Implicit Bilevel Optimization: Differentiating through Bilevel Optimization Programming

Francesco Alesiani

NEC Laboratories Europe, Heidelberg, Germany
Francesco.Alesiani@neclab.eu

Abstract

Bilevel Optimization Programming is used to model complex and conflicting interactions between agents, for example in Robust AI or Privacy-preserving AI. Integrating bilevel mathematical programming within deep learning is thus an essential objective for the Machine Learning community. Previously proposed approaches only consider single-level programming. In this paper, we extend existing single-level optimization programming approaches and thus propose *Differentiating through Bilevel Optimization Programming* (BIGRAD) for end-to-end learning of models that use Bilevel Programming as a layer. BIGRAD has wide applicability and can be used in modern machine learning frameworks. BIGRAD is applicable to both continuous and combinatorial Bilevel optimization problems. We describe a class of gradient estimators for the combinatorial case which reduces the requirements in terms of computation complexity; for the case of the continuous variable, the gradient computation takes advantage of the push-back approach (i.e. vector-jacobian product) for an efficient implementation. Experiments show that the BIGRAD successfully extends existing single-level approaches to Bilevel Programming.

Introduction

Neural networks provide unprecedented improvements in perception tasks, however, deep neural networks do not natively protect against adversarial attacks nor preserve the privacy of the training dataset. In recent years various approaches have been proposed to overcome this limitation (Shafique et al. 2020), for example by integrating adversarial training (Xiao et al. 2020). Some of these approaches require solving some optimization problems during training. Recent approaches propose thus differentiable layers that incorporate either quadratic (Amos and Kolter 2017), convex (Agrawal et al. 2019a), cone (Agrawal et al. 2019b), equilibrium (Bai, Kolter, and Koltun 2019), SAT (Wang et al. 2019) or combinatorial (Pogančić et al. 2019; Mandi and Guns 2020; Berthet et al. 2020) programs. The use of optimization programming as a layer of differentiable systems requires computing the gradients through these layers. With discrete variables, the gradient is zero almost everywhere, while with complex (black box) solvers, the gradient may not be accessible.

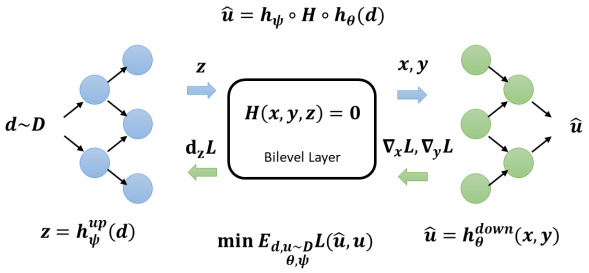


Figure 1: The Forward and backward passes of a Bilevel Programming (BIGRAD) layer: the larger system has input d and output $u = h_\psi \circ H \circ h_\theta(d)$; the bilevel layer has input z and output x, y, which are solutions of a Bilevel optimization problem represented by the implicit function $H(x, y, z) = 0$.

Proposed gradient estimates either relax the combinatorial problem (Mandi and Guns 2020), perturb the input variables (Berthet et al. 2020; Domke 2010) or linearly approximate the loss function (Pogančić et al. 2019). These approaches though, do now allow to directly express models with conflicting objectives, for example in structural learning (Elsken, Metzen, and Hutter 2019) or adversarial system (Goodfellow et al. 2014). We thus consider the use of bilevel optimization programming as a layer. Bilevel Optimization Program (Kleinert et al. 2021; Dempe 2018), also known as a generalization of Stackelberg Games, is the extension of a single-level optimization program, where the solution of one optimization problem (i.e. the outer problem) depends on the solution of another optimization problem (i.e. the inner problem). This class of problems can model interactions between two actors, where the action of the first depends on the knowledge of the counter-action of the second. Bilevel Programming finds application in various domains, as in Electricity networks, Economics, Environmental policy, Chemical plants, defense, and planning (Dempe 2018). We introduce at the end of the section example applications of Bilevel Optimization Programming.

In general, Bilevel programs are NP-hard (Dempe 2018), they require specialized solvers and it is not clear how to extend single-level approaches since the standard chain rule is not directly applicable. By modeling the bilevel optimiza-

tion problem as an implicit layer (Bai, Kolter, and Koltun 2019), we consider the more general case where 1) the solution of the bilevel problem is computed by a bilevel solver; thus leveraging on powerful solver developed over various decades (Kleinert et al. 2021); and 2) the computation of the gradient is more efficient since we do not have to propagate gradient through the solver.

We thus propose Differentiating through Bilevel Optimization Programming (BIGRAD):

- BIGRAD comprises of forwarding pass, where existing solvers (e.g. (Yang, Ji, and Liang 2021)) can be used, and backward pass, where BIGRAD estimates gradient for both continuous and combinatorial problems based on sensitivity analysis;
- we show how the proposed gradient estimators relate to the single-level analogous and that the proposed approach is beneficial in both continuous and combinatorial optimization learning tasks.

Adversarial Attack in Machine Learning Bilevel programming is used the represents the interaction between a machine learning model (y) and a potential attacker (x) (Goldblum, Fowl, and Goldstein 2019) and is used to increase the resilience to intentional or unintended adversarial attacks.

Min-Max Problems Min-max problems are used to model robust optimization problems (Ben-Tal, El Ghaoui, and Nemirovski 2009), where a second variable represents the environment and is constrained to an uncertain set that captures the unknown variability of the environment.

Closed-Loop Control of Physical Systems Bilevel Programming is able to model the interaction of a dynamical system (x) and its control sub-system (y), as, for example, of an industrial plant or a physical process. The control subsystem changes based on the state of the underlying dynamical system, which itself solves a physics constraint optimization problem (de Avila Belbute-Peres et al. 2018).

