On the Iteration Complexity of Hypergradient Computation

Riccardo Grazzi [1 2] **Luca Franceschi** [1 2] **Massimiliano Pontil** [1 2] **Saverio Salzo** [1]

Abstract

We study a general class of bilevel problems, consisting in the minimization of an upper-level objective which depends on the solution to a parametric fixed-point equation. Important instances arising in machine learning include hyperparameter optimization, meta-learning, and certain graph and recurrent neural networks. Typically the gradient of the upper-level objective (hypergradient) is hard or even impossible to compute exactly, which has raised the interest in approximation methods. We investigate some popular approaches to compute the hypergradient, based on reverse mode iterative differentiation and approximate implicit differentiation. Under the hypothesis that the fixed point equation is defined by a contraction mapping, we present a unified analysis which allows for the first time to quantitatively compare these methods, providing explicit bounds for their iteration complexity. This analysis suggests a hierarchy in terms of computational efficiency among the above methods, with approximate implicit differentiation based on conjugate gradient performing best. We present an extensive experimental comparison among the methods which confirm the theoretical findings.

1. Introduction

Several problems arising in machine learning and related disciplines can be formulated as bilevel problems, where the lower-level problem is a fixed point equation whose solution is part of an upper-level objective. Instances of this framework include hyperparameter optimization (Maclaurin et al., 2015; Franceschi et al., 2017; Liu et al., 2018; Lorraine et al., 2019; Elsken et al., 2019), meta-learning (Andrychowicz et al., 2016; Finn et al., 2017; Franceschi et al., 2018), as well as recurrent and graph neural networks (Almeida, 1987; Pineda, 1987; Scarselli et al., 2008).

In large scale scenarios, there are thousands or even millions of parameters to find in the upper-level problem, making black-box approaches like grid and random search (Bergstra & Bengio, 2012) or Bayesian optimization (Snoek et al., 2012) impractical. This has made gradient-based methods (Domke, 2012; Maclaurin et al., 2015; Pedregosa, 2016) popular in such settings, but also it has raised the issue of designing efficient procedures to approximate the gradient of the upper-level objective (hypergradient) when finding a solution to the lower-level problem is costly.

The principal goal of this paper is to study the degree of approximation to the hypergradient of certain iterative schemes based on iterative or implicit differentiation[1]. In the rest of the introduction we present the bilevel framework, alongside some relevant examples in machine learning. We then outline the gradient approximation methods that we analyse in the paper and highlight our main contributions. Finally, we discuss and compare our results with previous work in the field.

The bilevel framework. In this work, we consider the following bilevel problem.

$$\begin{aligned} &\min_{\lambda \in \Lambda} f(\lambda) := E(w(\lambda), \lambda) \\ &\text{subject to } \; w(\lambda) = \Phi(w(\lambda), \lambda), \end{aligned} \tag{1}$$

where Λ is a closed convex subset of $\mathbb{R}^n$ and $E\colon \mathbb{R}^d \times \Lambda \to \mathbb{R}$ and $\Phi\colon \mathbb{R}^d \times \Lambda \to \mathbb{R}^d$ are continuously differentiable functions. We assume that the lower-level problem in (1) (which is a fixed point-equation) admits a unique solution. However, in general, explicitly computing such solution is either impossible or expensive. When f is differentiable, this issue affects the evaluation of the hypergradient $\nabla f(\lambda)$, which at best can only be approximately computed.

A prototypical example of the bilevel problem (1) is

$$\begin{aligned} &\min_{\lambda \in \Lambda} f(\lambda) := E(w(\lambda), \lambda)) \\ &\text{subject to } \; w(\lambda) = \arg\min_{u \in \mathbb{R}^d} \ell(u, \lambda), \end{aligned} \tag{2}$$

[1]Computational Statistics and Machine Learning, Istituto Italiano di Tecnologia, Genoa, Italy [2]Department of Computer Science, University College London, London, UK. Correspondence to: Riccardo Grazzi <riccardo.grazzi@iit.it>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

[1]The reader interested in the convergence analysis of gradient-based algorithms for bilevel optimization is referred to (Pedregosa, 2016; Rajeswaran et al., 2019) and references therein.

where $\ell\colon \mathbb{R}^d \times \Lambda \to \mathbb{R}$ is a loss function, twice continuously differentiable and strictly convex w.r.t. the first variable. Indeed if we let Φ be such that $\Phi(w, \lambda) = w - \alpha(\lambda)\nabla_1 \ell(w, \lambda)$, where $\alpha\colon \Lambda \to \mathbb{R}_{++}$ is differentiable, then problem (2) and problem (1) are equivalent. Specific examples of problem (2), which include hyperparameter optimization and meta-learning, are discussed in Section 3.1.

Other instances of the bilevel problem (1), which are not of the special form of problem (2), arise in the context of so-called equilibrium models (EQM). Notably, these comprise some types of connectionist models employed in domains with structured data. Stable recurrent neural networks (Miller & Hardt, 2019), graph neural networks (Scarselli et al., 2008) and the formulations by Bai et al. (2019) belong to this class. EQM differ from standard (deep) neural networks in that the internal representations are given by fixed points of learnable dynamics rather than compositions of a finite number of layers. The learning problem for such type of models can be written as

$$\begin{aligned} &\min_{\lambda=(\gamma,\theta)\in\Lambda} f(\lambda) := \sum_{i=1}^{n} E_i(w_i(\gamma), \theta), \\ &\text{subject to } w_i(\gamma) = \phi_i(w_i(\gamma), \gamma), \text{ for } 1 \le i \le n, \end{aligned} \tag{3}$$

where the operators $\phi_i : \mathbb{R}^d \times \Lambda \to \mathbb{R}^d$ (here $\Phi = (\phi_i)_{i=1}^n$) are associated to the training points x_i's, and the error functions E_i are the losses incurred by a standard supervised algorithm on the transformed dataset $\{w_i(\gamma), y_i\}_{i=1}^n$. A specific example is discussed in Section 3.2

In this paper, we present a unified analysis which allows for the first time to quantitatively compare popular methods to approximate $\nabla f(\lambda)$ in the general setting of problem (1). The strategies we consider can be divided in two categories:

1. *Iterative Differentiation* (ITD) (Maclaurin et al., 2015; Franceschi et al., 2017; 2018; Finn et al., 2017). One defines the sequence of functions $f_t(\lambda) = E(w_t(\lambda), \lambda)$, where $w_t(\lambda)$ are the fixed-point iterates generated by the map $\Phi(\cdot, \lambda)$. Then $\nabla f(\lambda)$ is approximated by $\nabla f_t(\lambda)$, which in turn is computed using forward (FMAD) or reverse (RMAD) mode automatic differentiation (Griewank & Walther, 2008).
2. *Approximate Implicit Differentiation* (AID) (Pedregosa, 2016; Rajeswaran et al., 2019; Lorraine et al., 2019). First, an (implicit) equation for $\nabla f(\lambda)$ is obtained through the implicit function theorem. Then, this equation is approximately solved by using a two stage scheme. We analyse two specific methods in this class: the *fixed-point method* (Lorraine et al., 2019), also referred to as recurrent backpropagation in the context of recurrent neural networks (Almeida, 1987; Pineda, 1987), and the *conjugate gradient method* (Pedregosa, 2016).

