

(Automatic) Iterative Differentiation: some old (& new) results

Samuel Vaiter, CNRS, Université Côte d'Azur

`samuel.vaiter@cnr.fr`

November 6th, 2023

@Séminaire SPO, IHP

Joint works with Jérôme Bolte and Edouard Pauwels

Gradients/Jacobian everywhere

First order information

1. Gradient descent

$$x_{t+1} = x_t - \eta^{(t)} \nabla f(x_t) \quad (\text{GD})$$

f a simple function, a neural network, ...

2. Newton's root finding

$$\text{Jac}_f(x_t)(x_{t+1} - x_t) = -f(x_t) \quad (\text{NEWTON})$$

f a vector field, (potentially itself a first order one)

3. Bilevel optimization (hyperparameter optimization, games)

$$\min_x h(x) = f(y^*(x)) \quad \text{subject to} \quad y^*(x) \in \operatorname{argmin}_y g(x, y)$$

→ Generic hypergradient descent (chain rule)

$$x_{t+1} = x_t - \eta^{(t)} \operatorname{Jac}_{y^*}(x_t)^\top \nabla f(y^*(x_t)) \quad (\text{HGD})$$

4. Implicit model (e.g., Deep Equilibrium Networks, parametric fixed point),

$$f(x^*, y) = x^* \quad (\text{DEQ})$$

How to compute $\nabla f(x)$ (or $\operatorname{Jac}_f(x)$) on a computer?

The “obvious” answer: finite differences

For $\varepsilon > 0$ small enough,

$$f'(x) \approx \frac{f(x + \varepsilon) - f(x)}{\varepsilon} \stackrel{\text{def.}}{=} \Delta_\varepsilon f(x)$$

Proposition $\Delta_\varepsilon f(x)$ has a $O(\varepsilon)$ approximation error of $f'(x)$.

Simple to implement

```
def diff(f, x, eps=1e-5):  
    return (f(x + eps) - f(x)) / eps
```

The “obvious” answer: finite differences

For $\varepsilon > 0$ small enough,

$$f'(x) \approx \frac{f(x + \varepsilon) - f(x)}{\varepsilon} \stackrel{\text{def.}}{=} \Delta_\varepsilon f(x)$$

Proposition $\Delta_\varepsilon f(x)$ has a $O(\varepsilon)$ approximation error of $f'(x)$.

Simple to implement

```
def diff(f, x, eps=1e-5):  
    return (f(x + eps) - f(x)) / eps
```

Are we done yet?

The “obvious” answer: finite differences

For $\varepsilon > 0$ small enough,

$$f'(x) \approx \frac{f(x + \varepsilon) - f(x)}{\varepsilon} \stackrel{\text{def.}}{=} \Delta_\varepsilon f(x)$$

Proposition $\Delta_\varepsilon f(x)$ has a $O(\varepsilon)$ approximation error of $f'(x)$.

Simple to implement

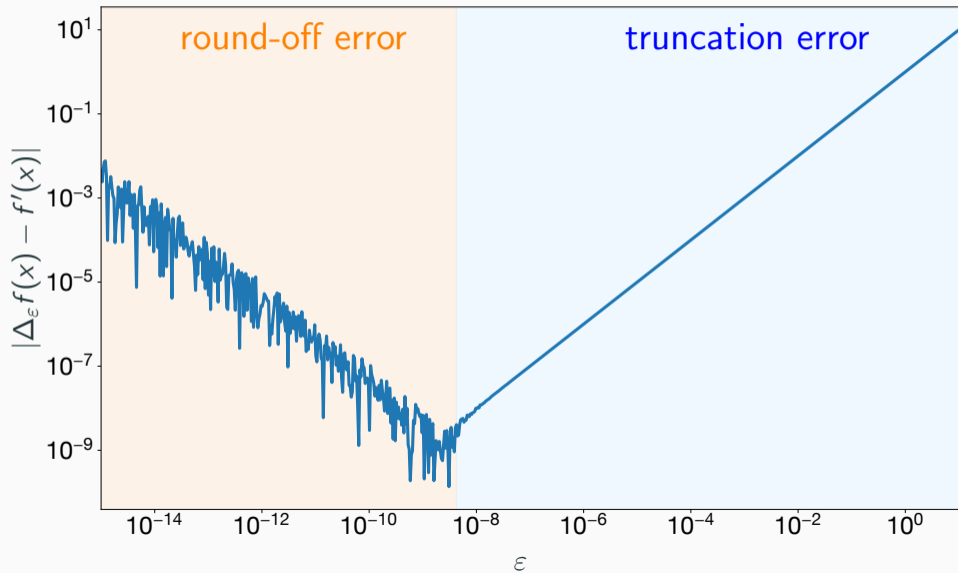
```
def diff(f, x, eps=1e-5):  
    return (f(x + eps) - f(x)) / eps
```

Are we done yet?

Two shortcomings:

1. round-off
2. dimensionality scaling

Shortcoming 1: round-off and truncation



Shortcoming 2: linear complexity w.r.t dimensions

$$f : \mathbb{R}^n \rightarrow \mathbb{R}$$

$$\nabla f(x) \approx \left(\begin{array}{c} \frac{f(x + \Delta h e_1) - f(x)}{\Delta h} \\ \vdots \\ \frac{f(x + \Delta h e_n) - f(x)}{\Delta h} \end{array} \right) \left. \vphantom{\begin{array}{c} \frac{f(x + \Delta h e_1) - f(x)}{\Delta h} \\ \vdots \\ \frac{f(x + \Delta h e_n) - f(x)}{\Delta h} \end{array}} \right\} n \text{ elements}$$


2 calls

Linear cost:



$$\text{Cost}(\nabla f) = 2n \cdot \text{Cost}(f)$$

Automatic Differentiation?