Interdiction Problems Two actors' discrete Interdiction problems (Fischetti et al. 2019) arise when one actor (x) tries to interdict the actions of another actor (y) under budget constraints. These problems can be found in marketing, protecting critical infrastructure, and preventing drug smuggling to hinder nuclear weapon proliferation.

Differentiable Bilevel Optimization Layer

We model the Bilevel Optimization Program as an Implicit Layer (Bai, Kolter, and Koltun 2019), i.e. as the solution of an implicit equation $H(x, y, z) = 0$. We thus compute the gradient using the implicit function theorem, where z is given and represents the parameters of our system we want to estimate, and x, y are output variables (Fig.1). We also assume we have access to a bilevel solver $(x, y) = \text{Solve}_H(z)$, e.g. (Yang, Ji, and Liang 2021). The bilevel Optimization Program is then used as layer of a differentiable system, whose input is d and output is given by $u = h_\psi \circ \text{Solve}_H \circ h_\theta(d) = h_{\psi,\theta}(d)$, where $\circ$ is the function composition operator. We want to learn the parameters ψ, θ of the function $h_{\psi,\theta}(d)$ that minimize the loss function $L(h_{\psi,\theta}(d), u)$, using the training data $D^{\text{tr}} = \{(d, u)_{i=1}^{N^{\text{tr}}}\}$. In order to be able to perform the end-to-end training, we need to back-propagate the gradient $\mathrm{d}_z L$ of the Bilevel Optimization Program Layer, which can not be accomplished only using the chain rule.

Continuous Bilevel Programming

We now present the definition of the continuous Bilevel Optimization problem, which comprises two non-linear functions f, g, as

$$\min_{x \in X} f(x, y, z) \qquad y \in \arg\min_{y \in Y} g(x, y, z) \tag{1}$$

where the left part problem is called *outer optimization problem* and resolves for the variable $x \in X$, with $X = \mathbb{R}^n$. The right problem is called the *inner optimization problem* and solves for the variable $y \in Y$, with $Y = \mathbb{R}^m$. The variable $z \in \mathbb{R}^p$ is the input variable and is a parameter for the bilevel problem. Min-max is a special case of Bilevel optimization problem $\min_{y \in Y} \max_{x \in X} g(x, y, z)$, where the minimization functions are equal and opposite in sign. In the Supplementary Material (Alesiani 2023), we describe how the model of Eq. 1 can be extended in the case of linear and nonlinear constraints.

Combinatorial Bilevel Programming

When the variables are discrete, we restrict the objective functions to be multi-linear (Greub 1967). Various important combinatorial problems are linear in discrete variables, e.g. vehicle routing Problem (VRP), traveling salesman problem (TSP), boolean satisfiability problem (SAT), which can be written as

$$\min_{x \in X} \langle z, x \rangle_A + \langle y, x \rangle_B, \;\; y \in \arg\min_{y \in Y} \langle w, y \rangle_C + \langle x, y \rangle_D \tag{2}$$

The variables x, y have domains in $x \in X, y \in Y$, where X, Y are convex polytopes that are constructed from a set of distinct points $\mathcal{X} \subset \mathbb{R}^n, \mathcal{Y} \subset \mathbb{R}^m$, as their convex hull. The outer and inner problems are Integer Linear Programs (ILPs). The multi-linear operator is represented by the inner product $\langle x, y \rangle_A = x^T A y$. We only consider the case where we have separate parameters for the outer and inner problems, $z \in \mathbb{R}^p$ and $w \in \mathbb{R}^q$.

BIGRAD: Gradient Estimation

BIGRAD provides gradient estimations for both continuous and discrete problems. We can identify the following common basic steps (Alg.1):

1. In the forward pass, solve the combinatorial or continuous Bilevel Optimisation problem as defined in Eq.1(or Eq.2) using existing solver ($\text{Solve}_H(z)$) e.g. (Yang, Ji, and Liang 2021);
2. During the backward pass, compute the gradient $\mathrm{d}_z L$ (and $\mathrm{d}_w L$) using the suggested gradients (see "Gradient Estimation" sections) starting from the gradients on the output variables $\nabla_x L$ and $\nabla_y L$.

Algorithm 1: BIGRAD Layer: Bilevel Optimization Programming Layer using BIGRAD

1. **Input**: Training sample $(\tilde{d}, \tilde{u})$
2. **Forward Pass**:
   (a) Compute $(x, y) \in \{x, y : H(x, y, z) = 0\}$ using Bilevel Solver: $(x, y) \in \text{Solve}_H(z)$
   (b) Compute the loss function $L(h_\psi \circ H \circ h_\theta(\tilde{d}), \tilde{u})$,
   (c) Save (x, y, z) for the backward pass
3. **Backward Pass**:
   (a) updates the parameter of the downstream layers ψ using back-propagation
   (b) For the continuous variable case, compute based on Theorem 2 around the current solution (x, y, z), without solving the Bilevel Problem
   (c) For the discrete variable case, use the gradient estimates of Theorem 3 (for the discrete case: Eq.8, or Eq.9) by solving, when needed, the two separate problems
   (d) Back-propagate the estimated gradient to the downstream parameters θ

Continuous Optimization Gradient Estimation

To evaluate the gradient of the variables z versus the loss function L, we need to propagate the gradients of the two output variables x, y through the two optimization problems. We can use the implicit function theorem to approximate locally the function $z \to (x, y)$. We have thus the following main results[1].