Both schemes can be efficiently implemented using automatic differentiation (Griewank & Walther, 2008; Baydin et al., 2018) achieving similar cost in time, while ITD has usually a larger memory cost than AID[2].

Contributions. Although there is a vast amount of literature on the two hypergradient approximation strategies previously described, it remains unclear whether it is better to use one or the other. In this work, we shed some light over this issue both theoretically and experimentally. Specifically our contributions are the following:

- We provide iteration complexity results for ITD and AID when the mapping defining the fixed point equation is a contraction. In particular, we prove non-asymptotic linear rates for the approximation errors of both approaches.
- We make a theoretical and numerical comparison among different ITD and AID strategies considering several experimental scenarios.

We note that, to the best of our knowledge, non-asymptoptic rates of convergence for AID were only recently given in the case of meta-learning (Rajeswaran et al., 2019). Furthermore, we are not aware of any previous results about non-asymptotic rates of convergence for ITD.

Related Work. Iterative differentiation for functions defined implicitly has been extensively studied in the automatic differentiation literature. In particular (Griewank & Walther, 2008, Chap. 15) derives asymptotic linear rates for ITD under the assumption that $\Phi(\cdot, \lambda)$ is a contraction. Another attempt to theoretically analyse ITD is made by Franceschi et al. (2018) in the context of the bilevel problem (2). There, the authors provide sufficient conditions for the asymptotic convergence of the set of minimizers of the approximate problem to the set of minimizers of the bilevel problem. In contrast, in this work, we give rates for the convergence of the approximate hypergradient $\nabla f_t(\lambda)$ to the true one (i.e. $\nabla f(\lambda)$). ITD is also considered in (Shaban et al., 2019) where $\nabla f_t(\lambda)$ is approximated via a procedure which is reminiscent of truncated backpropagation. The authors bound the norm of the difference between $\nabla f_t(\lambda)$ and its truncated version as a function of the truncation steps. This is different from our analysis which directly considers the problem of estimating the gradient of f.

In the case of AID, an asymptotic analysis is presented in (Pedregosa, 2016), where the author proves the convergence of an inexact gradient projection algorithm for the minimization of the function f defined in problem (2), using increasingly accurate estimates of $\nabla f(\lambda)$. Whereas

[2]This is true when ITD is implemented using RMAD, which is the standard approach when λ is high dimensional.

Rajeswaran et al. (2019) present complexity results in the setting of meta-learning with biased regularization. Here, we extend this line of work by providing complexity results for AID in the more general setting of problem (1).

We also mention the papers by Amos & Kolter (2017) and Amos (2019), which present techniques to differentiate through the solutions of quadratic and cone programs respectively. Using such techniques allows one to treat these optimization problems as layers of a neural network and to use backpropagation for the end-to-end training of the resulting learning model. In the former work, the gradient is obtained by implicitly differentiating through the KKT conditions of the lower-level problem, while the latter performs implicit differentiation on the residual map of Minty's parametrization.

A different approach to solve bilevel problems of the form (2) is presented by Mehra & Hamm (2019), who consider a sequence of "single level" objectives involving a quadratic regularization term penalizing violations of the lower-level first-order stationary conditions. The authors provide asymptotic convergence guarantees for the method, as the regularization parameter tends to infinity, and show that it outperforms both ITD and AID on different settings where the lower-level problem is non-convex.

All previously mentioned works except (Griewank & Walther, 2008) consider bilevel problems of the form (2). Another exception is (Liao et al., 2018), which proposes two improvements to recurrent backpropagation, one based on conjugate gradient on the normal equations, and another based on Neumann series approximation of the inverse.

2. Theoretical Analysis

In this section we establish non-asymptotic bounds on the hypergradient (i.e. $\nabla f(\lambda)$) approximation errors for both ITD and AID schemes (proofs can be found in Appendix A). In particular, in Section 2.1 we address the iteration complexity of ITD, while in Section 2.2, after giving a general bound for AID, we focus on two popular implementations of the AID scheme: one based on the conjugate gradient method and the other on the fixed-point method.

Notations. We denote by $\|\cdot\|$ applied to a vector (matrix) the Euclidean (spectral) norm. For a differentiable function $f\colon \mathbb{R}^n \to \mathbb{R}^m$ we denote by $f'(x) \in \mathbb{R}^{m\times n}$ the derivative of f at x. When $m = 1$, we denote by $\nabla f\colon \mathbb{R}^n \to \mathbb{R}^n$ the gradient of f. For a real-valued function $g\colon \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ we denote by $\nabla_1 g(x, y) \in \mathbb{R}^n$ and $\nabla_2 g(x, y) \in \mathbb{R}^m$ the partial derivatives w.r.t. the first and second variable respectively. We also denote by $\nabla_1^2 g(x, y) \in \mathbb{R}^{n\times n}$ and $\nabla_{12}^2 g(x, y) \in \mathbb{R}^{n\times m}$ the second derivative of g w.r.t. the first variable and the mixed second derivative of g w.r.t. the first and second variable. For a vector-valued function $h\colon \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^k$ we denote, by $\partial_1 h(x, y) \in \mathbb{R}^{k\times n}$ and $\partial_2 h(x, y) \in \mathbb{R}^{k\times m}$ the partial Jacobians w.r.t. the first and second variable respectively at $(x, y) \in \mathbb{R}^m \times \mathbb{R}^n$.

In the rest of the section, referring to problem (1), we will group the assumptions as follows. Assumption A is general while Assumption B and C are specific enrichments for ITD and AID respectively.

Assumption A. *For every $\lambda \in \Lambda$,*

(i) *$w(\lambda)$ is the unique fixed point of $\Phi(\cdot, \lambda)$.*

(ii) *$I - \partial_1\Phi(w(\lambda), \lambda)$ is invertible.*

(iii) *$\partial_1\Phi(\cdot, \lambda)$ and $\partial_2\Phi(\cdot, \lambda)$ are Lipschitz continuous with constants $\nu_{1,\lambda}$ and $\nu_{2,\lambda}$ respectively.*

(iv) *$\nabla_1 E(\cdot, \lambda)$ and $\nabla_2 E(\cdot, \lambda)$ are Lipschitz continuous with constants $\eta_{1,\lambda}$ and $\eta_{2,\lambda}$ respectively.*

A direct consequence of Assumption A(i)-(ii) and of the implicit function theorem is that $w(\cdot)$ and $f(\cdot)$ are differentiable on Λ. Specifically, for every $\lambda \in \Lambda$, it holds that

$$w'(\lambda) = (I - \partial_1\Phi(w(\lambda), \lambda))^{-1}\partial_2\Phi(w(\lambda), \lambda) \tag{4}$$

$$\nabla f(\lambda) = \nabla_2 E(w(\lambda), \lambda) + w'(\lambda)^\top \nabla_1 E(w(\lambda), \lambda). \tag{5}$$

See Theorem A.1 for details. In the special case of problem (2), equation (4) reduces (see Corollary A.1) to

$$w'(\lambda) = -\nabla_1^2 \ell(w(\lambda), \lambda)^{-1}\nabla_{21}^2 \ell(w(\lambda), \lambda).$$

Before starting with the study of the two methods ITD and AID, we give a lemma which introduces three additional constants that will occur in the complexity bounds.

