Analyzing Inexact Hypgergradients for Bilevel Learning

Joint work with Matthias Ehrhardt (Bath)

Lindon Roberts, University of Sydney (lindon.roberts@sydney.edu.au)

SIAM Conference on Optimization (Seattle)
1 June 2023

Outline

1. **Motivation: Bilevel learning**
2. Hypergradient algorithms
3. Unified perspective
4. Numerical results

Bilevel Learning

Goal

Can we use a data-driven approach to tune hyperparameters for inverse problems (e.g. regularization weight)?

Suppose we have training data $(x_1, y_1), \ldots, (x_n, y_n)$ — ground truth and noisy observations.

Attempt to recover x_i from y_i by solving inverse problem with hyperparameters θ:

$$\hat{x}_i(\theta) := \arg\min_x \Phi_i(x, \theta), \qquad \text{e.g. } \Phi_i(x, \theta) = \mathcal{D}(Ax, y_i) + \theta \mathcal{R}(x).$$

Try to find θ by making $\hat{x}_i(\theta)$ close to x_i

$$\min_\theta \frac{1}{n} \sum_{i=1}^{n} \|\hat{x}_i(\theta) - x_i\|^2.$$

Bilevel Optimization

The bilevel learning problem is:

$$\min_{\theta} \quad f(\theta) := \frac{1}{n}\sum_{i=1}^{n} \|\hat{x}_i(\theta) - x_i\|^2 + \mathcal{J}(\theta),$$
$$\text{s.t.} \quad \hat{x}_i(\theta) := \arg\min_{x} \Phi_i(x,\theta), \quad \forall i = 1, \ldots, n.$$

- If Φ_i are strongly convex in x and sufficiently smooth in x and θ, then $\hat{x}_i(\theta)$ is well-defined and continuously differentiable.
- Upper-level problem ($\min_\theta f(\theta)$) is a smooth nonconvex optimization problem

Problem

Existing methods assume access to exact f and ∇f, but cannot compute $\hat{x}_i(\theta)$ exactly!

[e.g. Kunisch & Pock (2013), Sherry et al. (2020)]

Outline

1. Motivation: Bilevel learning
2. **Hypergradient algorithms**
3. Unified perspective
4. Numerical results

Hypergradient

Consider the simple bilevel problem:

$$\min_{\theta \in \mathbb{R}^n} \quad F(\theta) := f(x^*(\theta)), \qquad \text{s.t.} \quad x^*(\theta) := \arg\min_{y \in \mathbb{R}^d} g(y, \theta).$$

Theorem (Inverse Function Theorem)

If g sufficiently smooth (in y and θ) and strongly convex in y, then $\theta \mapsto x^*(\theta)$ is continuously differentiable with

$$\nabla x^*(\theta) = -[\partial_{yy} g(x^*(\theta), \theta)]^{-1} \partial_y \partial_\theta g(x^*(\theta), \theta) \in \mathbb{R}^{d \times n}$$

Hypergradient

Consider the simple bilevel problem:

$$\min_{\theta \in \mathbb{R}^n} \quad F(\theta) := f(x^*(\theta)), \qquad \text{s.t.} \quad x^*(\theta) := \arg\min_{y \in \mathbb{R}^d} g(y, \theta).$$

Theorem (Inverse Function Theorem)

If g sufficiently smooth (in y and θ) and strongly convex in y, then $\theta \mapsto x^*(\theta)$ is continuously differentiable with

$$\nabla x^*(\theta) = -[\partial_{yy} g(x^*(\theta), \theta)]^{-1} \partial_y \partial_\theta g(x^*(\theta), \theta) \in \mathbb{R}^{d \times n}$$

This gives us the exact hypergradient

$$\nabla F(\theta) = -[\partial_y \partial_\theta g(x^*(\theta), \theta)]^T [\partial_{yy} g(x^*(\theta), \theta)]^{-1} \nabla f(x^*(\theta))$$

Hypergradient Computation

The exact hypergradient is

$$\nabla F(\theta) = -[\partial_y \partial_\theta g(x^*(\theta), \theta)]^T [\partial_{yy} g(x^*(\theta), \theta)]^{-1} \nabla f(x^*(\theta))$$

Hypergradient Computation

The exact hypergradient is

$$\nabla F(\theta) = -[\partial_y \partial_\theta g(x^*(\theta), \theta)]^T [\partial_{yy} g(x^*(\theta), \theta)]^{-1} \nabla f(x^*(\theta))$$