Theorem 1. *Considering the bilevel problem of Eq.1, we can build the following set of equations that represent the equivalent problem around a given solution x^*, y^*, z^*:*

$$F(x, y, z) = 0 \qquad G(x, y, z) = 0 \tag{3}$$

where

$$F(x, y, z) = \nabla_x f - \nabla_y f \nabla_y G \nabla_x G, \quad G(x, y, z) = \nabla_y g \tag{4}$$

where we used the short notation $f = f(x, y, z), g = g(x, y, z), F = F(x, y, z), G = G(x, y, z)$

Theorem 2. *Consider the problem defined in Eq.1, then the total gradient of the parameter z w.r.t. the loss function $L(x, y, z)$ is computed from the partial gradients $\nabla_x L, \nabla_y L, \nabla_z L$ as*

$$\mathrm{d}_z L = \nabla_z L - |\nabla_x L \quad \nabla_y L| \begin{vmatrix} \nabla_x F & \nabla_y F \\ \nabla_x G & \nabla_y G \end{vmatrix}^{-1} \begin{vmatrix} \nabla_z F \\ \nabla_z G \end{vmatrix} \tag{5}$$

The implicit layer is thus defined by the two conditions $F(x, y, z) = 0$ and $G(x, y, z) = 0$. We notice that Eq.5 can be solved without explicitly computing the Jacobian matrices and inverting the system, but by adopting the Vector-Jacobian product approach we can proceed from left to right to evaluate $\mathrm{d}_z L$. In the following section, we describe how affine equality constraints and nonlinear inequality can be used when modeling f, g. We also notice that the solution of Eq.5 does not require solving the original problem, but only applying matrix-vector products, i.e. linear algebra, and the evaluation of the gradient that can be computed using automatic differentiation. The extension of Theorem.2 to cone programming is presented the Supplementary Material (Alesiani 2023).

[1] Proofs are in the Supplementary Material (Alesiani 2023)

Combinatorial Optimization Gradient Estimation

When we consider discrete variables, the gradient is zero almost everywhere. We thus need to resort to estimating gradients. For the bilevel problem with discrete variables of Eq.2, when the solution of the bilevel problem exists and its solution is given (Kleinert et al. 2021), Thm.3 gives the gradients of the loss function with respect to the input parameters.

Theorem 3. *Given the Eq.2 problem, the partial variation of a cost function $L(x, y, z, w)$ on the input parameters has the following form:*

$$\mathrm{d}_z L = \nabla_z L + [\nabla_x L + \nabla_y L \nabla_x y] \nabla_z x \tag{6a}$$
$$\mathrm{d}_w L = \nabla_w L + [\nabla_x L \nabla_y x + \nabla_y L] \nabla_w y \tag{6b}$$

The $\nabla_x y, \nabla_y x$ terms capture the interaction between outer and inner problems. We could estimate the gradients in Thm.3 using the perturbation approach suggested in (Berthet et al. 2020), which estimates the gradient as the expected value of the gradient of the problem after perturbing the input variable, but, similar to REINFORCE (Williams 1992), this introduces large variance. While it is possible to reduce variance in some cases (Grathwohl et al. 2017) with the use of additional trainable functions, we consider alternative approaches as described in the following.

Differentiation of Black-Box Combinatorial Solvers (Pogančić et al. 2019) propose a way to propagate the gradient through a single-level combinatorial solver, where $\nabla_z L \approx \frac{1}{\tau}[x(z + \tau \nabla_x L) - x(z)]$ when $x(z) = \arg\max_{x \in X} \langle x, z \rangle$. We thus propose to compute the variation on the input variables from the two separate problems of the Bilevel Problem:

$$\nabla_z L \approx 1/\tau [x(z + \tau A \nabla_x L, y) - x(z, y)] \tag{7a}$$
$$\nabla_w L \approx 1/\tau [y(w + \tau C \nabla_y L, x) - y(w, x)] \tag{7b}$$

or alternatively, if we have only access to the Bilevel solver and not to the separate ILP solvers, we can express

$$\nabla_{z,w} L \approx 1/\tau [s(v + \tau E \nabla_{x,y} L) - s(v)] \tag{8}$$

where $x(z, y)$ and $y(w, x)$ represent the solutions of the two problems separately, $s(v) = (z, w) \to (x, y)$ the complete solution to the Bilevel Problem, $\tau \to 0$ is a hyper-parameter and $E = \begin{bmatrix} A & 0 \\ 0 & C \end{bmatrix}$. This form is more convenient than Eq.6 since it does not require computing the cross terms, ignoring thus the interaction of the two levels.

Straight-Through Gradient In estimating the input variables z, w of our model, we may not be interested in the interaction between the two variables x, y. Let us consider,

for example, the squared ℓ_2 loss function defined over the output variables

$$L^2(x, y) = L^2(x) + L^2(y)$$

where $L^2(x) = \frac{1}{2}\|x - x^*\|_2^2$ and x^* is the true value. The loss is non-zero only when the two vectors disagree, and with integer variables, it counts the difference squared, or, in the case of the binary variables, it counts the number of differences. If we compute $\nabla_x L^2(x) = (x - x^*)$ in the binary case, we have that $\nabla_{x_i} L^2(x) = +1$ if $x_i^* = 0 \wedge x_i = 1$, $\nabla_{x_i} L^2(x) = -1$ if $x_i^* = 1 \wedge x_i = 0$, and 0 otherwise. This information can be directly used to update the z_i variable in the linear term $\langle z, x \rangle$, thus we can estimate the gradients of the input variables as $\nabla_{z_i} L^2 = -\lambda \nabla_{x_i} L^2$ and $\nabla_{w_i} L^2 = -\lambda \nabla_{y_i} L^2$, with some weight $\lambda > 0$. The intuition is that the weight z_i associated with the variable x_i is increased when the value of the variable x_i reduces. In the general multilinear case, we have additional multiplicative terms. Following this intuition (see Sec.A.3), we thus use it as an estimate of the gradient of the variables

$$\nabla_z L = -A \nabla_x L \qquad \nabla_w L = -C \nabla_y L \tag{9}$$

This is equivalent in Eq.2 where $\nabla_z x = \nabla_w y = -I$ and $\nabla_y x = 0$, thus $\nabla_x y = 0$. This update is also equivalent to Eq.7, without the solution computation. The advantage of this form is that it does not require solving for an additional solution in the backward pass. For the single-level problem, the gradient has the same form as the Straight-Through gradient proposed by (Bengio, Léonard, and Courville 2013), with surrogate gradient $\nabla_z x = -I$.

