On the complexity of nonsmooth automatic differentiation

Jérôme Bolte , Ryan Boustany , Edouard Pauwels and Béatrice Pesquet-Popescu

ICLR 2023 - The Eleventh International Conference on Learning Representations

$$\min_{\theta \in \mathbb{R}^P} \quad J(\theta) := \frac{1}{N} \sum_{i=1}^{N} \ell(f(x_i, \theta), y_i)$$

- $(x_i, y_i)_{i=1}^N$: training set
- f: can be composed of nonsmooth functions (e.g., ReLU, MaxPooling)
- $\theta \in \mathbb{R}^P$: weight parameters
- ℓ: loss function

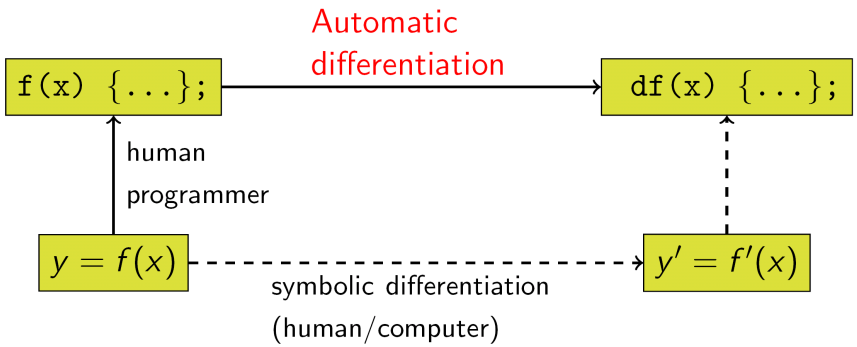


Figure: Illustration of how AD works

Motivation

Two assumptions for training nonsmooth DNNs:

1. Backpropagation outputs a gradient almost everywhere (theorem)
2. The process is fast (empirical observation)

$f : \mathbb{R}^p \to \mathbb{R}$ differentiable function.

P: program computing f.

$\mathrm{backprop}(P)$: program computing $(f, \nabla f)$ using $\mathrm{backprop}$ AD.