- We can never evaluate $x^*(\theta)$ exactly (minimizer of g).
- If dimension of y is large, solve linear system inexactly ($\partial_{yy} g$ is SPD so use CG)

Hypergradient Computation

The exact hypergradient is

$$\nabla F(\theta) = -[\partial_y \partial_\theta g(x^*(\theta), \theta)]^T [\partial_{yy} g(x^*(\theta), \theta)]^{-1} \nabla f(x^*(\theta))$$

- We can never evaluate $x^*(\theta)$ exactly (minimizer of g).
- If dimension of y is large, solve linear system inexactly ($\partial_{yy} g$ is SPD so use CG)

Inverse Function Theorem (+ CG) approach:

1. Solve lower-level problem to get x^*_ε such that $\|x^*_\varepsilon - x^*(\theta)\| \le \varepsilon$
2. Using CG, find $q_{\varepsilon,\delta}$ such that $\|[\partial_{yy} g(x^*_\varepsilon, \theta)] q_{\varepsilon,\delta} - \nabla f(x^*_\varepsilon)\| \le \delta$.
3. Return hypergradient estimate $h_{\varepsilon,\delta} := -[\partial_y \partial_\theta g(x^*_\varepsilon, \theta)]^T q_{\varepsilon,\delta}$.

Hypergradient Computation

The exact hypergradient is

$$\nabla F(\theta) = -[\partial_y \partial_\theta g(x^*(\theta), \theta)]^T [\partial_{yy} g(x^*(\theta), \theta)]^{-1} \nabla f(x^*(\theta))$$

- We can never evaluate $x^*(\theta)$ exactly (minimizer of g).
- If dimension of y is large, solve linear system inexactly ($\partial_{yy} g$ is SPD so use CG)

Inverse Function Theorem (+ CG) approach:

1. Solve lower-level problem to get x_ε^* such that $\|x_\varepsilon^* - x^*(\theta)\| \le \varepsilon$
2. Using CG, find $q_{\varepsilon,\delta}$ such that $\|[\partial_{yy} g(x_\varepsilon^*, \theta)] q_{\varepsilon,\delta} - \nabla f(x_\varepsilon^*)\| \le \delta$.
3. Return hypergradient estimate $h_{\varepsilon,\delta} := -[\partial_y \partial_\theta g(x_\varepsilon^*, \theta)]^T q_{\varepsilon,\delta}$.

Theorem (Pedregosa (2016); Zucchet & Sacramento (2022))

If ε is sufficiently small, then $\|h_{\varepsilon,\delta} - \nabla F(\theta)\| = \mathcal{O}(\varepsilon + \delta)$.

Iterative AD

An alternative approach for calculating $\nabla F(\theta)$ is to use automatic differentiation (AD).

Iterative AD

An alternative approach for calculating $\nabla F(\theta)$ is to use automatic differentiation (AD).

Given some algorithm for approximating $x^*(\theta) := \arg\min_{y\in\mathbb{R}^d} g(y,\theta)$, we can apply AD to that algorithm. [Christianson (1994)]

Iterative AD

An alternative approach for calculating $\nabla F(\theta)$ is to use automatic differentiation (AD).

Given some algorithm for approximating $x^*(\theta) := \arg\min_{y\in\mathbb{R}^d} g(y,\theta)$, we can apply AD to that algorithm. [Christianson (1994)]

For example, run K iterations of GD with fixed stepsize starting from $x^{(0)}$:

$$x^{(k+1)} = x^{(k)} - \alpha\partial_y g(x^{(k)},\theta), \qquad k = 0,\dots,K-1.$$

Our estimate is $x^{(K)} \approx x^*(\theta)$.

Iterative AD

An alternative approach for calculating $\nabla F(\theta)$ is to use automatic differentiation (AD).

