# Analyzing Inexact Hypergradients for Bilevel Learning

*Joint work with Matthias Ehrhardt (Bath)*

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

AustMS Annual Meeting (UNSW Sydney)
9 December 2022

# Outline

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

# Variational Regularization

Many inverse problems can be posed in the form

$$\min_x \mathcal{D}(Ax, y) + \alpha \mathcal{R}(x),$$

where we wish to find $x$ given data $y \approx Ax$.

# Variational Regularization

Many inverse problems can be posed in the form

$$\min_x \mathcal{D}(Ax, y) + \alpha\mathcal{R}(x),$$

where we wish to find $x$ given data $y \approx Ax$.

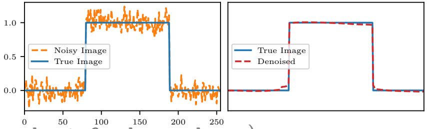
**Example (image denoising):** given a noisy image $y$, find a denoised image $x$ by solving:

$$\min_x \underbrace{\frac{1}{2}\|x - y\|_2^2}_{\mathcal{D}(x,y)} + \alpha \underbrace{\sum_j \sqrt{\|\nabla x_j\|_2^2 + \nu^2}}_{\approx \mathrm{TV}(x)} + \frac{\xi}{2}\|x\|_2^2$$

Solution depends on choices of $\alpha$, $\nu$ and $\xi$:

Example

$(\alpha = 1,\ \nu = \xi = 10^{-3})$



Many inverse problems can be posed in the form

$$\min_x \mathcal{D}(Ax, y) + \alpha \mathcal{R}(x),$$

where we wish to find $x$ given data $y \approx Ax$.

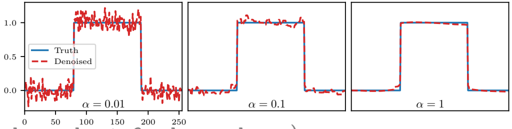
**Example (image denoising):** given a noisy image $y$, find a denoised image $x$ by solving:

$$\min_x \underbrace{\frac{1}{2}\|x-y\|_2^2}_{\mathcal{D}(x,y)} + \alpha \underbrace{\sum_j \sqrt{\|\nabla x_j\|_2^2 + \nu^2}}_{\approx \mathrm{TV}(x)} + \frac{\xi}{2}\|x\|_2^2$$

Solution depends on choices of $\alpha$, $\nu$ and $\xi$:

Vary $\alpha$

($\nu = 10^{-3}$, $\xi = 10^{-3}$)



# Variational Regularization

Many inverse problems can be posed in the form

$$\min_x \mathcal{D}(Ax, y) + \alpha \mathcal{R}(x),$$

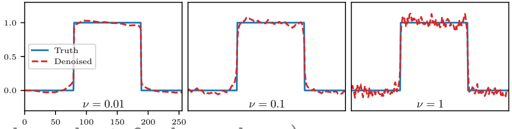
where we wish to find $x$ given data $y \approx Ax$.

**Example (image denoising):** given a noisy image $y$, find a denoised image $x$ by solving:

$$\min_x \underbrace{\frac{1}{2}\|x - y\|_2^2}_{\mathcal{D}(x,y)} + \alpha \underbrace{\sum_j \sqrt{\|\nabla x_j\|_2^2 + \nu^2}}_{\approx \mathrm{TV}(x)} + \frac{\xi}{2}\|x\|_2^2$$

Solution depends on choices of $\alpha$, $\nu$ and $\xi$:

$$\frac{\text{Vary } \nu}{(\alpha = 1,\ \xi = 10^{-3})}$$



# Variational Regularization

Many inverse problems can be posed in the form

$$\min_x \mathcal{D}(Ax, y) + \alpha \mathcal{R}(x),$$

where we wish to find $x$ given data $y \approx Ax$.