Could you explain me what is automatic differentiation? I heard the term backpropagation in multiple papers but how does it work? 



Automatic differentiation (AD) is a method for computing the derivatives of a function with respect to its input variables. It is often used in machine learning algorithms, particularly in the training of neural networks, where it is known as backpropagation.  

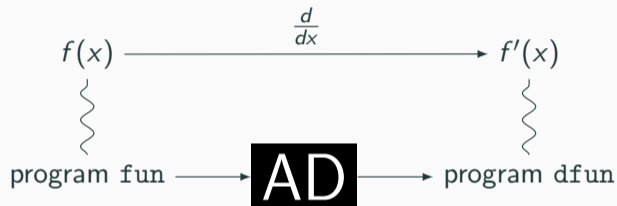
In backpropagation, the error in the predicted output of the neural network is propagated backwards through the network, adjusting the weights of the connections between neurons in order to minimize the error. This is done using the chain rule of calculus, which allows the derivatives of the error with respect to the network's weights to be calculated.

AD algorithms use the structure of the computational graph, which represents the sequence of operations performed by the function, to efficiently compute the derivatives. This allows for the calculation of derivatives even for complex functions with a large number of operations and input variables.

Overall, automatic differentiation is a powerful tool for optimizing machine learning algorithms and allows for efficient training of complex models.

Automatic Differentiation (AD) as a blackbox

Autodiff acts on **computer programs** [Wengert '64, Linnainmaa '70]



[Elliot 2018]: Bijection between composition of **augmented** programs and composition of **differentiable** functions

AD is “just” the chain rule (**it's not**)

Constant complexity of AD

$$f : \mathbb{R}^p \rightarrow \mathbb{R}$$

Finite difference cost

$$\text{Cost}(\nabla f) = 2p \cdot \text{Cost}(f) \quad \text{linear in } p$$

Automatic differentiation cost [Baur & Strassen '83]

$$\text{Cost}(\nabla f) \leq 5 \cdot \text{Cost}(f) \quad \text{constant in } p$$

Computational graph (feed-forward)

Composition of functions ($f_0 : \mathbb{R}^P = \mathbb{R}^{n_0} \rightarrow \mathbb{R}^{n_1}, \dots, f_3 : \mathbb{R}^{n_3} \rightarrow \mathbb{R}^{n_4} = \mathbb{R}^n$)

$$f = f_0 \circ f_1 \circ f_2 \circ f_3 : \mathbb{R}^P \rightarrow \mathbb{R}^n$$

Computational graph



Chain rule

$$\frac{\partial y}{\partial x} = \frac{\partial w_1}{\partial w_0} \frac{\partial w_2}{\partial w_1} \frac{\partial w_3}{\partial w_2} \frac{\partial w_4}{\partial w_3} = \text{Jac}_{f_3}(w_3) \text{Jac}_{f_2}(w_2) \text{Jac}_{f_1}(w_1) \text{Jac}_{f_0}(w_0)$$

Computational graph (feed-forward)

Composition of functions ($f_0 : \mathbb{R}^P = \mathbb{R}^{n_0} \rightarrow \mathbb{R}^{n_1}, \dots, f_3 : \mathbb{R}^{n_3} \rightarrow \mathbb{R}^{n_4} = \mathbb{R}^n$)

$$f = f_0 \circ f_1 \circ f_2 \circ f_3 : \mathbb{R}^P \rightarrow \mathbb{R}^n$$

Computational graph

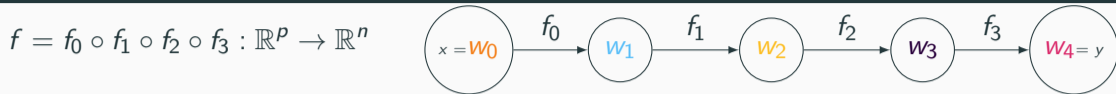


Chain rule

$$\frac{\partial y}{\partial x} = \frac{\partial w_1}{\partial w_0} \frac{\partial w_2}{\partial w_1} \frac{\partial w_3}{\partial w_2} \frac{\partial w_4}{\partial w_3} = \text{Jac}_{f_3}(w_3) \text{Jac}_{f_2}(w_2) \text{Jac}_{f_1}(w_1) \text{Jac}_{f_0}(w_0)$$

How to compute this product?

Forward mode



Jacobian-vector products (JVPs)

$$\text{Jac}_f(x) = \left(\text{Jac}_f(x)\mathbf{e}_1 \quad \cdots \quad \text{Jac}_f(x)\mathbf{e}_p \right) \implies \text{need } p \text{ JVPs (column-per-column)}$$

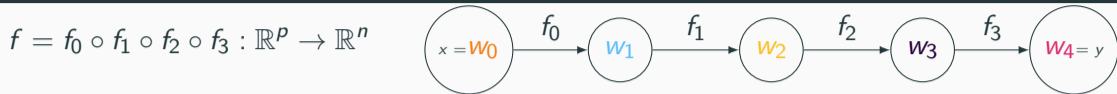
Chain rule for JVPs

$$\text{Jac}_f(x)\mathbf{e}_k = \underbrace{\text{Jac}_{f_3}(w_3) [\text{Jac}_{f_2}(w_2) \{ \text{Jac}_{f_1}(w_1) (\text{Jac}_{f_0}(w_0)\mathbf{e}_k) \}]}_{\text{right-to-left multiplication (forward } w_0 \rightarrow w_1 \rightarrow w_2 \rightarrow w_3 \rightarrow w_4)}$$