Lemma 2.1. *Let $\lambda \in \Lambda$ and let $D_\lambda > 0$ be such that $\|w(\lambda)\| \le D_\lambda$. Then there exist $L_{E,\lambda}, L_{\Phi,\lambda} \in \mathbb{R}_+$ s.t.*

$$\sup_{\|w\|\le 2D_\lambda} \|\nabla_1 E(w, \lambda)\| \le L_{E,\lambda}, \quad \sup_{\|w\|\le 2D_\lambda} \|\partial_2\Phi(w, \lambda)\| \le L_{\Phi,\lambda}$$

The proof exploits the fact that the image of a continuous function applied to a compact set remains compact.

2.1. Iterative Differentiation

In this section we replace $w(\lambda)$ in (1) by the t-th iterate of $\Phi(\cdot, \lambda)$, for which we additionally require the following.

Assumption B. *For every $\lambda \in \Lambda$, $\Phi(\cdot, \lambda)$ is a contraction with constant $q_\lambda \in (0, 1)$.*

The approximation of the hypergradient $\nabla f(\lambda)$ is then obtained as in Algorithm 1. Assumption B looks quite restrictive, however it is satisfied in a number of interesting cases:

(a) In the setting of the bilevel optimization problem (2), suppose that for every $\lambda \in \Lambda$, $\ell(\cdot, \lambda)$ is $\mu_\ell(\lambda)$-strongly

Algorithm 1 Iterative Differentiation (ITD)

1. Let $t \in \mathbb{N}$, set $w_0(\lambda) = 0$, and compute,
$$\text{for } i = 1, 2, \dots t$$
$$\quad w_i(\lambda) = \Phi(w_{i-1}(\lambda), \lambda).$$
2. Set $f_t(\lambda) = E(w_t(\lambda), \lambda)$.
3. Compute $\nabla f_t(\lambda)$ using automatic differentiation.

convex and $L_\ell(\lambda)$-Lipschitz smooth for some continuously differentiable functions $\mu_\ell \colon \Lambda \to \mathbb{R}_{++}$, and $L_\ell \colon \Lambda \to \mathbb{R}_{++}$. Set $\kappa(\lambda) = L_\ell(\lambda)/\mu_\ell(\lambda)$,

$$\alpha(\lambda) = \frac{2}{\mu_\ell(\lambda) + L_\ell(\lambda)}, \text{ and } q_\lambda = \frac{\kappa(\lambda) - 1}{\kappa(\lambda) + 1}. \quad (6)$$

Then, $\Phi(w, \lambda) = w - \alpha(\lambda)\nabla_1 \ell(w, \lambda)$ is a contraction w.r.t. w with constant q_λ (see Appendix B).

(b) For strongly convex quadratic functions, accelerated methods like Nesterov's (Nesterov, 1983) or heavy-ball (Polyak, 1987) can be formulated as fixed-point iterations of a contraction in the norm defined by a suitable positive definite matrix.

(c) In certain graph and recurrent neural networks of the form (3), where the transition function is assumed to be a contraction (Scarselli et al., 2008; Almeida, 1987; Pineda, 1987).

The following lemma is a simple consequence of the theory on Neumann series and shows that Assumption B is stronger than Assumption A(i)-(ii). For reader's convenience the proof is given in Appendix A.

Lemma 2.2. *Let Assumption B be satisfied. Then, for every $\lambda \in \Lambda$, $\Phi(\cdot, \lambda)$ has a unique fixed point and, for every $w \in \mathbb{R}^d$, $I - \partial_1 \Phi(w, \lambda)$ is invertible and*

$$\|(I - \partial_1 \Phi(w, \lambda))^{-1}\| \leq \frac{1}{1 - q_\lambda}.$$

In particular, (i) and (ii) in Assumption A hold.

With Assumption B in force and if $w_t(\lambda)$ is defined as at point 1 in Algorithm 1, we have the following proposition that is essential for the final bound.

Proposition 2.1. *Suppose that Assumptions A(iii) and B hold and let $t \in \mathbb{N}$, with $t \geq 1$. Moreover, for every $\lambda \in \Lambda$, let $w_t(\lambda)$ be computed by Algorithm 1 and let D_λ and $L_{\Phi,\lambda}$ be as in Lemma 2.1. Then, $w_t(\cdot)$ is differentiable and, for every $\lambda \in \Lambda$,*

$$\|w_t'(\lambda) - w'(\lambda)\| \leq \left(\nu_{2,\lambda} + \nu_{1,\lambda} \frac{L_{\Phi,\lambda}}{1 - q_\lambda}\right) D_\lambda t q_\lambda^{t-1} + \frac{L_{\Phi,\lambda}}{1 - q_\lambda} q_\lambda^t. \quad (7)$$

Leveraging Proposition 2.1, we give the main result of this section.

Theorem 2.1. *(ITD bound) Suppose that Assumptions A(iii)-(iv) and B hold and let $t \in \mathbb{N}$ with $t \geq 1$. Moreover, for every $\lambda \in \Lambda$, let $w_t(\lambda)$ and f_t be defined according to Algorithm 1 and let D_λ, $L_{E,\lambda}$, and $L_{\Phi,\lambda}$ be as in Lemma 2.1. Then, f_t is differentiable and, for every $\lambda \in \Lambda$,*

$$\|\nabla f_t(\lambda) - \nabla f(\lambda)\| \leq \left(c_1(\lambda) + c_2(\lambda)\frac{t}{q_\lambda} + c_3(\lambda)\right) q_\lambda^t, \quad (8)$$

where

$$c_1(\lambda) = \left(\eta_{2,\lambda} + \frac{\eta_{1,\lambda} L_{\Phi,\lambda}}{1 - q_\lambda}\right) D_\lambda,$$
$$c_2(\lambda) = \left(\nu_{2,\lambda} + \frac{\nu_{1,\lambda} L_{\Phi,\lambda}}{1 - q_\lambda}\right) L_{E,\lambda} D_\lambda,$$
$$c_3(\lambda) = \frac{L_{E,\lambda} L_{\Phi,\lambda}}{1 - q_\lambda}.$$

In this generality this is a new result that provides a non-asymptotic linear rate of convergence for the gradient of f_t towards that of f.