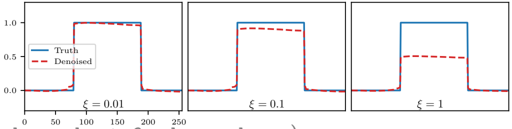
**Example (image denoising):** given a noisy image $y$, find a denoised image $x$ by solving:

$$\min_x \underbrace{\frac{1}{2}\|x - y\|_2^2}_{\mathcal{D}(x,y)} + \alpha \underbrace{\sum_j \sqrt{\|\nabla x_j\|_2^2 + \nu^2}}_{\approx \mathrm{TV}(x)} + \frac{\xi}{2}\|x\|_2^2$$

Solution depends on choices of $\alpha$, $\nu$ and $\xi$:

Vary $\xi$

$(\alpha = 1,\ \nu = 10^{-3})$



# Choosing Parameters

Recovered solution depends strongly on problem parameters (e.g. $\alpha$, $\nu$ and $\xi$)

**Question**

How to choose good problem parameters?

# Choosing Parameters

Recovered solution depends strongly on problem parameters (e.g. $\alpha$, $\nu$ and $\xi$)

**Question**

How to choose good problem parameters?

- Trial & error
- L-curve criterion
- **Bilevel learning** — data-driven approach

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 parameters $\theta \in \mathbb{R}^m$:

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

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

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

with optional (smooth) term $\mathcal{J}(\theta)$ to encourage particular choices of $\theta$.

# 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 $\Phi_i$ are strongly convex in $x$ and sufficiently smooth in $x$ and $\theta$, 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**

Convergent algorithms require exact derivatives of $f(\theta)$, but not available (cannot even compute $\hat{x}_i(\theta)$ exactly)! [e.g. Kunisch & Pock (2013), Sherry et al. (2020)]

# Outline

# 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 $\theta$) 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 $\theta$) 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 $x_\varepsilon^*$ such that $\|x_\varepsilon^* - x^*(\theta)\| \leq \varepsilon$
2. Using CG, 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 $h_{\varepsilon,\delta} := -[\partial_y \partial_\theta g(x_\varepsilon^*, \theta)]^T q_{\varepsilon,\delta}$.

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

# Automatic Differentiation

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

Reverse-mode AD—aka backpropagation—uses the chain rule to build up (symbolic) gradients given source code for function evaluation.

# Automatic Differentiation

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

Reverse-mode AD—aka backpropagation—uses the chain rule to build up (symbolic) gradients given source code for function evaluation.

**Example:** calculate $\nabla f$ for $f(x_1, x_2) = \sin(x_1)/x_2$.

```
def f(x1,x2):      # 1. Forward pass (evaluate f and save adjoint variables)
    x3 = sin(x1);          dx3_dx1 = cos(x1);
    x4 = x3 / x2;          dx4_dx3 = 1/x2;    dx4_dx2 = - x3 / x2**2;
    return x4
```

```
def gradf(x1,x2):  # 2. Backward pass (compute gradient)
    dx1, dx2, dx3 = 0;      dx4  = 1              # dxi = df/dxi
    dx2 += dx4 * dx4_dx2;   dx3 += dx4 * dx4_dx3; # process x4(x2, x3)
    dx1 += dx3 * dx3_dx1;                         # process x3(x1)
    return dx1, dx2
```

# Iterative 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 gradient descent 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 $x^{(K)} \approx x^*(\theta)$.

# Iterative 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 gradient descent 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:

- Initialize $\widetilde{x}^{(0)} := \nabla f(x^{(K)})$ and $h^{(0)} := 0 \in \mathbb{R}^n$.
- For $k = 0, \ldots, K-1$, iterate (backward pass)

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

Final hypergradient is $h^{(K)}$.

# Inexact AD

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

# Inexact AD

Since we are solving a smooth, strongly convex problem, if $\alpha$ 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

Since we are solving a smooth, strongly convex problem, if $\alpha$ 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*

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

This can be improved using inexact AD: evaluate all second derivatives at the best estimate $x^{(K)}$.

**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. 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

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

for $k = 0, \ldots, K-1$, with $\widetilde{x}^{(0)} := \nabla f(x^{(K)})$ and $h^{(0)} := 0 \in \mathbb{R}^n$.

## Key Insight

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

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

for $k = 0, \ldots, K-1$, with $\widetilde{x}^{(0)} := \nabla f(x^{(K)})$ and $h^{(0)} := 0 \in \mathbb{R}^n$.