Cost of p JVPs: $p \sum_{i=0}^3 n_i n_{i+1}$

$O(p^3)$ if $n_i = p$ for $i \neq 4$

Reverse mode (i.e., Backpropagation)



Vector-Jacobian products (VJPs)

$$\text{Jac}_f(x) = \left(\mathbf{e}_1^\top \text{Jac}_f(x) \quad \cdots \quad \mathbf{e}_n^\top \text{Jac}_f(x) \right)^\top \implies \text{need } n \text{ VJPs (row-by-row)}$$

Chain rule for VJPs

$$E_l \text{Jac}_f(x) = \underbrace{\left[\left\{ \left(E_l \text{Jac}_{f_3}(w_3) \right) \text{Jac}_{f_2}(w_2) \right\} \text{Jac}_{f_1}(w_1) \right]}_{\text{left-to-right multiplication (reverse } w_4 \rightarrow w_3 \rightarrow w_2 \rightarrow w_1 \rightarrow w_0)} \text{Jac}_{f_0}(w_0)$$

Cost of n VJPs: $n \sum_{i=0}^3 n_i n_{i+1}$

$O(p^2)$ if $n_i = p$ for $i \neq 4$

This talk

Trying to make sense of

$\frac{\partial}{\partial \alpha}$

```
def forward_backward(alpha, prox_f, grad_g, x0, max_iter):  
    x = x0  
    for _ in range(max_iter):  
        x = prox_f(x - alpha*grad_g(x), alpha)  
    return x
```

and what happens when

$\text{max_iter} \rightarrow +\infty$

Iterative differentiation

Fixed point operator $F : \mathbb{R}^n \times \mathbb{R}^p \rightarrow \mathbb{R}^n$

Parametric iterative algorithm

$$\begin{cases} x^{(0)}(\theta) & \in \mathbb{R}^n \\ x_{t+1}(\theta) & = F(x_t(\theta), \theta) \end{cases}$$

Theorem (Fixed-point)

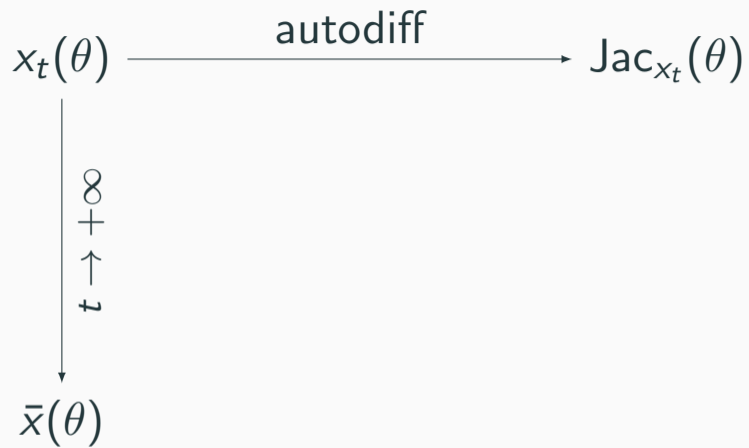
If for all θ , $x \mapsto F(x, \theta)$ is ρ -Lipschitz with $\rho < 1$,

$$\lim_{t \rightarrow +\infty} x_t(\theta) = \text{Fix } F(\cdot, \theta) = \bar{x}(\theta).$$

$$x_t(\theta)$$

$$\begin{array}{c} x_t(\theta) \\ \downarrow \\ \bar{x}(\theta) \end{array}$$

δ
+
 \uparrow
 τ





Smooth iterations: $F \in \mathcal{C}^2$ and $x_{t+1}(\theta) = F(x_t(\theta), \theta)$.

Piggyback recursion (total derivative)

$$\text{Jac}_{x_{t+1}}(\theta) = \partial_1 F(x_t(\theta), \theta) \text{Jac}_{x_t}(\theta) + \partial_2 F(x_t(\theta), \theta)$$

Theorem ([Gilbert '92, Prop. 1])

If F is ρ -Lipschitz ($\rho < 1$) and $\sigma_{\max}(\partial_1 F) < 1$, then

$$\lim_{t \rightarrow +\infty} \text{Jac}_{x_t}(\theta) = \text{Jac}_{\bar{x}}(\theta) = \text{Jac}_{\lim_{t \rightarrow +\infty} x_t}(\theta)$$

Smooth iterations: $F \in \mathcal{C}^2$ and $x_{t+1}(\theta) = F(x_t(\theta), \theta)$.

Piggyback recursion (total derivative)

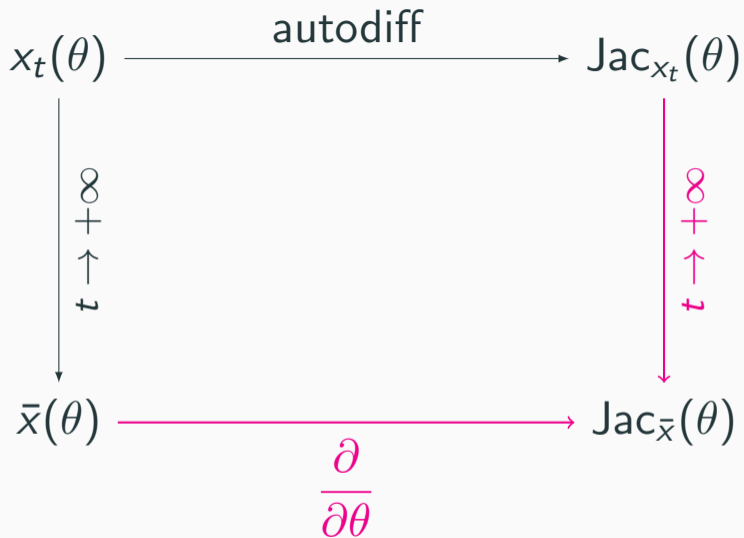
$$\text{Jac}_{x_{t+1}}(\theta) = \partial_1 F(x_t(\theta), \theta) \text{Jac}_{x_t}(\theta) + \partial_2 F(x_t(\theta), \theta)$$