Related Work

Bilevel Programming in Machine Learning Various papers model machine learning problems as Bilevel problems, for example in Hyper-parameter Optimization (MacKay et al. 2019; Franceschi et al. 2018), Meta-Feature Learning (Li and Malik 2016), Meta-Initialization Learning (Rajeswaran et al. 2019), Neural Architecture Search (Liu, Simonyan, and Yang 2018), Adversarial Learning (Li et al. 2019) and Multi-Task Learning (Alesiani et al. 2020). In these works, the main focus is to compute the solution to the bilevel optimization problems. In (MacKay et al. 2019; Lorraine and Duvenaud 2018), the best response function is modeled as a neural network and the solution is found using iterative minimization, without attempting to estimate the complete gradient. Many bilevel approaches rely on the use of the implicit function to compute the hyper-gradient (Sec. 3.5 of (Colson, Marcotte, and Savard 2007)) but do not use bilevel as a layer.

Quadratic, Cone and Convex single-level Programming Various works have addressed the problem of differentiate through quadratic, convex, or cone programming (Amos 2019; Amos and Kolter 2017; Agrawal et al. 2019b,a). In these approaches, the optimization layer is modeled as an implicit layer and for the cone/convex case, the normalized residual map is used to propagate the gradients. Contrary to our approach, this work only addresses single-level problems. These approaches do not consider combinatorial optimization.

Implicit Layer Networks While classical deep neural networks perform a single pass through the network at inference time, a new class of systems performs inference by solving an optimization problem. Examples of this are Deep Equilibrium Network (DEQ) (Bai, Kolter, and Koltun 2019) and NeurolODE (NODE) (Chen et al. 2018). Similar to our approach, the gradient is computed based on a sensitivity analysis of the current solution. These methods only consider continuous optimization.

Combinatorial Optimization (CO) Various papers estimate gradients of single-level combinatorial problems using relaxation. (Wilder, Dilkina, and Tambe 2019; Elmachtoub and Grigas 2017; Ferber et al. 2020; Mandi and Guns 2020) for example use ℓ_1, ℓ_2 or log barrier to relax the Integer Linear Programming (ILP) problem. Once relaxed the problem is solved using standard methods for continuous variable optimization. An alternative approach is suggested in other papers. For example, in (Pogančić et al. 2019) the loss function is approximated with a linear function and this leads to an estimate of the gradient of the input variable similar to the implicit differentiation by perturbation form (Domke 2010). (Berthet et al. 2020) is another approach that uses also perturbation and change of variables to estimate the gradient in an ILP problem. SatNet (Wang et al. 2019) solves MAXSAT problems by solving a continuous semidefinite program (SDP) relaxation of the original problem. These works only consider single-level problems.

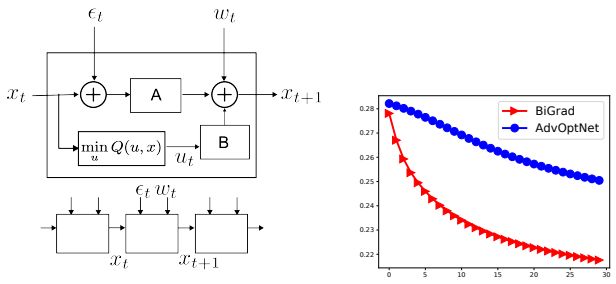
Discrete Latent Variables Discrete random variables provide an effective way to model multi-modal distributions over discrete values, which can be used in various machine learning problems. Gradients of discrete distribution are not mathematically defined, thus, in order to use the gradient-based method, gradient estimations have been proposed. A class of methods is based on the Gumbel-Softmax estimator (Maddison, Mnih, and Teh 2016). Gradient estimation of the exponential family of distributions over discrete variables is estimated using the perturb-and-MAP method in (Niepert, Minervini, and Franceschi 2021).

Predict Then Optimize Predict then Optimize (two-stage) (Elmachtoub and Grigas 2017; Ferber et al. 2020) or solving linear programs and submodular maximization from (Wilder, Dilkina, and Tambe 2019) solve optimization problems when the cost variable or the minimization function is directly observable. On the contrary, in our approach we only have access to a loss function on the output of the bilevel problem, thus allowing us to use it as a layer.

Neural Combinatorial Optimization (NCO) NCO employs deep neural networks to derive efficient CO heuristics. NCO includes supervised learning (Joshi, Laurent, and Bresson 2019) and reinforcement learning (Kool, Van Hoof, and Welling 2019).

Experiments

We evaluate BIGRAD with continuous and combinatorial problems to show that improves over single-level approaches. In the first experiment, we compare the use of



(a) (b)

Figure 2: (a) Visualization of the Optimal Control Learning network, where a disturbance ϵ_t is injected based on the control signal u_t. (b) Comparison of the training performance (cost vs iteration) for $N = 2$, $T = 20$ and epochs=10 of the BIGRAD and the Adversarial version of the OptNet (Amos and Kolter 2017).