2.2. Approximate Implicit Differentiation

In this section we study another approach to approximate the gradient of f. We derive from (4) and (5) that

$$\nabla f(\lambda) = \nabla_2 E(w(\lambda), \lambda) + \partial_2 \Phi(w(\lambda), \lambda)^\top v(\lambda) \quad (9)$$

where $v(\lambda)$ is the solution of the linear system

$$(I - \partial_1 \Phi(w(\lambda), \lambda)^\top) v = \nabla_1 E(w(\lambda), \lambda). \quad (10)$$

However, in the above formulas $w(\lambda)$ is usually not known explicitly or is expensive to compute exactly. To solve this issue $\nabla f(\lambda)$ is estimated as in Algorithm 2. Note that, unlike ITD, this procedure is agnostic about the algorithms used to compute the sequences $w_t(\lambda)$ and $v_{t,k}(\lambda)$. Interestingly, in the context of problem (2), choosing $\Phi(w, \lambda) = w - \nabla_1 \ell(w, \lambda)$ in Algorithm 2 yields the same procedure studied by Pedregosa (2016).

The number of iterations t and k in Algorithm 2 give a direct way of trading off accuracy and speed. To quantify this trade off we consider the following assumptions.

Assumption C. *For every $\lambda \in \Lambda$,*

(i) *$\forall\, w \in \mathbb{R}^d$, $I - \partial_1 \Phi(w, \lambda)$ is invertible.*

(ii) *$\|w_t(\lambda) - w(\lambda)\| \leq \rho_\lambda(t)\|w(\lambda)\|$, $\rho_\lambda(t) \leq 1$, and $\rho_\lambda(t) \to 0$ as $t \to +\infty$.*

(iii) *$\|v_{t,k}(\lambda) - v_t(\lambda)\| \leq \sigma_\lambda(k)\|v_t(\lambda)\|$ and $\sigma_\lambda(k) \to 0$ as $k \to +\infty$.*

Algorithm 2 Approximate Implicit Differentiation (AID)

1. Let $t \in \mathbb{N}$ and compute $w_t(\lambda)$ by t steps of an algorithm converging to $w(\lambda)$, starting from $w_0(\lambda) = 0$.
2. Compute $v_{t,k}(\lambda)$ after k steps of a solver for the system
$$(I - \partial_1 \Phi(w_t(\lambda), \lambda)^\top) v = \nabla_1 E(w_t(\lambda), \lambda). \quad (11)$$
3. Compute the approximate gradient as
$$\hat{\nabla} f(\lambda) := \nabla_2 E(w_t(\lambda), \lambda) + \partial_2 \Phi(w_t(\lambda), \lambda)^\top v_{t,k}(\lambda).$$

If Assumption C(i) holds, then, for every $\lambda \in \Lambda$, since the map $w \mapsto \|(I - \partial_1 \Phi(w, \lambda))^{-1}\|$ is continuous, we have

$$\sup_{\|w\| \le 2D_\lambda} \|(I - \partial_1 \Phi(w, \lambda))^{-1}\| \le \frac{1}{\mu_\lambda} < +\infty, \quad (12)$$

for some $\mu_\lambda > 0$. We note that, in view of Lemma 2.2, Assumption B implies Assumption C(i) (which in turn implies Assumption A(ii)) and in (12) one can take $\mu_\lambda = 1 - q_\lambda$. We stress that, Assumption C(ii)-(iii) are general and do not specify the type of algorithms solving the fixed-point equation $w = \Phi(w, \lambda)$ and the liner system (11). It is only required that such algorithms have explicit rates of convergence $\rho_\lambda(t)$ and $\sigma_\lambda(k)$ respectively. Finally, we note that Assumption C(ii) is less restrictive than Assumption B and encompasses the procedure at point 1 in Algorithm 1: indeed in such case C(ii) holds with $\rho_\lambda(t) = q_\lambda^t$.

It is also worth noting that the AID procedure requires only to store the last lower-level iterate, i.e. $w_t(\lambda)$. This is a considerable advantage over ITD, which instead requires to store the entire lower-level optimization trajectory $(w_i(\lambda))_{0 \le i \le t}$, if implemented using RMAD.

The iteration complexity bound for AID is given below. This is a general bound which depends on the rate of convergence $\rho_\lambda(t)$ of the sequence $(w_t(\lambda))_{t \in \mathbb{N}}$ and the rate of convergence $\sigma_\lambda(k)$ of the sequence $(v_{t,k}(\lambda))_{k \in \mathbb{N}}$.

Theorem 2.2. *(AID bound) Suppose that Assumptions A(i)(iii)(iv) and C(i)–(iii) hold. Let $\lambda \in \Lambda$, $t \in \mathbb{N}$, $k \in \mathbb{N}$. Let $D_\lambda, L_{E,\lambda}$, and $L_{\Phi,\lambda}$ be as in Lemma 2.1 and let μ_λ be defined according to (12). Let $\hat{\nabla} f(\lambda)$ be defined as in Algorithm 2 and let $\hat{\Delta} = \|\hat{\nabla} f(\lambda) - \nabla f(\lambda)\|$. Then,*

$$\hat{\Delta} \le \left(\eta_{2,\lambda} + \frac{\eta_{1,\lambda} L_{\Phi,\lambda}}{\mu_\lambda} + \frac{\nu_{2,\lambda} L_{E,\lambda}}{\mu_\lambda} + \frac{\nu_{1,\lambda} L_{\Phi,\lambda} L_{E,\lambda}}{\mu_\lambda^2} \right) \times D_\lambda \rho_\lambda(t) + \frac{L_{\Phi,\lambda} L_{E,\lambda}}{\mu_\lambda} \sigma_\lambda(k). \quad (13)$$

Furthermore, if Assumption B holds, then $\mu_\lambda = 1 - q_\lambda$ and

$$\hat{\Delta} \le \left(c_1(\lambda) + \frac{c_2(\lambda)}{1 - q_\lambda} \right) \rho_\lambda(t) + c_3(\lambda) \sigma_\lambda(k). \quad (14)$$

where $c_1(\lambda)$, $c_2(\lambda)$ and $c_3(\lambda)$ are defined in Theorem 2.1.

Theorem 2.2 provides a non-asymptotic rate of convergence for $\hat{\nabla} f$ which contrasts with the asymptotic result given in Pedregosa (2016). In this respect, making Assumption C(i) instead of the weaker Assumption A(ii) is critical.

Depending on the choice of the solver for the linear system (11) different AID methods are obtained. In the following we consider two cases.