Theorem ([Gilbert '92, Prop. 1])

If F is ρ -Lipschitz ($\rho < 1$) and $\sigma_{\max}(\partial_1 F) < 1$, then

$$\lim_{t \rightarrow +\infty} \text{Jac}_{x_t}(\theta) = \text{Jac}_{\bar{x}}(\theta) = \text{Jac}_{\lim_{t \rightarrow +\infty} x_t}(\theta)$$

Condition $\sigma_{\max}(\partial_1 F) < 1$ is sufficient but **not necessary**



Implicit differentiation

Fixed-point equation

$$x_{t+1}(\theta) = F(x_t(\theta), \theta) \implies \bar{x}(\theta) = F(\bar{x}(\theta), \theta)$$

Differentiation of fixed-point equation

$$\text{Jac}_{\bar{x}}(\theta) = \partial_1 F(\bar{x}(\theta), \theta) \text{Jac}_{\bar{x}}(\theta) + \partial_2 F(x_t(\theta), \theta)$$

Implicit function theorem (assuming **invertibility** of $\text{Id} - \partial_2$)

$$\text{Jac}_{\bar{x}}(\theta) = (\text{Id} - \partial_2 F(x(\theta), \theta))^{-1} \partial_1 F(x(\theta), \theta)$$

Proposition (Autodiff \equiv implicit differentiation)

If F is ρ -Lipschitz ($\rho < 1$) and $\sigma_{\max}(\partial_1 F) < 1$, then

$$\lim_{t \rightarrow +\infty} \text{Jac}_{x_t}(\theta) = \text{Jac}_{\lim_{t \rightarrow +\infty} x_t}(\theta) = (\text{Id} - \partial_2 F(x(\theta), \theta))^{-1} \partial_1 F(x(\theta), \theta)$$

Parametric entropic optimal transport

$$\hat{P}(\theta) = \operatorname{argmin}_{P \in U(\theta)} \langle P, C(\theta) \rangle - \varepsilon(\theta) \operatorname{Ent}(P) \quad \text{where} \quad U(\theta) = \left\{ P \in \mathbb{R}_{\geq 0}^{n \times m} : \begin{array}{l} P \mathbf{1}_m = a(\theta) \\ P^\top \mathbf{1}_n = b(\theta) \end{array} \right\} \quad (\text{OT}_\theta)$$

Sinkhorn algorithm \equiv matrix scaling

$$u_{t+1}(\theta) = \frac{a(\theta)}{K(\theta)v_t(\theta)} \quad \text{and} \quad v_{t+1}(\theta) = \frac{b(\theta)}{K(\theta)^\top u_{t+1}(\theta)}, \quad (\text{SK}_\theta)$$

Current optimal plan at time t

$$P_t(\theta) = \operatorname{diag}(u_t(\theta)) K(\theta) \operatorname{diag}(v_t(\theta)) \quad \text{where} \quad K_{i,j}(\theta) = \exp\left(-\frac{C_{i,j}(\theta)}{\varepsilon(\theta)}\right) > 0,$$

Sinkhorn as a fixed point algorithm

$$\hat{P}(\theta) = \operatorname{argmin}_{P \in U(\theta)} \mathcal{L}(P, \theta) \quad \text{and} \quad \begin{array}{ll} x_t(\theta) = F(x_t(\theta), \theta) & \text{reduced "log" variable} \\ P_t(\theta) = P(x_t(\theta), \theta) & \text{transport plan} \end{array}$$

$$F(x, \theta) = \log a(\theta) - \log \left(K(\theta) \frac{b(\theta)}{K^\top(\theta) e^x} \right) \quad \text{and} \quad P(x, \theta) = e^x K(\theta) \operatorname{diag} \left(\frac{b(\theta)}{K^\top(\theta) e^x} \right)$$

Theorem ([Franklin & Lorenz '89])

$P_t(\theta)$ converges (linearly) towards $\hat{P}(\theta)$.

Fact

For all t , $\theta \mapsto P_t(\theta)$ is differentiable.

BUT

$$\sigma_{\max}(\partial_1 F(x, \theta)) = 1$$

\implies can't apply directly [Gilbert '92]

Convergence of the derivatives of Sinkhorn

$$\hat{P}(\theta) = \operatorname{argmin}_{P \in U(\theta)} \mathcal{L}(P, \theta) \quad \text{and} \quad \begin{array}{ll} x_t(\theta) = F(x_t(\theta), \theta) & \text{reduced "log" variable} \\ P_t(\theta) = P(x_t(\theta), \theta) & \text{transport plan} \end{array}$$

Theorem ([Pauwels-V. 2022])

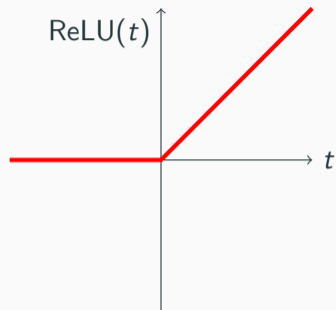
Assume that a, b, C, ε are C^2 . Then,

- The optimal coupling \hat{P} is C^1 (with an implicit equation linked to \bar{x})
- The AD iterates P_t is C^1 and $\frac{dP_t}{d\theta}$ converges at a linear rate, locally uniformly in θ .
- We have