BIGRAD versus the use of the implicit layer proposed in (Amos and Kolter 2017) for the design of Optimal Control with adversarial noise. In the second part, after experimenting with an adversarial attack, we explore the performance of BIGRAD with two combinatorial problems with Interdiction, where we adapted the experimental setup proposed in (Pogančić et al. 2019). In these latter experiments, we compare the formulation in Eq.8 (denoted by Bigrad(BB)) and the formulation of Eq.9 (denoted by Bigrad(PT)). In addition, we compare with the single level BB-1 from (Pogančić et al. 2019) and single level straight-through (Bengio, Léonard, and Courville 2013; Paulus, Maddison, and Krause 2021), with the surrogate gradient $\nabla_z x = -I$, (PT-1) gradient estimations. We compare against Supervised learning (SL), which ignores the underlying structure of the problem and directly predicts the solution of the bilevel problem.

Optimal Control with Adversarial Disturbance

We consider the design of robust stochastic control for a Dynamical System (Agrawal et al. 2019b). The problem is to find a feedback function $u = \phi(x)$ that minimizes

$$\min_{\phi} \mathbb{E}\, \frac{1}{T} \sum_{t=0}^{T} \|x_t\|^2 + \|\phi(x_t)\|^2 \tag{10a}$$

$$\text{s.t. } x_{t+1} = Ax_t + B\phi(x_t) + w_t, \forall t \tag{10b}$$

where $x_t \in \mathbb{R}^n$ is the state of the system, while w_t is a i.i.d. random disturbance and x_0 is given initial state. To solve this problem we use Approximate Dynamic Programming (ADP) (Wang and Boyd 2010) that solves a proxy quadratic problem

$$\min_{u_t} \; u_t^T P u_t + x_t Q u_t + q^T u_t \qquad \text{s.t. } \|u_t\|_2 \leq 1 \tag{11}$$

We can use the optimization layer as shown in Fig.2(a) and update the problem variables (e.g. P, Q, q) using gradient descent. We use the linear quadratic regulator (LQR) solution as the initial solution (Kalman 1964). The optimization

	LQR	OptNet	Bilevel
Adversarial (10 steps)	2.736	0.2722	**0.2379**
(30 steps)	-	0.2511	**0.2181**

Table 1: Optimal Control Average Cost; Bilevel approach improves (lower cost) over the two-step approach because is able to better capture the interaction between noise and control dynamics.

$L_\infty \leq \alpha$	DCNN	Bi-DCNN	CNN	CNN*
0	62.9 ± 0.3	**64.0** ± 0.4	63.4 ± 0.7	63.6 ± 0.5
5	42.6 ± 1.0	**44.5** ± 0.2	43.8 ± 1.2	44.3 ± 1.0
10	23.5 ± 1.5	**25.3** ± 0.8	24.3 ± 1.0	24.2 ± 1.0
15	14.4 ± 1.4	**15.6** ± 0.7	14.6 ± 0.7	14.3 ± 0.4
20	9.1 ± 1.2	**10.0** ± 0.6	9.2 ± 0.4	8.9 ± 0.2
25	6.1 ± 1.0	**6.8** ± 0.5	6.0 ± 0.2	5.9 ± 0.2
30	3.9 ± 0.7	**4.4** ± 0.5	3.9 ± 0.2	3.9 ± 0.1

Table 2: Performance on the adversarial attack with discrete features, with $Q = 10$. DCNN is the single-level discrete CNN, Bi-DCNN is the bilevel discrete CNN, CNN is the vanilla CNN, while CNN* is the CNN where we add the bilevel discrete layer after vanilla training.

module is replicated for each time step t, similarly to the Recursive Neural Network (RNN).

We can build a resilient version of the controller in the hypothesis that an adversarial is able to inject a noise of limited energy, but is arbitrarily dependent on the control u, by solving the following bilevel optimization problem

$$\max_{\epsilon} \; Q(u_t, x_t + \epsilon) \qquad \text{s.t. } ||\epsilon|| \leq \sigma \tag{12a}$$

$$u_t(\epsilon) = \arg\min_{u_t} Q(u_t, x_t) \qquad \text{s.t. } \|u_t\|_2 \leq 1 \tag{12b}$$

where $Q(u, x) = u^T P u + x_t Q u + q^t u$ and we want to learn the parameters $z = (P, Q, q)$, where $y = u_t, x = \epsilon$ of Eq.1.

We evaluate the performance to verify the viability of the proposed approach and compare with LQR and OptNet (Amos and Kolter 2017), where the outer problem is substituted with the best response function that computes the adversarial noise based on the computed output; in this case, the adversarial noise is a scaled version of Qu of Eq.11. Tab.1 and Fig.2(b) present the performance using BIGRAD, LQR, and the adversarial version of OptNet. BIGRAD improves over two-step OptNet (Tab.1), because is able to better model the interaction between noise and control dynamic.

Adversarial ML with Discrete Latent Variables

Machine learning models are heavily affected by the injection of intentional noise (Madry et al. 2017; Goodfellow, Shlens, and Szegedy 2014). An adversarial attack typically requires access to the machine learning model, in this way the attack model can be used during training to include its effect. Instead of training an end-to-end system as in

gradient type	accuracy [12x12 maps] train	validation	accuracy [18x18 maps] train	validation	accuracy [24x24 maps] train	validation
BIGRAD (BB)	95.8 ± 0.2	**94.5** ± 0.2	**97.1** ± 0.0	**96.4** ± 0.2	98.0 ± 0.0	**97.8** ± 0.0
BIGRAD (PT)	91.7 ± 0.1	91.6 ± 0.1	94.3 ± 0.0	94.2 ± 0.1	95.7 ± 0.0	95.6 ± 0.1
BB-1	95.9 ± 0.2	91.7 ± 0.1	96.7 ± 0.2	94.5 ± 0.1	97.1 ± 0.1	96.3 ± 0.2
PT-1	88.3 ± 0.2	87.5 ± 0.2	90.9 ± 0.4	90.6 ± 0.5	92.8 ± 0.1	92.8 ± 0.2
SL	**100.0** ± 0.0	26.2 ± 2.4	**99.9** ± 0.1	20.2 ± 0.5	**99.1** ± 0.2	14.0 ± 1.0

Table 3: Performance on the Dynamic Programming Problem with Interdiction. SL uses ResNet18.