Given some algorithm for approximating $x^*(\theta) := \arg\min_{y \in \mathbb{R}^d} g(y, \theta)$, we can apply AD to that algorithm. [Christianson (1994)]

For example, run K iterations of GD with fixed stepsize starting from $x^{(0)}$:

$$x^{(k+1)} = x^{(k)} - \alpha \partial_y g(x^{(k)}, \theta), \qquad k = 0, \ldots, K-1.$$

Our estimate is $x^{(K)} \approx x^*(\theta)$. Reverse mode AD on this iteration then gives:

- Forward pass: define adjoint variables $\widetilde{x}^{(0)} := \nabla f(x^{(K)})$ and

$$\widetilde{x}^{(K-k-1)} = \widetilde{x}^{(K-k)} - \alpha [\partial_{yy} g(x^{(K-k-1)}, \theta)] \widetilde{x}^{(K-k)}.$$

- Backward pass: $h^{(0)} := 0 \in \mathbb{R}^n$ and

$$h^{(k+1)} = h^{(k)} - \alpha [\partial_y \partial_\theta g(x^{(K-k-1)}, \theta)]^T \widetilde{x}^{(K-k)}. \qquad [\rightarrow \text{return } h^{(K)}]$$

Iterative AD

We are solving the lower-level problem with GD ($x^{(K)} \approx x^*(\theta)$):

$$x^{(k+1)} = x^{(k)} - \alpha \partial_y g(x^{(k)}, \theta), \qquad k = 0, \ldots, K-1,$$

with corresponding AD iteration ($h^{(K)} \approx \nabla F(\theta)$)

$$\begin{aligned} h^{(k+1)} &= h^{(k)} - \alpha [\partial_y \partial_\theta g(x^{(K-k-1)}, \theta)]^T \widetilde{x}^{(K-k)}, \\ \widetilde{x}^{(K-k-1)} &= \widetilde{x}^{(K-k)} - \alpha [\partial_{yy} g(x^{(K-k-1)}, \theta)] \widetilde{x}^{(K-k)}. \end{aligned}$$

Iterative AD

We are solving the lower-level problem with GD ($x^{(K)} \approx x^*(\theta)$):

$$x^{(k+1)} = x^{(k)} - \alpha \partial_y g(x^{(k)}, \theta), \qquad k = 0, \dots, K-1,$$

with corresponding AD iteration ($h^{(K)} \approx \nabla F(\theta)$)

$$h^{(k+1)} = h^{(k)} - \alpha [\partial_y \partial_\theta g(x^{(K-k-1)}, \theta)]^T \widetilde{x}^{(K-k)},$$
$$\widetilde{x}^{(K-k-1)} = \widetilde{x}^{(K-k)} - \alpha [\partial_{yy} g(x^{(K-k-1)}, \theta)] \widetilde{x}^{(K-k)}.$$

Since we are solving a smooth, strongly convex problem, if α is small enough then $\|x^{(K)} - x^*(\theta)\| \le \lambda^K \|x^{(0)} - x^*(\theta)\|$ for some $\lambda < 1$.

Iterative AD

We are solving the lower-level problem with GD ($x^{(K)} \approx x^*(\theta)$):

$$x^{(k+1)} = x^{(k)} - \alpha \partial_y g(x^{(k)}, \theta), \qquad k = 0, \ldots, K-1,$$

with corresponding AD iteration ($h^{(K)} \approx \nabla F(\theta)$)

$$h^{(k+1)} = h^{(k)} - \alpha [\partial_y \partial_\theta g(x^{(K-k-1)}, \theta)]^T \widetilde{x}^{(K-k)},$$
$$\widetilde{x}^{(K-k-1)} = \widetilde{x}^{(K-k)} - \alpha [\partial_{yy} g(x^{(K-k-1)}, \theta)] \widetilde{x}^{(K-k)}.$$

Since we are solving a smooth, strongly convex problem, if α is small enough then $\|x^{(K)} - x^*(\theta)\| \leq \lambda^K \|x^{(0)} - x^*(\theta)\|$ for some $\lambda < 1$.

Theorem (Mehmood & Ochs (2020))

The reverse mode AD hypergradient $h^{(K)}$ satisfies $\|h^{(K)} - \nabla F_K\| = \mathcal{O}(K\lambda^K)$, where

$$\nabla F_K := -[\partial_y \partial_\theta g(x^{(K)}, \theta)]^T [\partial_{yy} g(x^{(K)}, \theta)]^{-1} \nabla f(x^{(K)}).$$

Inexact AD

Our full iteration is

$$
\begin{aligned}
h^{(k+1)} &= h^{(k)} - \alpha[\partial_y \partial_\theta g(x^{(K-k-1)}, \theta)]^T \widetilde{x}^{(K-k)}, \\
\widetilde{x}^{(K-k-1)} &= \widetilde{x}^{(K-k)} - \alpha[\partial_{yy} g(x^{(K-k-1)}, \theta)]\widetilde{x}^{(K-k)}.
\end{aligned}
$$