AID with the Conjugate Gradient Method (AID-CG). For the sake of brevity we set $A_{\lambda,t} = I - \partial_1 \Phi(w_t(\lambda), \lambda)^\top$ and $b_{\lambda,t} = \nabla_1 E(w_t(\lambda), \lambda)$. Then, the linear system (11) is equivalent to the following minimization problem

$$\min_{v \in \mathbb{R}^d} \frac{1}{2} \|A_{\lambda,t} v - b_{\lambda,t}\|^2, \quad (15)$$

which, if $\partial_1 \Phi(w_t(\lambda), \lambda)$ is symmetric (so that $A_{\lambda,t}$ is also symmetric) is in turn equivalent to

$$\min_{v \in \mathbb{R}^d} \frac{1}{2} v^\top A_{\lambda,t} v - v^\top b_{\lambda,t}. \quad (16)$$

Several first order methods solving problems (15) or (16) satisfy assumption C(iii) with linear rates and require only Jacobian-vector products. In particular, for the symmetric case (16), the *conjugate gradient* method features the following linear rate

$$\|v_{t,k}(\lambda) - v_t(\lambda)\| \le 2\sqrt{\kappa(A_{\lambda,t})} \left(\frac{\sqrt{\kappa(A_{\lambda,t})} - 1}{\sqrt{\kappa(A_{\lambda,t})} + 1} \right)^k \|v_{t,0}(\lambda) - v_t(\lambda)\|, \quad (17)$$

where $\kappa(A_{\lambda,t})$ is the condition number of $A_{\lambda,t}$. In the setting of case (a) outlined in Section 2.1, $A_{\lambda,t} = \alpha(\lambda) \nabla_1^2 \ell(w_t(\lambda), \lambda)$ and

$$\mu_\ell(\lambda) I \preccurlyeq \nabla_1^2 \ell(w_t(\lambda), \lambda) \preccurlyeq L_\ell(\lambda) I.$$

Therefore the condition number of $A_{\lambda,t}$ satisfies $\kappa(A_{\lambda,t}) \le L_\ell(\lambda)/\mu_\ell(\lambda) = \kappa(\lambda)$ and hence

$$\frac{\sqrt{\kappa(A_{\lambda,t})} - 1}{\sqrt{\kappa(A_{\lambda,t})} + 1} \le \frac{\sqrt{\kappa(\lambda)} - 1}{\sqrt{\kappa(\lambda)} + 1} \le \frac{\kappa(\lambda) - 1}{\kappa(\lambda) + 1} = q_\lambda. \quad (18)$$

AID with the Fixed-Point Method (AID-FP). In this paragraph we make a specific choice for the sequence $(v_{t,k}(\lambda))_{k \in \mathbb{N}}$ in Assumption C(iii). We let Assumption B be satisfied and consider the following algorithm. For every $\lambda \in \Lambda$ and $t \in \mathbb{N}$, we choose $v_{t,0}(\lambda) = 0 \in \mathbb{R}^d$ and,

$$\text{for } k = 1, 2, \dots \quad v_{t,k}(\lambda) = \partial_1 \Phi(w_t(\lambda), \lambda)^\top v_{t,k-1}(\lambda) + \nabla_1 E(w_t(\lambda), \lambda). \quad (19)$$

In such case the rate of convergence $\sigma_\lambda(k)$ is linear. More precisely, since $\|\partial_1 \Phi(w_t(\lambda), \lambda)\| \leq q_\lambda < 1$ (from Assumption B), then the mapping

$$T \colon v \mapsto \partial_1 \Phi(w_t(\lambda), \lambda) v + \nabla_1 E(w_t(\lambda), \lambda)$$

is a contraction with constant q_λ. Moreover, the fixed-point of T is the solution of (11). Therefore, $\|v_{t,k}(\lambda) - v_t(\lambda)\| \leq q_\lambda^k \|v_{t,0}(\lambda) - v_t(\lambda)\|$. In the end the following result holds.

Theorem 2.3. *If Assumption B holds and $(v_{t,k}(\lambda))_{k\in\mathbb{N}}$ is defined according to (19), then Assumption C(iii) is satisfied with $\sigma_\lambda(k) = q_\lambda^k$.*

Now, plugging the rate $\sigma_\lambda(k) = q_\lambda^k$ into the general bound (14) yields

$$\hat{\Delta} \leq \Big(c_1(\lambda) + \frac{c_2(\lambda)}{1 - q_\lambda}\Big)\rho_\lambda(t) + c_3(\lambda) q_\lambda^k. \tag{20}$$

However, an analysis similar to the one in Section 2.1 shows that the above result can be slightly improved as follows.

Theorem 2.4. *(AID-FP bound) Suppose that Assumptions A(i)(iii)(iv) and Assumption B hold. Suppose also that (19) holds. Let $\hat{\nabla} f(\lambda)$ be defined according to Algorithm 2 and $\hat{\Delta} = \|\hat{\nabla} f(\lambda) - \nabla f(\lambda)\|$. Then, for every $t, k \in \mathbb{N}$,*

$$\hat{\Delta} \leq \Big(c_1(\lambda) + c_2(\lambda)\frac{1 - q_\lambda^k}{1 - q_\lambda}\Big)\rho_\lambda(t) + c_3(\lambda) q_\lambda^k, \tag{21}$$

where $c_1(\lambda)$, $c_2(\lambda)$ and $c_3(\lambda)$ are given in Theorem 2.1.

We end this section with a discussion about the consequences of the presented results.

2.3. Discussion

Theorem 2.2 shows that Algorithm 2 computes an approximate gradient of f with a linear convergence rate (in t and k), provided that the solvers for the lower-level problem and the linear system converge linearly. Furthermore, under Assumption B, both AID-FP and ITD converge linearly. However, if in Algorithm 2 we define $w_t(\lambda)$ as at point 1 in Algorithm 1 (so that $\rho_\lambda(t) = q_\lambda^t$), and take $k = t$, then the bound for AID-FP (21) is lower than that of ITD (8), since $q_\lambda(1 - q_\lambda^t)/(1 - q_\lambda) = \sum_{i=1}^t q_\lambda^i < t$ for every $t \geq 1$. This analysis suggests that AID-FP converge faster than ITD.

We now discuss the choice of the algorithm to solve the linear system (11) in Algorithm 2. Theorem 2.4 provides a bound for AID-FP, which considers procedure (19). However, we see from (14) in Theorem 2.2 that a solver for the linear system with rate of convergence $\sigma_\lambda(k)$ faster than q_λ^k may give a better bound. The above discussion, together with (17) and (18), proves that AID-CG has a better asymptotic rate than AID-FP for instances of problem (2) where the lower-level objective $\ell(\cdot, \lambda)$ is Lipschitz smooth and strongly convex (case (a) outlined in Section 2.1).

Finally, we note that both ITD and AID consider the initialization $w_0(\lambda) = 0$. However, in a gradient-based bilevel optimization algorithm, it might be more convenient to use a warm start strategy where $w_0(\lambda)$ is set based on previous upper-level iterations. Our analysis can be applied also in this case, but the related upper bounds will depend on the upper-level dynamics. This aspect makes it difficult to theoretically analyse the benefit of a warm start strategy, which remains an open question.

