terms of accuracy over ApproxGrad, Penalty also gives better run-time per upper-level iteration and higher convergence speed leading to a decrease in the overall wall-clock time of the experiments (Fig. 4(a) and Fig. 5(a) in Appendix D.3).

Next, evaluate Penalty against recent methods [Ren et al. (2018); Shu et al. (2019)] which use a meta-learning-based approach to assigns weights to the examples. We used the same setting as Ren et al.'s uniform flip experiment with 36% label noise on the CIFAR10 dataset and Wide ResNet 28-10 (WRN-28-10) model (see Appendix D.1). Using $T = 1$ for Penalty and 1000 validation points, we get an accuracy of 87.41 ± 0.26 (mean ± s.d. of 5 trials) comparable to 86.92 (Ren et al.) and 89.27 (Shu et al.). Thus, Penalty achieves comparable performance to these specialized methods designed for the data denoising problem. The enormous size of WRN-28-10, restricted us to use $T = 1$ but we expect larger T to improve the results (Fig. 5(a) in Appendix D.3). We also compared Penalty against an RMD-based method [Franceschi et al. (2017)], using the same setting as their Sec. 5.1, on a subset of MNIST data corrupted with 50% label noise and softmax regression as the model (see Appendix D.1). The accuracy of the classifier trained on a subset of the data with points having importance values greater than 0.9 (as computed by Penalty with $T = 20$) along with the validation set is 90.77% better than 90.09% reported by the RMD-based method.

3.3. Few-shot learning

Next, we evaluate the performance of Penalty on the task of learning a common representation for the few-shot learning problem. We use the formulation presented in Eq. (4) and use

Omniglot [Lake et al. (2015)] and Mini-ImageNet [Vinyals et al. (2016)] datasets for our experiments. Following the protocol proposed by [Vinyals et al. (2016)] for N-way K-shot classification, we generate meta-training and meta-testing datasets. Each meta-set is built using images from disjoint classes. For Omniglot, our meta-training set comprises of images from the first 1200 classes and the remaining 423 classes are used in the meta-testing dataset. We also augment the meta-datasets with three different rotations (90, 180, and 270 degrees) of the images as used by [Santoro et al. (2016)]. For the experiments with Mini-Imagenet, we used the split of 64 classes in meta-training, 16 classes in meta-validation, and 20 classes in meta-testing as used by [Ravi and Larochelle (2017)].

Each meta-batch of the meta-training and meta-testing dataset comprises of a number of tasks which is called the meta-batch-size. Each task in the meta-batch consists of a training set with K images and a testing set consists of 15 images from N classes. We train Penalty using a meta-batch-size of 30 for 5 way and 15 for 20-way classification for Omniglot and with a meta-batch-size of 2 for Mini-ImageNet experiments. The training sets of the meta-train-batch are used to train the lower-level problem and the test sets are used as validation sets for the upper-level problem in Eq. (4). The final accuracy is reported using the meta-test-set, for which we fix the common representation learned during meta-training. We then train the classifiers at the lower-level for 100 steps using the training sets from the meta-test-batch and evaluate the performance of each task on the associated test set from the meta-test-batch. The average performance of Penalty and ApproxGrad over 600 tasks is reported in Table 3. Penalty outperforms other bilevel methods namely the ApproxGrad (trained with 20 lower-level iterations and 20 updates for the linear system) and the RMD-based method [Franceschi et al. (2018)] on Mini-Imagenet and is comparable to them on Omniglot. We demonstrate the convergence speed of Penalty in comparison to ApproxGrad (Fig. 4(b)) and the trade-off between using higher T and time for the two methods (see Fig. 5 and Appendix D.3 for a detailed evaluation of the impact of T on the performance of Penalty) and show that Penalty converges much faster than ApproxGrad. In comparison to non-bilevel approaches that used models of a similar size as ours, Penalty is comparable to most approaches and is only slightly worse than [Mishra et al. (2017)] which makes use of temporal convolutions and soft attention. We used convolutional neural networks and a residual network for learning the common task representation (upper-level) for Omniglot and Mini-ImageNet, respectively, and use logistic regression to learn task-specific classifiers (lower-level).