Inexact AD

Our full iteration is

$$h^{(k+1)} = h^{(k)} - \alpha[\partial_y \partial_\theta g(x^{(K-k-1)}, \theta)]^T \widetilde{x}^{(K-k)},$$
$$\widetilde{x}^{(K-k-1)} = \widetilde{x}^{(K-k)} - \alpha[\partial_{yy} g(x^{(K-k-1)}, \theta)]\widetilde{x}^{(K-k)}.$$

We can get a better iteration using inexact AD: evaluate all second derivatives at the best estimate $x^{(K)}$.

$$h^{(k+1)} = h^{(k)} - \alpha[\partial_y \partial_\theta g(x^{(K)}, \theta)]^T \widetilde{x}^{(K-k)},$$
$$\widetilde{x}^{(K-k-1)} = \widetilde{x}^{(K-k)} - \alpha[\partial_{yy} g(x^{(K)}, \theta)]\widetilde{x}^{(K-k)}.$$

Inexact AD

Our full iteration is

$$h^{(k+1)} = h^{(k)} - \alpha[\partial_y \partial_\theta g(x^{(K-k-1)}, \theta)]^T \widetilde{x}^{(K-k)},$$
$$\widetilde{x}^{(K-k-1)} = \widetilde{x}^{(K-k)} - \alpha[\partial_{yy} g(x^{(K-k-1)}, \theta)] \widetilde{x}^{(K-k)}.$$

We can get a better iteration using inexact AD: evaluate all second derivatives at the best estimate $x^{(K)}$.

$$h^{(k+1)} = h^{(k)} - \alpha[\partial_y \partial_\theta g(x^{(K)}, \theta)]^T \widetilde{x}^{(K-k)},$$
$$\widetilde{x}^{(K-k-1)} = \widetilde{x}^{(K-k)} - \alpha[\partial_{yy} g(x^{(K)}, \theta)] \widetilde{x}^{(K-k)}.$$

Theorem (Mehmood & Ochs (2020))

The inexact AD hypergradient $h^{(K)}$ satisfies $\|h^{(K)} - \nabla F_K\| = \mathcal{O}(\lambda^K)$.

Inexact AD

Our full iteration is

$$h^{(k+1)} = h^{(k)} - \alpha[\partial_y \partial_\theta g(x^{(K-k-1)}, \theta)]^T \widetilde{x}^{(K-k)},$$
$$\widetilde{x}^{(K-k-1)} = \widetilde{x}^{(K-k)} - \alpha[\partial_{yy} g(x^{(K-k-1)}, \theta)]\widetilde{x}^{(K-k)}.$$

We can get a better iteration using inexact AD: evaluate all second derivatives at the best estimate $x^{(K)}$.

$$h^{(k+1)} = h^{(k)} - \alpha[\partial_y \partial_\theta g(x^{(K)}, \theta)]^T \widetilde{x}^{(K-k)},$$
$$\widetilde{x}^{(K-k-1)} = \widetilde{x}^{(K-k)} - \alpha[\partial_{yy} g(x^{(K)}, \theta)]\widetilde{x}^{(K-k)}.$$

Theorem (Mehmood & Ochs (2020))

The inexact AD hypergradient $h^{(K)}$ satisfies $\|h^{(K)} - \nabla F_K\| = \mathcal{O}(\lambda^K)$.

Note: Similar results hold using heavy ball (Polyak) momentum instead of GD.

Outline

1. Motivation: Bilevel learning
2. Hypergradient algorithms
3. **Unified perspective**
4. Numerical results

Unified Perspective

Questions

Two questions of interest:

1. What is the relationship (if any) between inexact AD and IFT+CG?
2. Can we get computable error bounds for these methods?

Unified Perspective

Questions

Two questions of interest:

1. What is the relationship (if any) between inexact AD and IFT+CG?
2. Can we get computable error bounds for these methods?