3. Experiments

In the first part of this section we focus on the hypergradient approximation error and show that the upper bounds presented in the previous section give a good estimate of the actual convergence behaviour of ITD and AID strategies on a variety of settings. In the second part we present a series of experiments pertaining optimization on both the settings of hyperparameter optimization, as in problem (2), and learning equilibrium models, as in problem (3). The algorithms have been implemented[3] in PyTorch (Paszke et al., 2019). In the following, we shorthand AID-FP and AID-CG with FP and CG, respectively.

3.1. Hypergradient Approximation

In this section, we consider several problems of type (2) with synthetic generated data (see Appendix C.1 for more details) where $D = (X, y)$ and $D' = (X', y')$ are the training and validation sets respectively, with $X \in \mathbb{R}^{n_e \times p}$, $X' \in \mathbb{R}^{n'_e \times p}$, being n_e, n'_e the number of examples in each set and p the number of features. Specifically we consider the following settings, which are representative instances of relevant bilevel problems in machine learning.

Logistic Regression with ℓ_2 Regularization (LR). This setting is similar to the one in Pedregosa (2016), but we introduce multiple regularization parameters:

$$f(\lambda) = \sum_{(x_e, y_e) \in D'} \psi(y_e x_e^\top w(\lambda)),$$

$$w(\lambda) = \operatorname*{argmin}_{w \in \mathbb{R}^p} \sum_{(x_e, y_e) \in D} \psi(y_e x_e^\top w) + \frac{1}{2} w^\top \operatorname{diag}(\lambda) w,$$

where $\lambda \in \mathbb{R}^p_{++}$, $\psi(x) = \log(1 + e^{-x})$ and $\operatorname{diag}(\lambda)$ is the diagonal matrix formed by the elements of λ.

Kernel Ridge Regression (KRR). We extend the setting presented by Pedregosa (2016) considering a p-dimensional Gaussian kernel parameter γ in place of the usual one:

[3] The code is freely available at the following link. https://github.com/prolearner/hypertorch

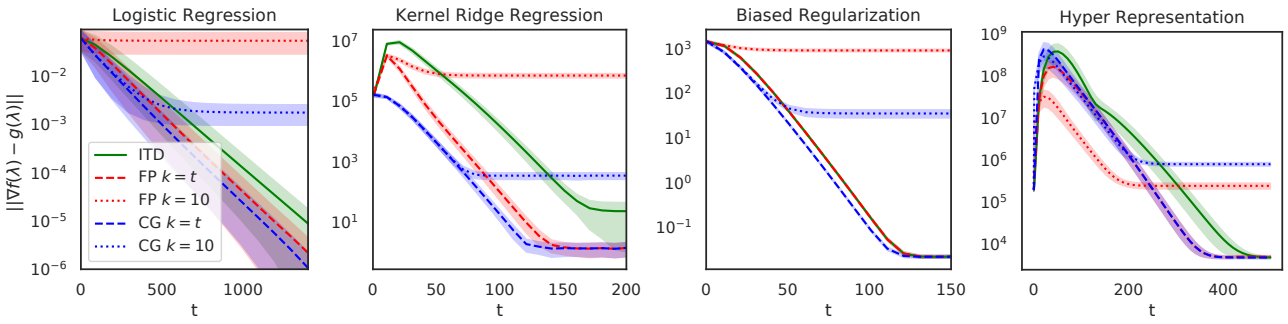

Figure 1. Convergence of different hypergradient approximations, where $g(\lambda)$ is equal to $\nabla f_t(\lambda)$ for ITD and to $\hat{\nabla} f(\lambda)$ for CG and FP. Mean and standard deviation (shaded areas) are computed over 20 values of λ sampled uniformly from $[\lambda_{\min}, \lambda_{\max}]^n$.

$$f(\beta, \gamma) = \frac{1}{2}\|y' - K'(\gamma)w(\beta, \gamma)\|^2,$$
$$w(\beta, \gamma) = \operatorname*{argmin}_{w \in \mathbb{R}^{n_e}} \frac{1}{2} w^\top \left(K(\gamma) + \beta I\right) w - w^\top y, \tag{22}$$

where $\beta \in (0, \infty)$, $\gamma \in \mathbb{R}^p_{++}$ and $K'(\gamma)$, $K(\gamma)$ are respectively the validation and training kernel matrices (see Appendix C.1).

Biased Regularization (BR). Inspired by Denevi et al. (2019); Rajeswaran et al. (2019), we consider the following.

$$f(\lambda) = \frac{1}{2}\|X'w(\lambda) - y'\|^2,$$
$$w(\lambda) = \operatorname*{argmin}_{w \in \mathbb{R}^p} \frac{1}{2}\|Xw - y\|^2 + \frac{\beta}{2}\|w - \lambda\|^2,$$

where $\beta \in \mathbb{R}_{++}$ and $\lambda \in \mathbb{R}^p$.

Hyper-representation (HR). The last setting, reminiscent of (Franceschi et al., 2018; Bertinetto et al., 2019), concerns learning a (common) linear transformation of the data and is formulated as

$$f(H) = \frac{1}{2}\|X'Hw(H) - y'\|^2$$
$$w(H) = \operatorname*{argmin}_{w \in \mathbb{R}^d} \frac{1}{2}\|XHw - y\|^2 + \frac{\beta}{2}\|w\|^2$$

where $H \in \mathbb{R}^{p \times d}$ and $\beta \in \mathbb{R}_{++}$.

LR and KRR are high dimensional extensions of classical hyperperparameter optimization problems, while BR and HR, are typically encountered in multi-task/meta-learning as single task objectives[4]. Note that Assumption B (i.e. $\Phi(\cdot, \lambda)$ is a contraction) can be satisfied for each of the aforesaid scenarios, since they all belong to case (a) of Section 2.1 (KRR, BR and HR also to case (b)).

We solve the lower-level problem in the same way for both ITD and AID methods. In particular, in LR we use the gradient descent method with optimal step size as in case (a) of Section 2.1, while for the other cases we use the heavy-ball method with optimal step size and momentum constants. Note that this last method is not a contraction in the original norm, but only in a suitable norm depending on the lower-level problem itself. To compute the exact hypergradient, we differentiate $f(\lambda)$ directly using RMAD for KRR, BR and HR, where the closed form expression for $w(\lambda)$ is available, while for LR we use CG with $t = k = 2000$ in place of the (unavailable) analytic gradient.

Figure 1 shows how the approximation error is affected by the number of lower-level iterations t. As suggested by the iteration complexity bounds in Section 2, all the approximations, after a certain number of iterations, converge linearly to the true hypergradient[5]. Furthermore, in line with our analysis (see Section 2.3), CG gives the best gradient estimate (on average), followed by FP, while ITD performs the worst. For HR, the error of all the methods increases significantly at the beginning, which can be explained by the fact that the heavy ball method is not a contraction in the original norm and may diverge at first. CG $k = 10$ outperforms FP $k = 10$ on 3 out of 4 settings but both remain far from convergence.