(Goldblum, Fowl, and Goldstein 2019), where the attacker is aware of the model, we consider the case where the attacker can inject a noise at the feature level, as opposed to the input level (as in (Goldblum, Fowl, and Goldstein 2019)), this allows us to model the interaction as a bilevel problem. Thus, to demonstrate the use of a bilevel layer, we design a system that is composed of a feature extraction layer, followed by a discretization layer that operates on the space of $\{0,1\}^m$, where m is the hidden feature size, followed by a classification layer. The network used in the experiments is composed of two convolutional layers with max-pooling and two linear layers, all with relu activation functions, while the classification is a linear layer. We consider a more limited attacker that is not aware of the loss function of the model and does not have access to the full model, but rather only to the input of the discrete layer and is able to switch Q discrete variables, The interaction of the discrete layer with the attacker is described by the following bilevel problem:

$$\min_{x \in Q} \max_{y \in B} \langle z + x, y \rangle. \tag{13}$$

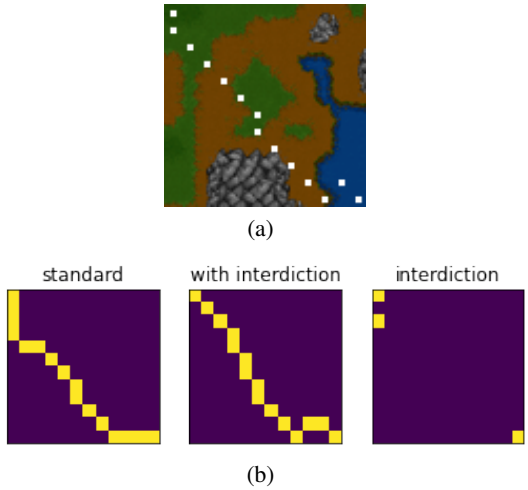
where Q represents the sets of all possible attacks, B is the budget of the discretization layer and y is the output of the layer. For the simulation, we compute the solution by sorting the features by values and considering only the first B values, while the attacker will obscure (i.e. set to zero) the first Q positions. The output y thus will have ones on the Q to B non-zero positions, and zero elsewhere. We train three models, on CIFAR-10 dataset for 50 epochs. For comparison we consider:1) the vanilla CNN network (i.e. without the discrete features); 2) the network with the single-level problem (i.e. the single-level problem without attacker) and; 3) the network with the bilevel problem (i.e. the min-max discretization problem defined in Eq.13). We then test the networks to adversarial attack using the PGD (Madry et al. 2017) attack similar to (Goldblum, Fowl, and Goldstein 2019). Similar results apply for FGSM attack (Fast Gradient Sign Attack) (Goodfellow, Shlens, and Szegedy 2014). We also tested the network trained as a vanilla network, where we added the min-max layer after training. From the results (Tab.2), we notice: 1) The min-max network shows improved resilience to adversarial attack wrt to the vanilla network, but also with respect to the max (single-level) network; 2) The min-max layer applied to the vanilla trained network is beneficial to adversarial attack; 3) The min-max network does not significantly change performance in presence of adversarial attack at the discrete layer (i.e. between Q=0 and Q=10). This example shows how bilevel layers can be successfully integrated into a Machine Learning system as differentiable layers.

Dynamic Programming: Shortest Path with Interdiction

We consider the problem of the Shortest Path with Interdiction, where the set of possible valid paths (see Fig.3(a)) is Y and the set of all possible interdiction is X. The mathematical problem can be written as

$$\min_{y \in Y} \max_{x \in X} \langle z + x \odot w, y \rangle \tag{14}$$

where $\odot$ is the element-wise product. This problem is multilinear in the discrete variables x, y, z. The z, w variables



(a)

(b)

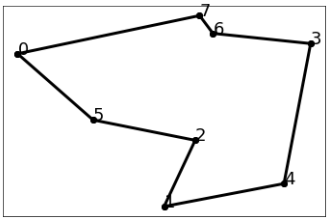
Figure 3: (a) Example Shortest Path in the Warcraft II tile set of (Guyomarch 2017). (b) Example Shortest Path without (left) and with interdiction (middle). Even a small interdiction (right) has a large effect on the output.

are the output of the neural network whose inputs are the Warcraft II tile images. The aim is to train the parameters of the weight network, such that we can solve the shortest path problem only based on the input image. For the experiments, we followed and adapted the scenario of (Pogančić et al. 2019) and used the Warcraft II tile maps of (Guyomarch 2017). We implemented the interdiction Game using a two-stage min-max-min algorithm (Kämmerling and Kurtz 2020). In Fig.3(b) it is possible to see the effect of interdiction on the final solution. Tab.3 shows the performances of the proposed approaches, where we allow for