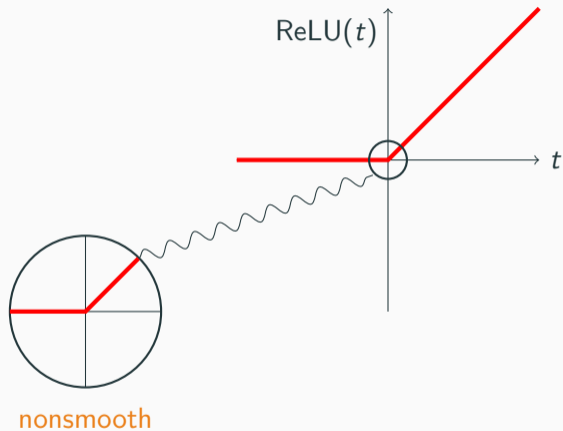
$$\lim_{t \rightarrow +\infty} \frac{dP_t}{d\theta}(\theta) = \frac{d\hat{P}}{d\theta}(\theta)$$

Main intuition: x_t is equivariant w.r.t $u = \exp(x)$, and so does \hat{P} .

Elephant in the room



Elephant in the room: nonsmoothness





$x_t(\theta)$ autodiff \rightarrow $\text{Jac}_{x_t}(\theta)$

∞
+
 \uparrow
 t

```
> from jax.nn import relu
> from jax import grad
> grad(relu)(0.0)
0.0
> grad(lambda x: relu(-x) + x)(0.0)
1.0
> grad(lambda x:
>     derivative(-x) + x - relu(x))(0.0)
1.0
```

limit?

????

Conservative gradient \equiv Subdifferential compatible with the chain rule

Definition (Conservative gradient [Bolte & Pauwels 2020])

$f : \mathbb{R}^n \rightarrow \mathbb{R}$ locally Lipschitz. $J : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$, closed, nonempty, locally bounded. f is path-differentiable with conservative gradient J if for any Lipschitz curve $\gamma : [0, 1] \rightarrow \mathbb{R}^n$, for all $v \in J(\gamma(t))$

$$\frac{d}{dt}f(\gamma(t)) = \langle v, \dot{\gamma}(t) \rangle \quad \text{a.e. } t \in [0, 1]$$

chain rule
along Lipschitz curve

Proposition

1. The Clarke subdifferential is *minimal* for conservative gradient: $\partial^c f(x) \subseteq J(x)$
2. The conservative gradient is reduced to $J(x) = \{\nabla f(x)\}$ a.e. (Radamacher)
3. Path-differentiability is compatible with sum and *composition*

AD of nonsmooth iterative programs

Nonsmooth iterations: F path-differentiable and $x_{t+1}(\theta) = F(x_t(\theta), \theta)$.

Here, nonsmooth **does not** mean that we minimize nonsmooth functions!

Smooth minimization problem, $h \in C^{1,1}$

$$\min_{x \in \mathbb{R}^n} h(x)$$

Nonsmooth algorithm

$$\begin{aligned}x_{t+1} &= x_t - \rho \nabla h(x_t) \\ &= F(x_t, \rho)\end{aligned}$$

where

$$F(x, \rho) = x - \rho \nabla h(x)$$

Smooth minimization pb, $h \in C^2$

$$\min_{x \in \mathbb{R}^n} h(x)$$

Smooth algorithm

$$\begin{aligned}x_{t+1} &= x_t - \rho \nabla h(x_t) \\ &= F(x_t, \rho)\end{aligned}$$

where

$$F(x, \rho) = x - \rho \nabla h(x)$$

AD of nonsmooth iterative programs

Nonsmooth iterations: F path-differentiable and $x_{t+1}(\theta) = F(x_t(\theta), \theta)$.

Set-valued piggyback recursion

$$J_{x_{t+1}}(\theta) = \{AJ + B : [A, B] \in J_F(x_t(\theta), \theta), J \in J_{x_t}(\theta)\}$$

AD frameworks gives an element $J_{t+1} = A_t J_t + B_t$ where $[A_t, B_t] \in J_F(x_t(\theta), \theta)$

Intuition: $[A, B]$ is the “nonsmooth total derivative”

$$A \cong \partial_1 F(x_t(\theta), \theta) \quad \text{and} \quad B \cong \partial_2 F(x_t(\theta), \theta)$$



Main result: limit-derivative exchange

Assumption A: J_F contracts w.r.t. x .

There exists $0 \leq \rho < 1$, such that for any (x, θ) and any conservative gradient $[A, B] \in J_F(x, \theta)$ (or at least locally),

$$\sigma_{\max}(A) < \rho,$$

Limit candidate

$$\begin{aligned} J_{\bar{x}}^{\text{pb}} : \theta &\rightrightarrows \text{fix} [J_F(\bar{x}(\theta), \theta)] \\ &= \text{fix} [J_F(\text{fix}(F(\cdot, \theta)), \theta)] \end{aligned}$$