3.4. Training-data poisoning

Here, we evaluate Penalty on the clean label data poisoning attack problem. We use the setting presented in [Shafahi et al. (2018)] which adds a single poison point to misclassify a particular target image from the test set. We use the dog vs. fish dataset and InceptionV3 network as the representation map. We choose a target image t and a base image b from the test set such that the representation of the base is closest to that of the target but has a different label. The poison point is initialized from the base image. Unlike the original approach [Shafahi et al. (2018)], we use a bilevel formulation along with a constraint on the maximum perturbation. We solve the bilevel problem in Eq. (5) with an additional feature collision term ($\|r(t) - r(u)\|_2^2$) from [Shafahi et al. (2018)] which is shown to be helpful in practice

where $r(\cdot)$ is the 2048-dimensional representation map. The lower-level problem trains a softmax classifier on top of this representation. (The full problem is in Eq. (12) of Appendix E.4). We evaluate the attack success by retraining the softmax classifier on the clean dataset augmented with the poison point. The attack is considered successful if the target point is misclassified after retraining. Choosing each correctly classified point from the test set as the target we search for the smallest $\epsilon \in \{1, 2, ..., 16\}$ that when used as the upper bound for the perturbation of the poison image, leads to misclassification of that target. For comparison, we use the Alg. 1 from [Shafahi et al. (2018)] using ℓ_2 projection after each step to constrain the amount of perturbation. Similar to Penalty, the smallest ϵ that causes misclassification is recorded for each target (see Appendix E.4 for details). Fig. 3 shows that Penalty achieves higher attack success with the same amount of distortion, or in another view, achieves the same attack success with much less distortion. We ascribe this benefit to the use of the bilevel method as opposed to a non-bilevel approach in [Shafahi et al. (2018)]. Poison points generated by Penalty are shown in Fig. 6 in Appendix. Lastly, we also test Penalty on a data poisoning problem without upper-level constraint (Appendix D.2). We compare it with RMD and ApproxGrad and show that Penalty outperforms RMD and is comparable to ApproxGrad.

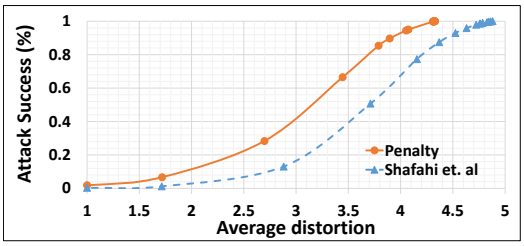
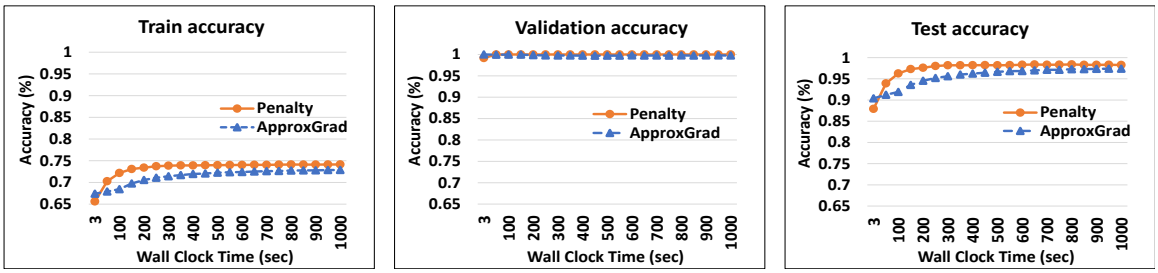


Figure 3: Penalty finds clean label poisoning data with smaller distortion compared to Shafahi et al. (2018) making the attack stealthier.