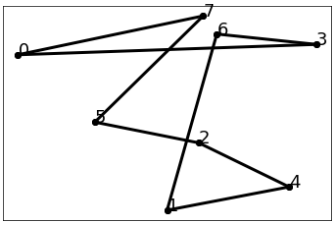
gradient type	k	accuracy train	accuracy validation	k	accuracy train	accuracy validation	k	accuracy train	accuracy validation
BB	8	89.2 ± 0.1	89.4 ± 0.2	10	91.9 ± 0.1	**92.0** ± 0.1	12	93.5 ± 0.1	93.5 ± 0.2
PT	8	89.3 ± 0.0	**89.4** ± 0.1	10	92.0 ± 0.0	91.9 ± 0.1	12	**93.7** ± 0.1	**93.7** ± 0.1
BB-1	8	84.0 ± 0.4	83.9 ± 0.4	10	87.4 ± 0.3	87.5 ± 0.4	12	89.3 ± 0.1	89.3 ± 0.1
PT-1	8	84.1 ± 0.4	84.1 ± 0.3	10	87.3 ± 0.3	87.0 ± 0.3	12	89.3 ± 0.0	89.5 ± 0.2
SL	8	**94.2** ± 5.0	10.7 ± 3.9	10	**92.7** ± 5.4	9.4 ± 0.4	12	91.4 ± 2.3	9.3 ± 1.2

Table 4: Performance in terms of the accuracy of the TSP use case with interdiction. SL has higher accuracy during train but fails at test time. BB and PT are BIGRAD variants.

$B = 3$ interdictions and we used tile size of 12×12, 18×18, 24×24. The loss function is the Hamming and ℓ_1 loss evaluated on both the shortest path y and the intervention x. The gradient estimated using Eq.8 (BB) provides more accurate results, at double of computation cost of PT. The single-level BB-1 approach outperforms PT, but shares similar computational complexity, while single-level PT-1 is inferior to PT. As expected, SL outperforms other methods during training, but completely fails during validation. Bigrad improves over single-level approaches because includes the interaction of the two problems.



(a)



(b)

Figure 4: Example of TSP with 8 cities and the comparison of a TSP tour without (a) or with (b) a single interdiction. Even a single interdiction has a large effect on the final tour.

Combinatorial Optimization: Travel Salesman Problem with Interdiction

Travel Salesman Problem (TSP) with interdiction consists of finding the shortest route $y \in Y$ that touches all cities, where some connections $x \in X$ can be removed. The mathematical problem to solve is given by

$$\min_{y \in Y} \max_{x \in X} \langle z + x \odot w, y \rangle \tag{15}$$

where z, w are cost matrices for the salesman and interceptor. Similar to the dynamic programming experiment, we implemented the interdiction Game using a two-stage min-max-min algorithm (Kämmerling and Kurtz 2020). Fig.4 shows the effect of a single interdiction. The aim is to learn the weight matrices, trained with the interdicted solutions on a subset of the cities. Tab.4 describes the performance in terms of accuracy on both the shortest tour and intervention. We use Hamming and ℓ_1 loss function. We only allow for $B = 1$ intervention but considered $k = 8, 10$, and 12 cities from a total of 100 cities. Single and two-level approaches perform similarly in the training and validation. Since the number of interdiction is limited to one, the performance of the single-level approach is not catastrophic, while the supervised learning approach completely fails in the validation set. Bigrad thus improves over single-level and SL approaches. Since Bigrad(PT) has a similar performance of BIGRAD (BB), thus PT is preferable in this scenario, since it requires fewer computation resources.

Conclusions

BIGRAD generalizes existing single-level gradient estimation approaches and can incorporate Bilevel Programming as a learnable layer in modern machine learning frameworks, which can model conflicting objectives as in adversarial attacks problems. The proposed novel gradient estimators are also efficient and the proposed framework is widely applicable to both continuous and discrete problems. The impact of BIGRAD has a marginal or similar cost with respect to the complexity of computing the solution of the Bilevel Programming problems. We show how BIGRAD is able to learn complex logic when the cost functions are multi-linear.

Limitations

Our approach models bilevel problems with discrete and continuous variables, but we have not explored the mixed integer programming problems, with mixed continuous and discrete variables. We rely on the use of existing solvers to compute the current solution and do not analyze the effect of using approximated solutions, thus we leave it to the next works to explore the potential to jointly solve and differentiate bilevel problems.

Ethical Statement

The present work does not have ethical implications, but shares with other machine learning approach the potential to be used in a large multitude of applications; we expect

our contribution to be used for the benefit and progress of our society.

References

Agrawal, A.; Amos, B.; Barratt, S.; Boyd, S.; Diamond, S.; and Kolter, Z. 2019a. Differentiable convex optimization layers. *arXiv:1910.12430*.

Agrawal, A.; Barratt, S.; Boyd, S.; Busseti, E.; and Moursi, W. M. 2019b. Differentiating through a cone program. *arXiv:1904.09043*.

Alesiani, F. 2023. Implicit Bilevel Optimization: Differentiating through Bilevel Optimization Programming. arXiv:2302.14473.

Alesiani, F.; Yu, S.; Shaker, A.; and Yin, W. 2020. Towards Interpretable Multi-Task Learning Using Bilevel Programming. *arXiv:2009.05483*.

Amos, B. 2019. *Differentiable optimization-based modeling for machine learning*. Ph.D. thesis, PhD thesis. Carnegie Mellon University.

Amos, B.; and Kolter, J. Z. 2017. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, 136–145. PMLR.

Bai, S.; Kolter, J. Z.; and Koltun, V. 2019. Deep equilibrium models. *arXiv:1909.01377*.

Ben-Tal, A.; El Ghaoui, L.; and Nemirovski, A. 2009. *Robust optimization*. Princeton university press.

Bengio, Y.; Léonard, N.; and Courville, A. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv:1308.3432*.

Berthet, Q.; Blondel, M.; Teboul, O.; Cuturi, M.; Vert, J.-P.; and Bach, F. 2020. Learning with differentiable perturbed optimizers. *arXiv:2002.08676*.