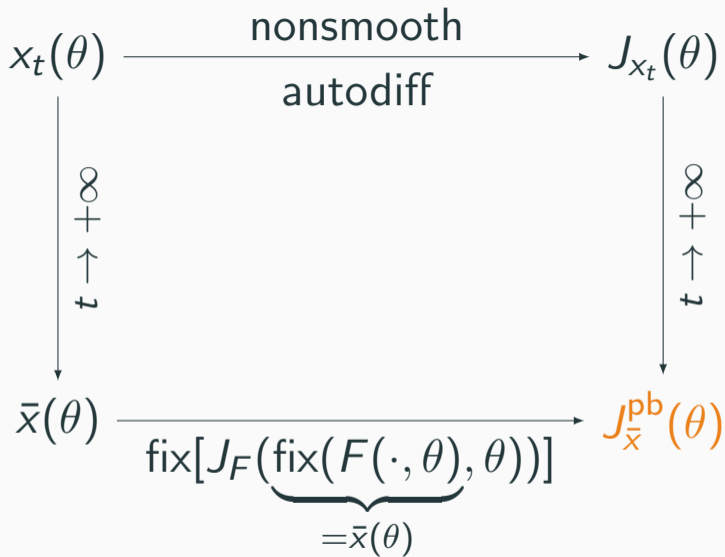
Gap between compact subsets

$$\text{gap}(\mathcal{X}, \mathcal{Y}) = \max_{x \in \mathcal{X}} d(x, \mathcal{Y}) \quad \text{where} \quad d(x, \mathcal{Y}) = \min_{y \in \mathcal{Y}} \|x - y\|,$$

Theorem ([Bolte–Pauwels–V. 2022])

Under Assumption **A**, $J_{\bar{x}}^{\text{pb}}$ is a conservative Jacobian for \bar{x} and

$$\lim_{t \rightarrow \infty} \text{gap}(J_{x_t}(\theta), J_{\bar{x}}^{\text{pb}}(\theta)) = 0.$$



Main result: limit-derivative exchange

Assumption A: J_F contracts w.r.t. x .

There exists $0 \leq \rho < 1$, such that for any (x, θ) and any conservative gradient $[A, B] \in J_F(x, \theta)$ (or at least locally),

$$\sigma_{\max}(A) < \rho,$$

Limit candidate

$$\begin{aligned} J_{\bar{x}}^{\text{pb}} : \theta &\rightrightarrows \text{fix} [J_F(\bar{x}(\theta), \theta)] \\ &= \text{fix} [J_F(\text{fix}(F(\cdot, \theta)), \theta)] \end{aligned}$$

Gap between compact subsets

$$\text{gap}(\mathcal{X}, \mathcal{Y}) = \max_{x \in \mathcal{X}} d(x, \mathcal{Y}) \quad \text{where} \quad d(x, \mathcal{Y}) = \min_{y \in \mathcal{Y}} \|x - y\|,$$

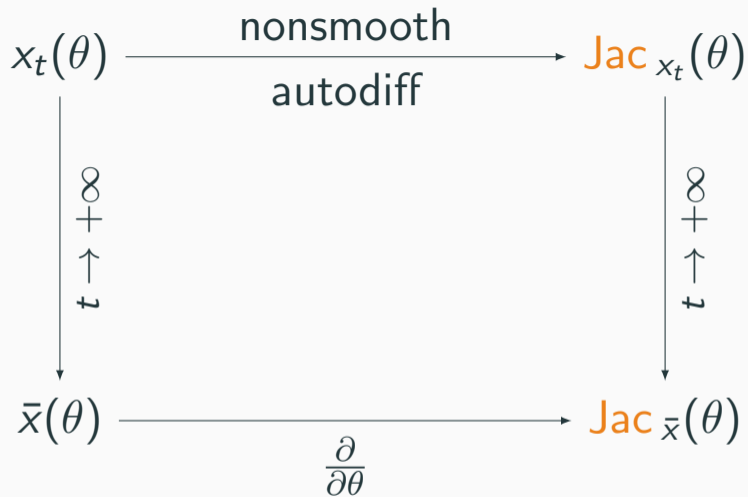
Theorem ([Bolte–Pauwels–V. 2022])

Under Assumption **A**, $J_{\bar{x}}^{\text{pb}}$ is a conservative Jacobian for \bar{x} and

$$\lim_{t \rightarrow \infty} \text{gap}(J_{x_t}(\theta), J_{\bar{x}}^{\text{pb}}(\theta)) = 0.$$

Moreover, for almost all θ ,

$$\lim_{k \rightarrow \infty} \text{Jac}_{x_t}(\theta) = \text{Jac}_{\bar{x}}(\theta).$$



almost everywhere

Main result: quantitative version

Assumption A: J_F contracts w.r.t. x .

There exists $0 \leq \rho < 1$, such that for any (x, θ) and any conservative gradient $[A, B] \in J_F(x, \theta)$ (or at least locally),

$$\sigma_{\max}(A) < \rho,$$

Limit candidate

$$\begin{aligned} J_{\bar{x}}^{\text{pb}} : \theta &\rightrightarrows \text{fix} [J_F(\bar{x}(\theta), \theta)] \\ &= \text{fix} [J_F(\text{fix}(F(\cdot, \theta)), \theta)] \end{aligned}$$

Gap between compact subsets

$$\text{gap}(\mathcal{X}, \mathcal{Y}) = \max_{x \in \mathcal{X}} d(x, \mathcal{Y}) \quad \text{where} \quad d(x, \mathcal{Y}) = \min_{y \in \mathcal{Y}} \|x - y\|,$$

Theorem ([Bolte–Pauwels–V. 2022])

Under Assumption **A**, $J_{\bar{x}}^{\text{pb}}$ is a conservative Jacobian for \bar{x} and

$$\lim_{t \rightarrow \infty} \text{gap}(J_{x_t}(\theta), J_{\bar{x}}^{\text{pb}}(\theta)) = 0.$$