Motivation for #2: algorithms for smooth nonconvex problems with inexact gradients typically require conditions such as

- $\|h_k - \nabla F(\theta_k)\| \leq C\|h_k\|$ for some (fixed) $C < 1$ [Berahas et al. (2021)]
- $\|h_k - \nabla F(\theta_k)\| \leq C_k$, for some (dynamically updated) $C_k > 0$ [Cao et al. (2022)]

We need some way to verify these (and solve to higher accuracy if not satisfied).

Key Insight

Inexact AD: given $x^{(K)} \approx x^*(\theta)$ from K iterations of GD, iterate

$$h^{(k+1)} = h^{(k)} - \alpha[\partial_y \partial_\theta g(x^{(K)}, \theta)]^T \widetilde{x}^{(K-k)},$$
$$\widetilde{x}^{(K-k-1)} = \widetilde{x}^{(K-k)} - \alpha[\partial_{yy} g(x^{(K)}, \theta)]\widetilde{x}^{(K-k)}.$$

Key Insight

Inexact AD: given $x^{(K)} \approx x^*(\theta)$ from K iterations of GD, iterate

$$h^{(k+1)} = h^{(k)} - \alpha[\partial_y \partial_\theta g(x^{(K)}, \theta)]^T \widetilde{x}^{(K-k)},$$
$$\widetilde{x}^{(K-k-1)} = \widetilde{x}^{(K-k)} - \alpha[\partial_{yy} g(x^{(K)}, \theta)]\widetilde{x}^{(K-k)}.$$

Rearrange to reduce Jacobian-vector products (and re-index $\widetilde{x}$)

$$q^{(k+1)} = q^{(k)} + \alpha \widetilde{x}^{(k)},$$
$$\widetilde{x}^{(k+1)} = \widetilde{x}^{(k)} - \alpha[\partial_{yy} g(x^{(K)}, \theta)]\widetilde{x}^{(k)},$$

with $q^{(0)} = 0$. Final estimate is $h^{(K)} = -[\partial_y \partial_\theta g(x^{(K)}, \theta)]^T q^{(K)}$.

Key Insight

Inexact AD: given $x^{(K)} \approx x^*(\theta)$ from K iterations of GD, iterate

$$h^{(k+1)} = h^{(k)} - \alpha[\partial_y \partial_\theta g(x^{(K)}, \theta)]^T \widetilde{x}^{(K-k)},$$
$$\widetilde{x}^{(K-k-1)} = \widetilde{x}^{(K-k)} - \alpha[\partial_{yy} g(x^{(K)}, \theta)] \widetilde{x}^{(K-k)}.$$

Rearrange to reduce Jacobian-vector products (and re-index $\widetilde{x}$)

$$q^{(k+1)} = q^{(k)} + \alpha \widetilde{x}^{(k)},$$
$$\widetilde{x}^{(k+1)} = \widetilde{x}^{(k)} - \alpha[\partial_{yy} g(x^{(K)}, \theta)] \widetilde{x}^{(k)},$$

with $q^{(0)} = 0$. Final estimate is $h^{(K)} = -[\partial_y \partial_\theta g(x^{(K)}, \theta)]^T q^{(K)}$.

Relabel $\partial_{yy} g(x^{(K)}, \theta) \to A$, $\widetilde{x}^{(k)} \to r^{(k)}$ and $q^{(k)} \to x^{(k)}$, and it is more familiar:

$$x^{(k+1)} = x^{(k)} + \alpha r^{(k)}, \qquad \text{and} \qquad r^{(k+1)} = r^{(k)} - \alpha A r^{(k)}$$

IFT vs. inexact AD

Theorem (Ehrhardt & R. (2023))

Inexact AD is exactly equivalent to using K iterations of GD with stepsize α to solve the symmetric positive definite linear system

$$[\partial_{yy} g(x^{(K)}, \theta)]q = \nabla f(x^{(K)}) \quad \iff \quad \min_q \frac{1}{2} q^T [\partial_{yy} g] q - \nabla f(x^{(K)})^T q,$$

starting from $q^{(0)} = 0$, and returning $-[\partial_y \partial_\theta g(x^{(K)}, \theta)]^T q^{(K)}$.

IFT vs. inexact AD

Theorem (Ehrhardt & R. (2023))

Inexact AD is exactly equivalent to using K iterations of GD with stepsize α to solve the symmetric positive definite linear system

$$[\partial_{yy} g(x^{(K)}, \theta)]q = \nabla f(x^{(K)}) \quad \Longleftrightarrow \quad \min_q \frac{1}{2} q^T [\partial_{yy} g] q - \nabla f(x^{(K)})^T q,$$

starting from $q^{(0)} = 0$, and returning $-[\partial_y \partial_\theta g(x^{(K)}, \theta)]^T q^{(K)}$.