3.2. Bilevel Optimization

In this section, we aim to solve instances of the bilevel problem (1) in which λ has a high dimensionality.

Kernel Ridge Regression on Parkinson. We take $f(\beta, \gamma)$ as defined in problem (22) where the data is taken from the UCI Parkinson dataset (Little et al., 2008), containing 195 biomedical voice measurements (22 features) from people with Parkinson's disease. To avoid projections, we replace β and γ respectively with $\exp(\beta)$ and $\exp(\gamma)$ in the RHS of the two equations in (22). We split the data randomly into three equal parts to make the train, validation and test sets.

[4] In multi-task/meta-learning the upper-level objectives are averaged over multiple tasks and the hypergradient is simply the average of the single task one.

[5] The asymptotic error can be quite large probably due to numerical errors (more details in Appendix C).

Table 1. Objective (test accuracy) values after s gradient descent steps where s is 1000, 500 and 4000 for Parkinson, 20 newsgroup and Fashion MNIST respectively. Test accuracy values are in %. $k_r = 10$ for Parkinson and 20 newsgroup while for Fashion MNIST $k_r = 5$.

	Parkinson		**20 newsgroup**			**Fashion MNIST**	
	$t = 100$	$t = 150$	$t = 10$	$t = 25$	$t = 50$	$t = 5$	$t = 10$
ITD	2.39 (75.8)	2.11 (69.7)	1.08 (61.3)	0.97 (62.8)	0.89 (64.2)	0.41 (84.1)	0.43 (83.8)
FP $k = t$	2.37 (81.8)	2.20 (77.3)	1.03 (62.1)	1.02 (62.3)	0.84 (64.4)	0.41 (84.1)	0.43 (83.8)
CG $k = t$	2.37 (78.8)	2.20 (77.3)	0.93 (63.7)	0.78 (63.3)	0.64 (63.1)	0.42 (83.9)	0.42 (84.0)
FP $k = k_r$	2.71 (80.3)	2.60 (78.8)	–	0.94 (63.6)	0.97 (63.0)	–	0.42 (83.9)
CG $k = k_r$	2.33 (77.3)	2.02 (77.3)	–	0.82 (64.3)	0.75 (64.2)	–	0.42 (84.0)

Logistic Regression on 20 Newsgroup[6]. This dataset contains 18000 news divided in 20 topics and the features consist in 101631 tf-idf sparse vectors. We split the data randomly into three equal parts for training, validation and testing. We aim to solve the bilevel problem

$$\min_{\lambda \in \mathbb{R}^p} \mathrm{CE}(X' w(\lambda), y')$$
$$w(\lambda) = \operatorname*{argmin}_{w \in \mathbb{R}^{p \times c}} \mathrm{CE}(Xw, y) + \frac{1}{2cp} \sum_{i=1}^{c} \sum_{j=1}^{p} \exp(\lambda_j) w_{ij}^2$$

where CE is the average cross-entropy loss, $c = 20$ and $p = 101631$. To improve the performance, we use warm-starts on the lower-level problem, i.e. we take $w_0(\lambda_i) = w_t(\lambda_{i-1})$ for all methods, where $(\lambda_i)_{i=1}^s$ are the upper-level iterates.

Training Data Optimization on Fashion MNIST. Similarly to (Maclaurin et al., 2015), we optimize the features of a set of 10 training points, each with a different class label on the Fashion MNIST dataset (Xiao et al., 2017). More specifically we define the bilevel problem as

$$\min_{X \in \mathbb{R}^{c \times p}} \mathrm{CE}(X' w(X), y')$$
$$w(X) = \arg \min_{w \in \mathbb{R}^{p \times c}} \mathrm{CE}(Xw, y) + \frac{\beta}{2cp} \|w\|^2$$

where $\beta = 1$, $c = 10$, $p = 784$, $y = (0, \dots, c)^\top$ and (X', y') contains the usual training set.

We solve each problem using (hyper)gradient descent with fixed step size selected via grid search (additional details are provided in Appendix C.2). The results in Table 1 show the upper-level objective and test accuracy both computed on the approximate lower-level solution $w_t(\lambda)$ after bilevel optimization[7]. For Parkinson and Fashion MNIST, there is little difference among the methods for a fixed t. For 20 newsgroup, CG $k = t$ reaches the lowest objective value, followed by CG $k = 10$. We recall that for ITD we have cost in memory which is linear in t and that, in the case of 20 newsgroup for some t between 50 and 100, this cost exceeded the 11GB on the GPU. AID methods instead, require little memory and, by setting $k < t$, yield similar or even better performance at a lower computation time. Finally, we stress that since the upper-level objective is nonconvex, possibly with several minima, gradient descent with a more precise estimate of the hypergradient may get more easily trapped in a bad local minima.

Equilibrium Models. Our last set of experiments investigates the behaviour of the hypergradient approximation methods on a simple instance of EQM (see problem (3)) on non-structured data. EQM are an attractive class of models due to their mathematical simplicity, enhanced interpretability and memory efficiency. A number of works (Miller & Hardt, 2019; Bai et al., 2019) have recently shown that EQMs can perform on par with standard deep nets on a variety of complex tasks, renewing the interest in these kind of models.

We use a subset of $n_e = 5000$ instances randomly sampled from the MNIST dataset as training data and employ a multiclass logistic classifier paired with a cross-entropy loss. We picked a small training set and purposefully avoided stochastic optimization methods to better focus on issues related to the computation of the hypergradients itself, avoiding the introduction of other sources of noise. We parametrize ϕ_i as

$$\phi_i(w_i, \gamma) = \tanh(Aw_i + Bx_i + c) \quad \text{for } 1 \le i \le n_e \tag{23}$$

where $x_i \in \mathbb{R}^p$ is the i-th example, $w_i \in \mathbb{R}^h$ and $\gamma = (A, B, C) \in \mathbb{R}^{h \times h} \times \mathbb{R}^{h \times p} \times \mathbb{R}^h$. Such a model may be viewed as a (infinite layers) feed-forward neural network with tied weights or as a recurrent neural network with static inputs. Additional experiments with convolutional equilibrium models may be found in Appendix C.3. Imposing $\|A\| < 1$ ensures that the transition functions (23), and hence Φ, are contractions. This can be achieved during optimization by projecting the singular values of A onto the interval $[0, 1 - \varepsilon]$ for $\varepsilon > 0$. We note that regularizing the norm of $\partial_1 \phi_i$ or adding L_1 or L_∞ penalty terms on A may encourage, but does not strictly enforce, $\|A\| < 1$.

We conducted a series of experiments to ascertain the importance of the contractiveness of the map Φ, as well as to understand which of the analysed methods is to be pre-

[6] http://qwone.com/ jason/20Newsgroups/

[7] For completeness, we also report in the Appendix (Table 2) the upper-level objective and test accuracy both computed on the exact lower-level solution $w(\lambda)$.