If moreover, F has a **Lipschitz gradient selection** (\approx piecewise semialgebraic), then the convergence is **linear**

$$\text{gap}(J_{x_t}(\theta), J_{\bar{x}}^{\text{pb}}(\theta)) \lesssim \rho^{t/2}.$$

Nonsmooth implicit differentiation

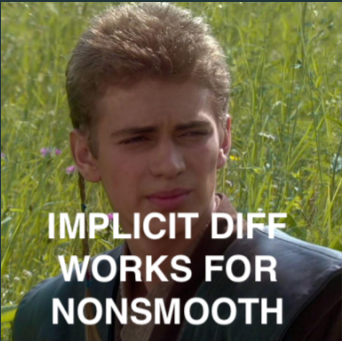
Fixed-point equation

$$x_{t+1}(\theta) = F(x_t(\theta), \theta) \implies \bar{x}(\theta) = F(\bar{x}(\theta), \theta)$$

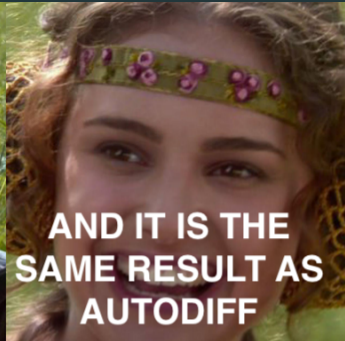
Implicit function theorem (qualification-free) [Bolte et al. 2021] Assuming that $I - A$ is invertible for all $[A, B] \in J_F(\bar{x}(\theta), \theta)$

$$J_{\bar{x}}^{\text{imp}} : \theta \rightrightarrows \{(I - A)^{-1}B : [A, B] \in J_F(\bar{x}(\theta), \theta)\}$$

is a conservative Jacobian for \bar{x} .



**IMPLICIT DIFF
WORKS FOR
NONSMOOTH**



**AND IT IS THE
SAME RESULT AS
AUTODIFF**



**AND IT IS THE
SAME RESULT AS
AUTODIFF,RIGHT?**

Nonsmooth implicit differentiation

Fixed-point equation

$$x_{t+1}(\theta) = F(x_t(\theta), \theta) \implies \bar{x}(\theta) = F(\bar{x}(\theta), \theta)$$

Implicit function theorem (qualification-free) [Bolte et al. 2021] Assuming that $I - A$ is invertible for all $[A, B] \in J_F(\bar{x}(\theta), \theta)$

$$J_{\bar{x}}^{\text{imp}} : \theta \rightrightarrows \{(I - A)^{-1}B : [A, B] \in J_F(\bar{x}(\theta), \theta)\}$$

is a conservative Jacobian for \bar{x} .

In general,

$$J_{\bar{x}}^{\text{imp}} \subsetneq J_{\bar{x}}^{\text{pb}}.$$

Implicit differentiation may fail to capture the output of autodiff

Consequences for VJPs and JVPs

Forward mode (JVP):

$$\dot{x}_0 = J\dot{\theta}, J \in J_{x_0}(\theta).$$

for $i = 1, \dots, t$ **do**

$$x_i = F(x_{i-1}, \theta)$$

$$\dot{x}_i = A_{i-1}\dot{x}_{i-1} + B_{i-1}\dot{\theta}$$

$$[A_{i-1}, B_{i-1}] \in J_F(x_{i-1}, \theta)$$

Return: \dot{x}_t

Reverse mode (VJP): $\bar{\theta}_t = 0.$

for $i = 1, \dots, t$ **do**

$$x_i = F(x_{i-1}, \theta)$$

for $i = t, \dots, 1$ **do**

$$\bar{\theta}_t = \bar{\theta}_t + B_{i-1}^T \bar{w}_i \quad \bar{w}_{i-1} = A_{i-1}^T \bar{w}_i$$

$$[A_{i-1}, B_{i-1}] \in J_F(x_{i-1}, \theta)$$

$$\bar{\theta}_t = \bar{\theta}_t + J^T \bar{w}_0, J \in J_{x_0}(\theta)$$

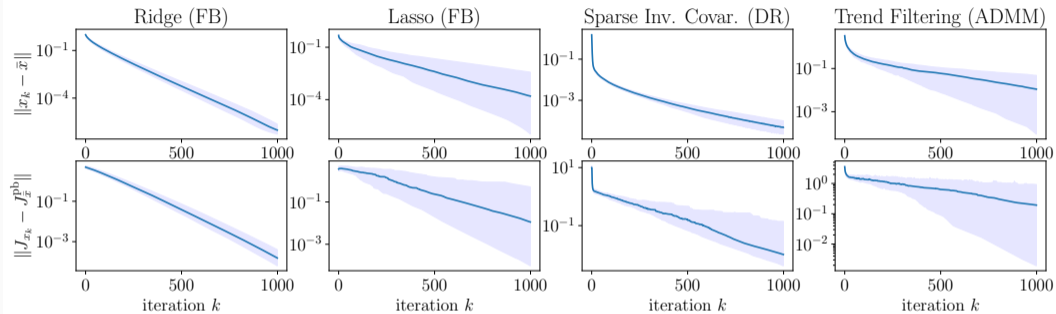
Return: $\bar{\theta}_t$

Proposition

Under Assumption **A**, for almost all $\theta \in \mathbb{R}^m$, $\dot{x}_t \rightarrow \frac{\partial \bar{x}}{\partial \theta} \dot{\theta}$.