Chen, R. T.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. 2018. Neural ordinary differential equations. In Anderson, D., ed., *Neural Information Processing Systems*, 22–30. American Institute of Physics.

Colson, B.; Marcotte, P.; and Savard, G. 2007. An overview of bilevel optimization. *Annals of operations research*, 153(1): 235–256.

de Avila Belbute-Peres, F.; Smith, K.; Allen, K.; Tenenbaum, J.; and Kolter, J. Z. 2018. End-to-end differentiable physics for learning and control. *Advances in NeurIPS*.

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

Domke, J. 2010. Implicit differentiation by perturbation. *Advances in NeurIPS*, 23: 523–531.

Elmachtoub, A. N.; and Grigas, P. 2017. Smart" predict, then optimize". *arXiv:1710.08005*.

Elsken, T.; Metzen, J. H.; and Hutter, F. 2019. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1): 1997–2017.

Ferber, A.; Wilder, B.; Dilkina, B.; and Tambe, M. 2020. Mipaal: Mixed integer program as a layer. In *AAAI*, volume 34, 1504–1511.

Fischetti, M.; Ljubić, I.; Monaci, M.; and Sinnl, M. 2019. Interdiction games and monotonicity, with application to knapsack problems. *INFORMS Journal on Computing*, 31(2): 390–410.

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

Goldblum, M.; Fowl, L.; and Goldstein, T. 2019. Adversarially robust few-shot learning: A meta-learning approach. *arXiv:1910.00982*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. *Advances in NeurIPS*.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv:1412.6572*.

Grathwohl, W.; Choi, D.; Wu, Y.; Roeder, G.; and Duvenaud, D. 2017. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *arXiv:1711.00123*.

Greub, W. 1967. *Multilinear Algebra*. Springer Verlag.

Guyomarch, J. 2017. Warcraft ii open-source map editor. http://github.com/war2/war2edit. Accessed: 2021-08-09.

Joshi, C. K.; Laurent, T.; and Bresson, X. 2019. An efficient graph convolutional network technique for the travelling salesman problem. *arXiv:1906.01227*.

Kalman, R. E. 1964. When is a linear control system optimal? *Trans ASME, J. Basic Eng.*, 51–60.

Kämmerling, N.; and Kurtz, J. 2020. Oracle-based algorithms for binary two-stage robust optimization. *Computational Optimization and Applications*, 77(2): 539–569.

Kleinert, T.; Labbé, M.; Ljubić, I.; and Schmidt, M. 2021. A Survey on Mixed-Integer Programming Techniques in Bilevel Optimization. *EURO Journal on Computational Optimization*, 9: 100007.

Kool, W.; Van Hoof, H.; and Welling, M. 2019. Attention, learn to solve routing problems! *ICLR*.

Li, K.; and Malik, J. 2016. Learning to optimize. *arXiv:1606.01885*.

Li, Y.; Song, L.; Wu, X.; He, R.; and Tan, T. 2019. Learning a bi-level adversarial network with global and local perception for makeup-invariant face verification. *Pattern Recognition*, 90: 99–108.

Liu, H.; Simonyan, K.; and Yang, Y. 2018. Darts: Differentiable architecture search. *arXiv:1806.09055*.

Lorraine, J.; and Duvenaud, D. 2018. Stochastic hyperparameter optimization through hypernetworks. *arXiv:1802.09419*.

MacKay, M.; Vicol, P.; Lorraine, J.; Duvenaud, D.; and Grosse, R. 2019. Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions. *arXiv:1903.03088*.

Maddison, C. J.; Mnih, A.; and Teh, Y. W. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv:1611.00712*.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv:1706.06083*.

Mandi, J.; and Guns, T. 2020. Interior Point Solving for LP-based prediction+ optimisation. *arXiv:2010.13943*.

Niepert, M.; Minervini, P.; and Franceschi, L. 2021. Implicit MLE: backpropagating through discrete exponential family distributions. *Advances in Neural Information Processing Systems*, 34: 14567–14579.

Paulus, M. B.; Maddison, C. J.; and Krause, A. 2021. Rao-Blackwellizing the Straight-Through Gumbel-Softmax Gradient Estimator. In *arXiv:2010.04838*, 11.

Pogančić, M. V.; Paulus, A.; Musil, V.; Martius, G.; and Rolinek, M. 2019. Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations*.

Rajeswaran, A.; Finn, C.; Kakade, S.; and Levine, S. 2019. Meta-learning with implicit gradients. *arXiv:1909.04630*.

Shafique, M.; Naseer, M.; Theocharides, T.; Kyrkou, C.; Mutlu, O.; Orosa, L.; and Choi, J. 2020. Robust machine learning systems: Challenges, current trends, perspectives, and the road ahead. *IEEE Design & Test*, 37(2): 30–57.

Wang, P.-W.; Donti, P.; Wilder, B.; and Kolter, Z. 2019. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *ICML*, 6545–6554. PMLR.

Wang, Y.; and Boyd, S. 2010. Fast evaluation of quadratic control-Lyapunov policy. *IEEE Transactions on Control Systems Technology*, 19(4): 939–946.

Wilder, B.; Dilkina, B.; and Tambe, M. 2019. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4): 229–256.

Xiao, T.; Tsai, Y.-H.; Sohn, K.; Chandraker, M.; and Yang, M.-H. 2020. Adversarial learning of privacy-preserving and task-oriented representations. In *AAAI*.

Yang, J.; Ji, K.; and Liang, Y. 2021. Provably faster algorithms for bilevel optimization. *Advances in Neural Information Processing Systems*, 34: 13670–13682.