So inexact AD is exactly an IFT method in disguise!

IFT vs. inexact AD

Theorem (Ehrhardt & R. (2023))

Inexact AD is exactly equivalent to using K iterations of GD with stepsize α to solve the symmetric positive definite linear system

$$[\partial_{yy} g(x^{(K)}, \theta)]q = \nabla f(x^{(K)}) \quad \Longleftrightarrow \quad \min_q \frac{1}{2} q^T [\partial_{yy} g] q - \nabla f(x^{(K)})^T q,$$

starting from $q^{(0)} = 0$, and returning $-[\partial_y \partial_\theta g(x^{(K)}, \theta)]^T q^{(K)}$.

So inexact AD is exactly an IFT method in disguise!

An equivalent result holds for inexact AD using heavy ball momentum instead of GD.

Unified Framework

This motivates a general hypergradient approximation framework:

1. Solve the lower-level problem to get x_ε^* such that $\|x_\varepsilon^* - x^*\| \leq \varepsilon$
2. Find $q_{\varepsilon,\delta}$ such that $\|[\partial_{yy} g(x_\varepsilon^*, \theta)] q_{\varepsilon,\delta} - \nabla f(x_\varepsilon^*)\| \leq \delta$.
3. Return hypergradient estimate $h_{\varepsilon,\delta} := -[\partial_y \partial_\theta g(x_\varepsilon^*, \theta)]^T q_{\varepsilon,\delta}$.

This is IFT+CG, but any algorithm can be used in the first two steps (and they don't have to be the same).

Unified Framework

This motivates a general hypergradient approximation framework:

1. Solve the lower-level problem to get x_ε^* such that $\|x_\varepsilon^* - x^*\| \leq \varepsilon$
2. Find $q_{\varepsilon,\delta}$ such that $\|[\partial_{yy} g(x_\varepsilon^*, \theta)] q_{\varepsilon,\delta} - \nabla f(x_\varepsilon^*)\| \leq \delta$.
3. Return hypergradient estimate $h_{\varepsilon,\delta} := -[\partial_y \partial_\theta g(x_\varepsilon^*, \theta)]^T q_{\varepsilon,\delta}$.

This is IFT+CG, but any algorithm can be used in the first two steps (and they don't have to be the same).

Covers IFT+CG and inexact AD methods (and AD methods don't have to be exactly K iterations in both passes).

Unified Framework

This motivates a general hypergradient approximation framework:

1. Solve the lower-level problem to get x_ε^* such that $\|x_\varepsilon^* - x^*\| \leq \varepsilon$
2. Find $q_{\varepsilon,\delta}$ such that $\|[\partial_{yy} g(x_\varepsilon^*, \theta)] q_{\varepsilon,\delta} - \nabla f(x_\varepsilon^*)\| \leq \delta$.
3. Return hypergradient estimate $h_{\varepsilon,\delta} := -[\partial_y \partial_\theta g(x_\varepsilon^*, \theta)]^T q_{\varepsilon,\delta}$.

This is IFT+CG, but any algorithm can be used in the first two steps (and they don't have to be the same).

Covers IFT+CG and inexact AD methods (and AD methods don't have to be exactly K iterations in both passes).

Theorem (Ehrhardt & R. (2023))

We have $\|h_{\varepsilon,\delta} - \nabla F(\theta)\| = \mathcal{O}(\varepsilon + \delta + \varepsilon^2 + \delta\varepsilon)$. *Holds for any* $\varepsilon > 0$ *(new!).*

Error Bounds

Interested in two types of error bounds:

- A priori: based on known linear convergence rates (e.g. λ^k)
- A posteriori: computable based on known quantities (e.g. $\|\partial_y g(x_\varepsilon^*, \theta)\|$)

Error Bounds

Interested in two types of error bounds:

- A priori: based on known linear convergence rates (e.g. λ^k)
- A posteriori: computable based on known quantities (e.g. $\|\partial_y g(x_\varepsilon^*, \theta)\|$)