Assume furthermore that, as $t \rightarrow \infty$, $\bar{w}_t \rightarrow \bar{w}$ (for example, $\bar{w}_t = \nabla \ell(x_t)$ for a C^1 loss ℓ), then for almost all $\theta \in \mathbb{R}^m$, $\bar{\theta}_t^T \rightarrow \bar{w}^T \frac{\partial \bar{x}}{\partial \theta}$.

Empirical observations



- FB: Forward–Backward
- DR: Douglas–Rachford
- ADMM: Alternating Direction Method of Multipliers

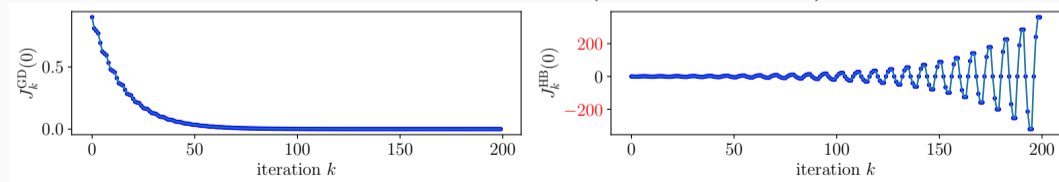
When nonsmoothness hurts: momentum methods

Heavy-ball method [Polyak '87]

$$(x_{t+1}, y_{t+1}) = F(x_t, y_t, \theta) \quad \text{where} \quad F(x, y, \theta) = x - \alpha \nabla f_\theta(x) + \beta(x - y)$$

If (x_t) converges and f_θ is C^2 , then Jac_{x_t} converges [Griewank & Faure 2003, Mehmood & Ochs 2020]

But the situation is different for $C^{1,1}$ functions (left GD, right HB) (example)



Conclusion

AD is a convergent process (under strong assumptions) even in the nonsmooth case.

Open problems

- How to handle in general AD of F such that $\sigma_{\max}(\partial_1 F) = 1$?
- How to handle stochastic algorithms $F(x, \theta; \xi)$, $\xi \sim \mu$?

Related papers:

- Bolte, Pauwels, V. *Automatic differentiation of nonsmooth iterative algorithms*. NeurIPS. 2022
- Pauwels, V. *The derivatives of Sinkhorn-Knopp converge*. SIAM J. Opt. 2023
- Bolte, Pauwels, V. *One-step differentiation of iterative algorithms*. NeurIPS. 2023

We are looking for a M2 intern to extend these results to stochastic settings!

Behind the scene: fixed-point theorem for set-valued mapping

Theorem ([Bolte-Pauwels-V 2022])

Let $\mathcal{J} \subset \mathbb{R}^{p \times (p+m)}$ be a compact subset of matrices with $\rho < 1$. Then there is a unique nonempty compact set $\text{fix}(\mathcal{J}) \subset \mathbb{R}^{p \times m}$ satisfying $\text{fix}(\mathcal{J}) = \mathcal{J}(\text{fix}(\mathcal{J}))$, where the action of \mathcal{J} is given by

$$\mathcal{J}(\mathcal{X}) = \{AX + B, [A, B] \in \mathcal{J}, X \in \mathcal{X}\}.$$

Let $(\mathcal{X}_t)_{t \in \mathbb{N}}$ be a sequence of compact subsets of $\mathbb{R}^{p \times m}$, such that $\mathcal{X}_0 \neq \emptyset$, and satisfying the recursion

$$\mathcal{X}_{t+1} = \mathcal{J}(\mathcal{X}_t) \quad \forall t \in \mathbb{N}.$$

We have for all $t \in \mathbb{N}$

$$\text{dist}(\mathcal{X}_t, \text{fix}(\mathcal{J})) \leq \rho^t \frac{\text{dist}(\mathcal{X}_0, \mathcal{J}(\mathcal{X}_0))}{1 - \rho},$$

Lipschitz gradient selection

Definition ([Bolte–Pauwels 2020])

Let $F: \mathbb{R}^p \mapsto \mathbb{R}^q$ be semialgebraic and continuous. We say that F has a **Lipschitz gradient selection** (s, F_1, \dots, F_m) if $s: \mathbb{R}^p \mapsto \{1, \dots, m\}$ is semialgebraic and there exists $L \geq 0$ such that for $i = 1, \dots, m$, $F_i: \mathbb{R}^p \mapsto \mathbb{R}^q$ is semialgebraic with L -Lipschitz Jacobian, and for all $x \in \mathbb{R}^p$, $F(x) = F_{s(x)}(x)$.

For any $x \in \mathbb{R}^p$, set $I(x) = \{i \in \{1, \dots, m\}, F(x) = F_i(x)\}$. The set-valued map $J_F^s: \mathbb{R}^p \rightrightarrows \mathbb{R}^{p \times q}$ given by $J_F^s: x \rightrightarrows \text{conv} \left(\left\{ \frac{\partial F_i}{\partial x}(x), i \in I(x) \right\} \right)$ is a conservative Jacobian for F . Here $\frac{\partial F_i}{\partial x}$ denotes the classical Jacobian of F_i .

Example where heavy-ball fails

Piecewise quadratic independent from θ

$$f(x, \theta) = \begin{cases} x^2/2 & \text{if } x \geq 0 \\ x^2/8 & \text{if } x < 0. \end{cases}$$

Heavy-ball

$$F(x, y, \theta) = x - \alpha \nabla f_{\theta}(x) + \beta(x - y) \quad \text{where } \alpha = 1 \text{ and } \beta = \frac{3}{4}$$

Proposition

HB converges globally to 0 and ∇F is path differentiable.

$$J_F(0, 0, 0) = \text{conv} \{M_1, M_2\},$$

where the product $M_1 M_1 M_2 M_2$ has eigenvalue $-9/8$.