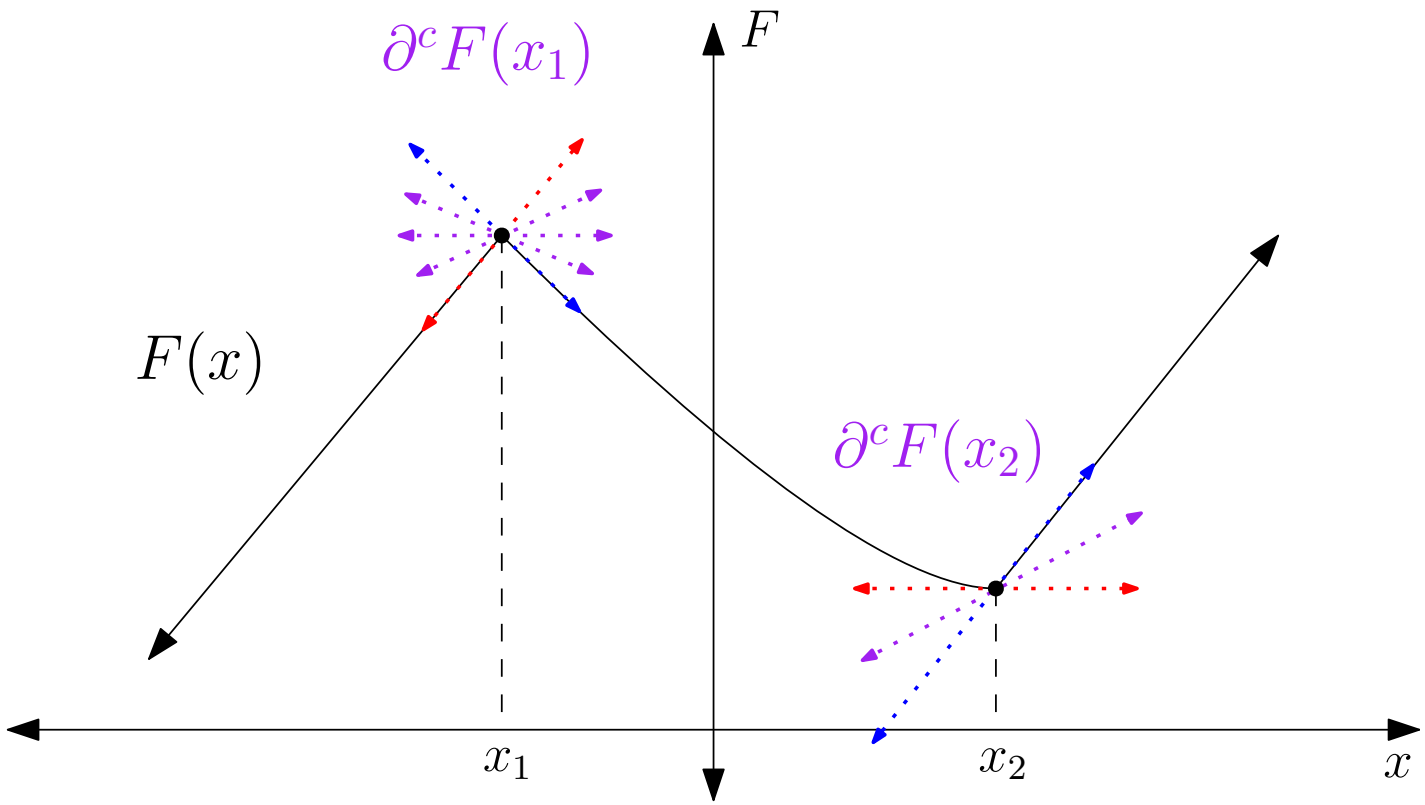
$\mathrm{cost}(\cdot)$: execution time to evaluate a program.

Theorem (Baur and Strassen, 1983)

For rational functions f: $\mathrm{cost}(\mathrm{backprop}(P)) \leqslant 5 \times \mathrm{cost}(P)$

Motivation: generalize to nonsmooth functions.

Locally Lipschitz Functions and the Clarke Subgradient



$$\partial^c F(x) = \text{conv}\left\{ \lim_{k \to +\infty} \nabla F(x_k) : x_k \in \text{diff}_F, x_k \underset{k \to +\infty}{\to} x \right\}$$

How does nonsmooth AD algorithm works ?

Nonsmooth AD

$F : \mathbb{R}^p \to \mathbb{R}$ locally Lipschitz function in compositional form

$$\boxed{F = g_1 \circ \ldots \circ g_m}.$$

- $\boxed{d_i(x) = \nabla g_i(x)}$ for smooth g_i
- $\boxed{d_i(x) \in \partial^c g_i(x)}$ (when you hit a nonsmooth part)
- Ex : $g_i = \mathrm{relu}$ and take $d_i(0) = \mathrm{relu}'(0) = 0$ (Tensorflow, Pytorch)
- $\mathrm{backprop}(P)$: chain rule the d_i's.

Artifacts

- $\mathrm{backprop}\ g_1(W) + \ldots + \mathrm{backprop}\ g_m(W) \notin \partial^c(g_1 + \ldots + g_m)(W)$
- $\mathrm{relu}_2 : t \to \mathrm{relu}(-t) + t$ and $\mathrm{relu}'(0) = 0$
- $zero = \mathrm{relu}_2 - \mathrm{relu}$ and $zero'(0) = 1$ (?!)

Conservative gradients (Bolte and Pauwels 2019)

Main properties of conservative gradients

Let $D_f : \mathbb{R}^p \rightrightarrows \mathbb{R}^p$ be a conservative gradient for $f : \mathbb{R}^p \to \mathbb{R}$ locally Lipschitz.