A priori bounds are $\mathcal{O}(\varepsilon + \delta + \varepsilon^2 + \delta\varepsilon)$ with (for k iterations of linear solve):

$$\begin{aligned} &(\text{IFT+CG}) && \delta \le C_1 \lambda_{\text{CG}}^k, \\ &(\text{AD+GD}) && \delta \le C_2 \lambda_{\text{GD}}^k, \\ &(\text{AD+HB}) && \delta \le C_3 (\lambda_{\text{HB}} + \gamma)^k. \end{aligned}$$

Error Bounds

Interested in two types of error bounds:

- A priori: based on known linear convergence rates (e.g. λ^k)
- A posteriori: computable based on known quantities (e.g. $\|\partial_y g(x_\varepsilon^*, \theta)\|$)

A priori bounds are $\mathcal{O}(\varepsilon + \delta + \varepsilon^2 + \delta\varepsilon)$ with (for k iterations of linear solve):

$$\begin{aligned} &(\text{IFT+CG}) && \delta \leq C_1 \lambda_{\text{CG}}^k, \\ &(\text{AD+GD}) && \delta \leq C_2 \lambda_{\text{GD}}^k, \\ &(\text{AD+HB}) && \delta \leq C_3 (\lambda_{\text{HB}} + \gamma)^k. \end{aligned}$$

Best λ values (depending on α, momentum): $\lambda_{CG} = \lambda_{\text{HB}}^* \ll \lambda_{GD}^*$.

(AD+HB) bound holds for any $\gamma > 0$ but no explicit form for $C_3(\gamma)$.

Error Bounds

A posteriori bounds look like:

- Use $G_\varepsilon := \|\partial_y g(x_\varepsilon^*, \theta)\|$ to measure accuracy of lower-level solve.
- Use current residual $R_{\varepsilon,\delta} := \|[\partial_{yy} g(x_\varepsilon^*, \theta)] q_{\varepsilon,\delta} - \nabla f(x_\varepsilon^*)\|$ to estimate accuracy of hypergradient.
- Overall bound is of the form

$$\|h_{\varepsilon,\delta} - \nabla F(\theta)\| \leq \mathcal{O}(R_{\varepsilon,\delta} + G_\varepsilon + G_\varepsilon^2),$$

where all constants are computable (i.e. only depend on $x_{\varepsilon,\delta}$, $q_{\varepsilon,\delta}$ and various Lipschitz constants, not x^*).

Outline

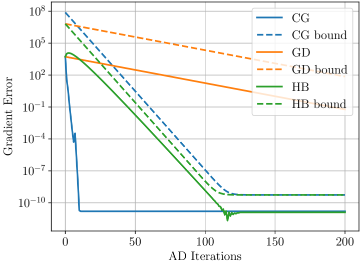
1. Motivation: Bilevel learning
2. Hypergradient algorithms
3. Unified perspective
4. **Numerical results**

Simple Problem

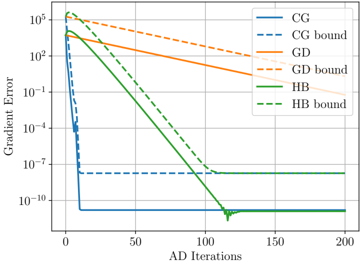
Simple least-squares test problem: [Li et al. (2022)]

$$\min_{\theta \in \mathbb{R}^n} \quad F(\theta) := ||Ax^*(\theta) - b||_2^2 \qquad \text{s.t.} \qquad x^*(\theta) := \arg\min_{y \in \mathbb{R}^d} ||C\theta + Dy - e||_2^2.$$

(analytic expression for $x^*(\theta)$, problem constants easy to evaluate)



A priori bounds



A posteriori bounds

Data Hypercleaning: [Yang et al. (2021)]

- Perform logistic regression on MNIST, but some training labels are corrupted (10%)
- Learn weights for each training example

$$\min_{\theta} \frac{1}{N_{\text{test}}} \sum_i \ell(w^*(\theta), x_i^{\text{test}}, y_i^{\text{test}}),$$

$$\text{s.t. } w^*(\theta) = \arg\min_w \frac{1}{N_{\text{train}}} \sum_j \sigma(\theta_j) \cdot \ell(w, x_j^{\text{train}}, y_j^{\text{train}}) + \alpha \|w\|^2.$$

Data Hypercleaning: [Yang et al. (2021)]