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

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

for $k = 0, \ldots, K-1$, with $\widetilde{x}^{(0)} := \nabla f(x^{(K)})$ and $h^{(0)} := 0 \in \mathbb{R}^n$.

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

This looks like a GD iteration!

# IFT vs. inexact AD

**Theorem (Ehrhardt & R. (2022))**

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

$$[\partial_{yy} g(x^{(K)}, \theta)] q = \nabla f(x^{(K)}),$$

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

*Note: if $A$ is SPD, then solving $Ax = b$ is the same as minmizing the strongly convex function $Q(x) := \frac{1}{2} x^T A x - b^T x$.*

# IFT vs. inexact AD

**Theorem (Ehrhardt & R. (2022))**

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

$$[\partial_{yy} g(x^{(K)}, \theta)] q = \nabla f(x^{(K)}),$$

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

*Note: if $A$ is SPD, then solving $Ax = b$ is the same as minmizing the strongly convex function $Q(x) := \frac{1}{2} x^T A x - b^T x$.*

So inexact AD is exactly an IFT method in disguise!

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

# Unified Framework

This motivates a general hypergradient approximation framework (based on IFT+CG):

1. Solve the lower-level problem to get $x_\varepsilon^*$ 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 (based on IFT+CG):

1. Solve the lower-level problem to get $x_\varepsilon^*$ 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 (based on IFT+CG):

1. Solve the lower-level problem to get $x_\varepsilon^*$ such that $\|x_\varepsilon^* - x^*\| \le \varepsilon$
2. 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 $h_{\varepsilon,\delta} := -[\partial_y \partial_\theta g(x_\varepsilon^*, \theta)]^T q_{\varepsilon,\delta}$.

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. (2022))**

*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. $\lambda^k$)
- A posteriori: based on measured progress (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. $\lambda^k$)
- A posteriori: based on measured progress (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. $\lambda^k$)
- A posteriori: based on measured progress (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):

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

Best $\lambda$ values (depending on $\alpha$, momentum): $\lambda_{CG} = \lambda_{\mathsf{HB}}^* \ll \lambda_{GD}^*$.

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

**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 $x^*$).

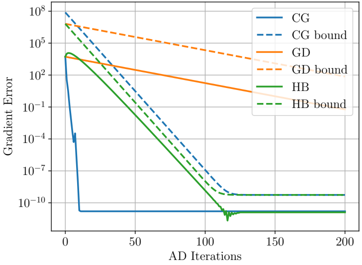
# Outline

# Simple Problem

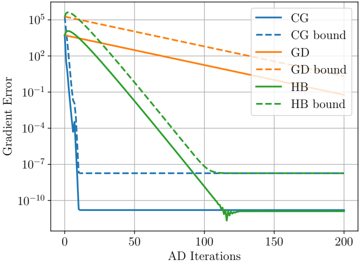
Simple least-squares test problem: [Li et al. (2022)]

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

(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

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

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

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

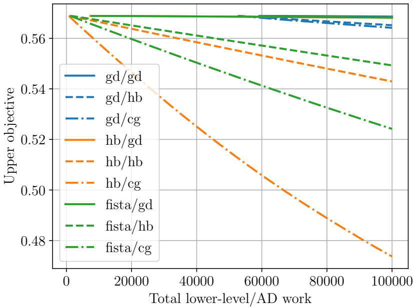
$$\text{s.t. } w^*(\theta) = \arg\min_{w} \frac{1}{N_{\text{train}}} \sum_j \sigma(\theta_j) \cdot \ell(w, x_j^{\text{train}}, y_j^{\text{train}}) + \alpha \|w\|^2.$$

Question: do better hypergradient methods yield better optimization?

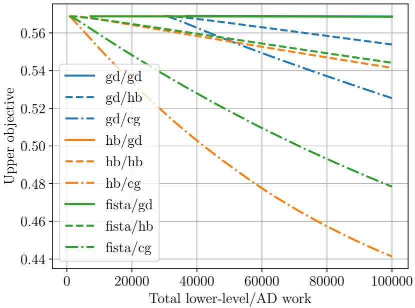
*Work: 1 lower-level iter ≈ 1 AD iter (lower-level gradient vs. Hessian-vector product)*

# 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

# 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.

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.

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.

# References ii


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).