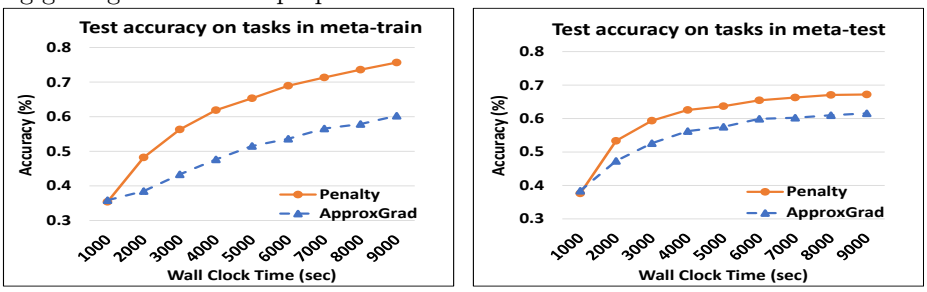
3.5. Convergence speed comparison of Penalty and ApproxGrad

Here, we compare the accuracy and wall clock time for Penalty and ApproxGrad on real (Fig. 4) and synthetic (Table 4 in Appendix) problems and show that Penalty converges faster than ApproxGrad. We use the data denoising and few-shot learning problems discussed earlier for this comparison. In Fig. 4, the training accuracy is indicative of the optimality of the lower-level problem and the validation/meta-train-test accuracy shows how well the upper-level problem is being solved during bilevel training. For data denoising, the test accuracy is a measure of how well the classifier learned using the importance values obtained from the bilevel training generalizes on the test set. On the other hand, for few-shot learning, the test accuracy on the tasks in the meta-test set is a measure of how well the common representation, learned from bilevel training, generalizes to new unseen tasks after training the softmax classifier on the training set of the meta-test dataset.

We see that for the importance learning problem Penalty quickly reaches a training accuracy of roughly 75% which is desirable since the dataset contains 25% noise. High test and validation accuracy are indicative that the importance values learned for the training data indeed helps to learn a good classifier. To report the performance for few-shot learning we train the softmax classifier on top of the common representation using the data from the meta-train-train and meta-test-train sets and then evaluate their performance on meta-train-test and meta-test-test sets. The results in Fig. 4(b) are an average of over 600 tasks. The performance on the meta-train-test is indicative of how well the upper-level problem is being solved. Accuracy on meta-test-test indicates the performance of the learned representation on new tasks which have not been observed during training. In both the experiments we see



(a) Data denoising for MNIST with 25% label noise and 1000 validation points. Figures show the performance of the classifier learned using the importance re-weighted dataset during bilevel training. 75% train accuracy and 100% validation accuracy is an indication that the bilevel problem is correctly solved. We see that Penalty reaches this point earlier than ApproxGrad signifying better convergence speed. High test accuracy indicates the importance values are enabling good generalization properties on the test set.



(b) 5-way 5-shot learning on Miniimagenet. Figures show the average performance of the softmax classifier trained using the train sets of meta-train and meta-test sets, respectively, on top of the common representation which is learned during bilevel training. The accuracy on the test set of meta-train indicates how well the upper-level problem is being solved. High performance on test sets of meta-test shows that the common representation generalizes to new tasks not observed during training.

Figure 4: Comparison of accuracy and wall clock time (convergence speed) during bilevel training with Penalty and ApproxGrad on the data denoising problem (Sec. 3.2) and the few-shot learning problem (Sec. 3.3).

Penalty has a better convergence speed than ApproxGrad. This directly translates to shorter running times for Penalty, making it a significantly more practical algorithm in comparison to existing bilevel solvers used for machine learning problems.

4. Conclusion

A wide range of important machine learning problems can be expressed as bilevel optimization problems, and new applications are still being discovered. So far, the difficulty of solving bilevel optimization has limited its widespread use for solving large problems involving deep models. In this work, we presented an efficient algorithm based on penalty function which is simple and has practical advantages over existing methods. Compared to previous methods we demonstrated our method's ability to handle constraints, achieve competitive performance on problems with convex lower-level costs, and get significant improvements on problems with non-convex lower-level costs in terms of accuracy and convergence speed, highlighting its effectiveness in deep learning settings. In future works, we plan to tackle other challenges in bilevel optimization such as handling problems with non-unique lower-level solutions.