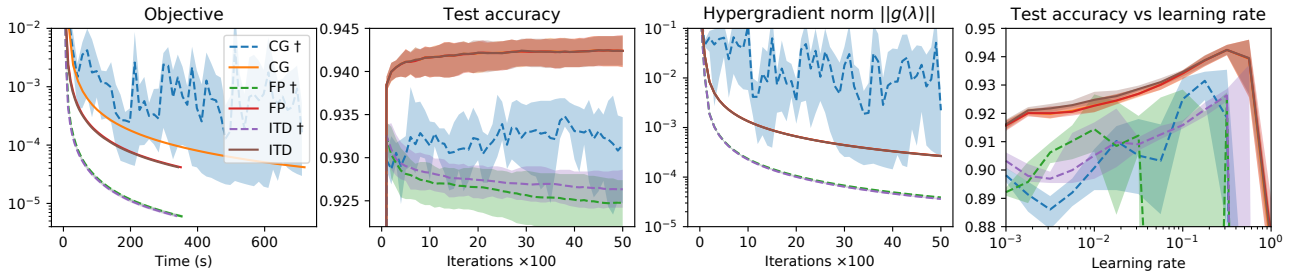

Figure 2. Experiments on EQM problems. Mean (solid or dashed lines) and point-wise minimum-maximum range (shaded regions) across 5 random seeds that only control the initialization of λ. The estimated hypergradient $g(\lambda)$ is equal to $\nabla f_t(\lambda)$ for ITD and $\hat{\nabla} f(\lambda)$ for AID. We used $t = k = 20$ for all methods and Nesterov momentum for optimizing λ, applying a projection operator at each iteration except for the methods marked with †. When performing projection, the curves produced by the three approximation schemes mostly overlap, indicating essentially the same performance (although at a different computational cost).

ferred in this setting. Since here $\partial_1 \Phi$ is not symmetric, the conjugate gradient method must be applied on the normal equations of problem (15). We set $h = 200$ and use $t = 20$ fixed-point iterations to solve the lower-level problem in all the experiments. The first three plots of Figure 2 report training objectives, test accuracies and norms of the estimated hypergradient for each of the three methods, either applying or not the constraint on A, while the last explores the sensitivity of the methods to the choice of the learning rate. Unconstrained runs are marked with †. Referring to the rightmost plot, it is clear (large shaded regions) that not constraining the spectral norm results in unstable behaviour of the "memory-less" AID methods (green and blue lines) for all but a few learning rates, while ITD (violet), as expected, suffers comparatively less. On the contrary, when $\|A\| < 1$ is enforced, all the approximation methods are successful and stable, with FP to be preferred being faster then CG on the normal equations and requiring substantially less memory than ITD. As a side note, referring to Figure 2 left and center-left, we observe that projecting onto the spectral ball acts as powerful regularizer, in line with the findings of Sedghi et al. (2019).

4. Conclusions

We studied a general class of bilevel problems where at the lower-level we seek for a solution to a parametric fixed point equation. This formulation encompasses several learning algorithms recently considered in the literature. We established results on the iteration complexity of two strategies to compute the hypergradient (ITD and AID) under the assumption that the fixed point equation is defined by a contraction mapping. Our practical experience with these methods on a number of bilevel problems indicates that there is a trade-off between the methods, with AID based on the conjugate gradient method being preferable due to a potentiality better approximation of the hypergradient and lower space complexity. When the contraction assumption is not satisfied, however, our experiments on equilibrium models suggest that ITD is more reliable than AID methods. In the future, it would be valuable to extend the ideas presented here to other challenging machine learning scenarios not covered by our theoretical analysis. These include bilevel problems in which the lower-level is only locally contactive, nonsmooth, possibly nonexpansive or can only be solved via a stochastic procedure. At the same time, there is a need to clarify the tightness of the iteration complexity bounds presented here.

Acknowledgments. This work was supported in part by a grant from SAP SE.

References

Almeida, L. B. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In *Proceedings, 1st First International Conference on Neural Networks*, volume 2, pp. 609–618, 1987.

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

Amos, B. and Kolter, J. Z. Optnet: Differentiable optimization as a layer in neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 136–145, 2017.

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and De Freitas, N. Learning to learn by gradient descent by gradient descent. In *Advances in neural information processing systems*, pp. 3981–3989, 2016.

Bai, S., Kolter, J. Z., and Koltun, V. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, pp. 688–699, 2019.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18(153), 2018.

Bergstra, J. and Bengio, Y. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

Bertinetto, L., Henriques, J. F., Torr, P. H., and Vedaldi, A. Meta-learning with differentiable closed-form solvers. *ICLR*, 2019.

Denevi, G., Ciliberto, C., Grazzi, R., and Pontil, M. Learning-to-learn stochastic gradient descent with biased regularization. In *Proc. 36th International Conference on Machine Learning*, volume 97 of *PMLR*, pp. 1566–1575, 2019.

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

Elsken, T., Metzen, J. H., and Hutter, F. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135, 2017.

Franceschi, L., Donini, M., Frasconi, P., and Pontil, M. Forward and reverse gradient-based hyperparameter optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1165–1173, 2017.

Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., and Pontil, M. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*, pp. 1563–1572, 2018.

Griewank, A. and Walther, A. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, volume 105. Siam, 2008.

Liao, R., Xiong, Y., Fetaya, E., Zhang, L., Yoon, K., Pitkow, X., Urtasun, R., and Zemel, R. Reviving and improving recurrent back-propagation. In *International Conference on Machine Learning*, pp. 3088–3097, 2018.

Little, M., McSharry, P., Hunter, E., Spielman, J., and Ramig, L. Suitability of dysphonia measurements for telemonitoring of parkinson's disease. *Nature Precedings*, pp. 1–1, 2008.

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

Lorraine, J., Vicol, P., and Duvenaud, D. Optimizing millions of hyperparameters by implicit differentiation. *arXiv preprint arXiv:1911.02590*, 2019.

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

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

Miller, J. and Hardt, M. Stable recurrent models. *ICLR*, 2019.

Nesterov, Y. E. A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. akad. nauk Sssr*, volume 269, pp. 543–547, 1983.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. 2019.

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

Pineda, F. J. Generalization of back-propagation to recurrent neural networks. *Physical review letters*, 59(19):2229, 1987.

Polyak, B. T. *Introduction to Optimization*. Optimization Software Inc. Publication Division, New York, NY, USA, 1987.

Rajeswaran, A., Finn, C., Kakade, S. M., and Levine, S. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems*, pp. 113–124, 2019.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

Sedghi, H., Gupta, V., and Long, P. M. The singular values of convolutional layers. *ICLR*, 2019.

Shaban, A., Cheng, C.-A., Hatch, N., and Boots, B. Truncated back-propagation for bilevel optimization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1723–1732, 2019.

Snoek, J., Larochelle, H., and Adams, R. P. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pp. 2951–2959, 2012.

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