- Conservative gradients $=$ gradients a.e
- For all $x \in \mathbb{R}^p$, $\partial^c F(x) \subset \mathrm{conv}(D_F(x))$.
- Compatible with calculus rules contrary to Clarke subdifferential.

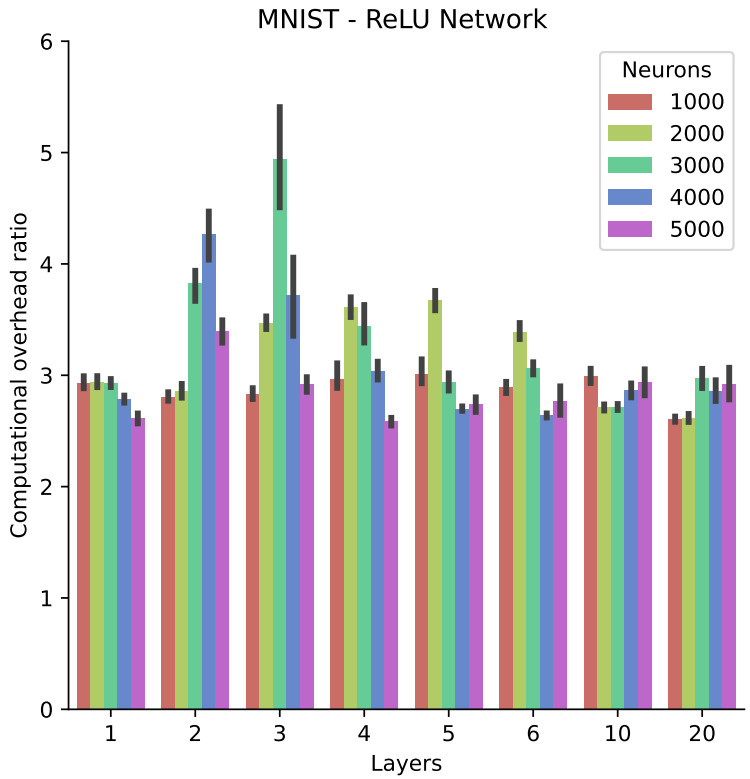
- Most common (virtually all, semialgebraic) functions using in DL admits conservative gradients.
- Faithfully model what is computed by backprop (generated by Pytorch).
- Preserving convergence guaranties.

Nonsmooth Cheap Gradient Principle

Theorem (Nonsmooth Cheap Gradient Principle, (Bolte et al. 2022))

Let P be a program that computes $F = g_1 \circ \ldots \circ g_m$ locally Lipschitz.

1. $\mathrm{backprop}(P)$ returns an element of a conservative gradient.
2. $\mathrm{cost}(\mathrm{backprop}(P)) \leqslant \boxed{\omega_b} \times \mathrm{cost}(P)$, where ω_b is a constant.
3. $\boxed{\omega_b \approx 5}$ for ReLU networks.



Comparison of conservative gradients with other nonsmooth AD

Alternative AD methods

For a locally Lipschitz function $F : \mathbb{R}^p \mapsto \mathbb{R}$:

- $\omega := \text{cost}(p \times p \text{ matrix multiplication}) \approx p^{2.7}$ (best algorithm)
- $\text{cost}(p \text{ directional derivatives of } F)/\text{cost}(F) \geqslant p^{\omega-2}$

Computational complexity comparison

F relu network with matrix and vector entries in $\{$-1,0,1$\}$.

1. Computing two distinct elements of $\partial^c F$ is NP-hard.
2. Computing two elements of conservative gradients is polynomial time solvable.

Conclusion

- Generalized subgradients cannot capture backpropagation (conservatives do).

- We extend the Baur-Strassen theorem (rational functions) to semi-algebraic functions (ubiquitous in ML) with a specific arithmetic model.

- We prove worst case lower bound in the nonsmooth context for directional derivatives and subgradient enumeration.

https://arxiv.org/abs/2206.01730

Thanks for your attention.