5. Acknowledgement

We thank the anonymous reviewers for their insightful comments and suggestions. This work was supported by the NSF EPSCoR-Louisiana Materials Design Alliance (LAMDA) program #OIA-1946231.

References

Eitaro Aiyoshi and Kiyotaka Shimizu. A solution method for the static constrained stackelberg problem via penalty method. *IEEE Transactions on Automatic Control*, 29(12):1111–1114, 1984.

Jonathan F Bard. Some properties of the bilevel programming problem. *Journal of optimization theory and applications*, 68(2):371–378, 1991.

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

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

Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

Stephan Dempe and Joydeep Dutta. Is bilevel programming a special case of a mathematical program with complementarity constraints? *Mathematical programming*, 131(1-2):37–48, 2012.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

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

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

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

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*, pages 1563–1572, 2018.

Riccardo Grazzi, Luca Franceschi, Massimiliano Pontil, and Saverio Salzo. On the iteration complexity of hypergradient computation. *International Conference on Machine Learning*, 2020.

Jihun Hamm and Yung-Kyun Noh. K-beam minimax: Efficient optimization for deep adversarial learning. *International Conference on Machine Learning (ICML)*, 2018.

Yo Ishizuka and Eitaro Aiyoshi. Double penalty method for bilevel optimization problems. *Annals of Operations Research*, 34(1):73–88, 1992.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894, 2017.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Risheng Liu, Xuan Liu, Xiaoming Yuan, Shangzhi Zeng, and Jin Zhang. A value-function-based interior-point method for non-convex bi-level optimization. *arXiv preprint arXiv:2106.07991*, 2021.

Tongliang Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. *IEEE Transactions on pattern analysis and machine intelligence*, 38(3):447–461, 2016.

Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*, pages 1540–1552. PMLR, 2020.

Jelena Luketina, Mathias Berglund, Klaus Greff, and Tapani Raiko. Scalable gradient-based tuning of continuous regularization hyperparameters. In *International Conference on Machine Learning*, pages 2952–2960, 2016.

Chunjie Luo, Jianfeng Zhan, Xiaohe Xue, Lei Wang, Rui Ren, and Qiang Yang. Cosine normalization: Using cosine similarity instead of dot product in neural networks. In *International Conference on Artificial Neural Networks*, pages 382–391. Springer, 2018.

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

Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *AAAI*, pages 2871–2877, 2015.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 27–38. ACM, 2017.

Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

Barak A Pearlmutter. Fast exact multiplication by the hessian. *Neural computation*, 6(1): 147–160, 1994.

Fabian Pedregosa. Hyperparameter optimization with approximate gradient. In *International conference on machine learning*, pages 737–746, 2016.

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

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. *International Conference on Learning Representations (ICLR)*, 2017.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *ICML*, 2018.

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.

Amirreza Shaban, Ching-An Cheng, Nathan Hatch, and Byron Boots. Truncated back-propagation for bilevel optimization. *arXiv preprint arXiv:1810.10667*, 2018.

Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems*, pages 6103–6113, 2018.

Q. Shi and M. Hong. Penalty dual decomposition method for nonsmooth nonconvex optimization—part i: Algorithms and convergence analysis. *IEEE Transactions on Signal Processing*, 68:4108–4122, 2020.

Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. In *NeurIPS*, 2019.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4080–4090, 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*, pages 1199–1208, 2018.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.

Heinrich von Stackelberg. *Market structure and equilibrium*. Springer Science & Business Media, 2010.

Xiyu Yu, Tongliang Liu, Mingming Gong, Kun Zhang, and Dacheng Tao. Transfer learning with label noise. *arXiv preprint arXiv:1707.09724*, 2017.