- Perform logistic regression on MNIST, but some training labels are corrupted (10%)
- Learn weights for each training example

$$\min_{\theta} \frac{1}{N_{\text{test}}} \sum_i \ell(w^*(\theta), x_i^{\text{test}}, y_i^{\text{test}}),$$

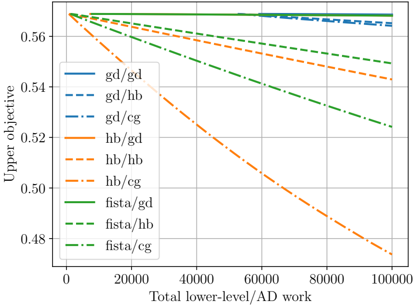
$$\text{s.t. } w^*(\theta) = \arg\min_w \frac{1}{N_{\text{train}}} \sum_j \sigma(\theta_j) \cdot \ell(w, x_j^{\text{train}}, y_j^{\text{train}}) + \alpha \|w\|^2.$$

Question: do better hypergradient methods yield better optimization?

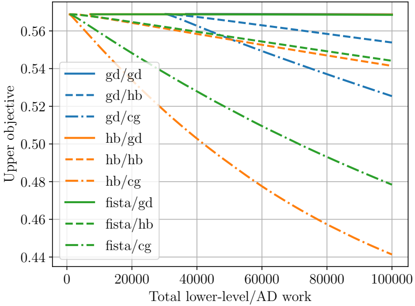
Work: 1 lower-level iter ≈ 1 AD iter (lower-level gradient ≈ Hessian-vector product)

Data Hypercleaning

Data Hypercleaning Results:



$\varepsilon = \mathbf{0.01},\ \delta = \mathbf{0.01}$



$\varepsilon = \mathbf{0.1},\ \delta = \mathbf{0.01}$

Better AD method gives better optimization results (c.f. stochastic gradients).

Conclusions & Future Work

Conclusions

- Can compute hypergradients using either IFT or AD methods
  - Best AD methods are actually a special case of IFT
- Unified analysis and bounds with flexible choice of solvers
- A posteriori bounds computable and more accurate
- Good hypergradient method similarly important as good lower-level solver

Future Work

- Incorporate into rigorous bilevel optimization algorithm
- More sophisticated problems; e.g. neural network regularizers, learning MRI sample patterns

Preprint: https://arxiv.org/abs/2301.04764

References i

A. S. Berahas, L. Cao, and K. Scheinberg, *Global convergence rate analysis of a generic line search algorithm with noise*, SIAM Journal on Optimization, 31 (2021), pp. 1489–1518.

L. Cao, A. S. Berahas, and K. Scheinberg, *First- and second-order high probability complexity bounds for trust-region methods with noisy oracles*, arXiv preprint 2205.03667, (2022).

B. Christianson, *Reverse accumulation and attractive fixed points*, Optimization Methods and Software, 3 (1994), pp. 311–326.

M. J. Ehrhardt and L. Roberts, *Analyzing inexact hypergradients for bilevel learning*, arXiv:2301.04764, (2023).

K. Kunisch and T. Pock, *A Bilevel Optimization Approach for Parameter Learning in Variational Models*, SIAM Journal on Imaging Sciences, 6 (2013), pp. 938–983.

J. Li, B. Gu, and H. Huang, *A fully single loop algorithm for bilevel optimization without Hessian inverse*, Proceedings of the AAAI Conference on Artificial Intelligence, 36 (2022), pp. 7426–7434.

References ii

S. Mehmood and P. Ochs, *Automatic differentiation of some first-order methods in parametric optimization*, in Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS), Palermo, Italy, 2020.

F. Pedregosa, *Hyperparameter optimization with approximate gradient*, in Proceedings of the 33rd International Conference on Machine Learning, New York, 2016.

F. Sherry, M. Benning, J. C. De los Reyes, M. J. Graves, G. Maierhofer, G. Williams, C.-B. Schonlieb, and M. J. Ehrhardt, *Learning the sampling pattern for MRI*, IEEE Transactions on Medical Imaging, 39 (2020), pp. 4310–4321.

J. Yang, K. Ji, and Y. Liang, *Provably faster algorithms for bilevel optimization*, arXiv preprint arXiv:2106.04692, (2021).

N. Zucchet and J. Sacramento, *Beyond backpropagation: implicit gradients for bilevel optimization*, arXiv preprint arXiv:2205.03076, (2022).