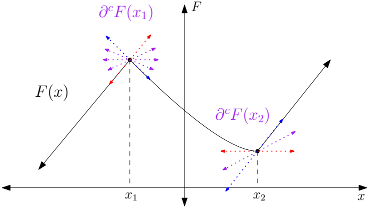
Nonsmooth calculus and optimization for machine learning: first-order sampling and implicit differentiation



Tam Le (PhD at TSE and Université Toulouse Capitole, funded by ANITI)

advised by Jérôme Bolte (TSE) and Edouard Pauwels (TSE),
joint work with Antonio Silveti-Falls (now CentraleSupelec)

Nonsmooth optimization in machine learning

Many machine learning problems write as an optimization problem.

$$\underset{\mathbf{w}\in\mathbb{R}^p}{\text{Minimize}} \quad F(\mathbf{w})$$

First-order methods, e.g. **gradient method** are really popular

$$\boxed{w_{k+1} = w_k - \alpha_k \nabla F(w_k)}$$

Nonsmooth optimization in machine learning

Many machine learning problems write as an optimization problem.

$$\underset{\mathbf{w}\in\mathbb{R}^p}{\text{Minimize}} \quad F(\mathbf{w})$$

First-order methods, e.g. **gradient method** are really popular

$$\boxed{w_{k+1} = w_k - \alpha_k(\nabla F(w_k) + \epsilon_k)}$$

- Automatic differentiation libraries (Pytorch, Tensorflow, JAX)
- Can be adapted to handle massive training sets (Stochastic algorithms)

Nonsmooth optimization in machine learning

Many machine learning problems write as an optimization problem.

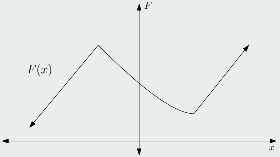
$$\underset{\mathbf{w}\in\mathbb{R}^p}{\text{Minimize}} \quad F(\mathbf{w})$$

First-order methods, e.g. **gradient method** are really popular

$$\boxed{w_{k+1} = w_k - \alpha_k(\nabla F(w_k) + \epsilon_k)}$$

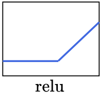
- Automatic differentiation libraries (Pytorch, Tensorflow, JAX)
- Can be adapted to handle massive training sets (Stochastic algorithms)

Observation: In many practical situations, F is **nonconvex and nonsmooth**.

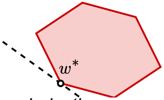


F is often nonsmooth!

Activation functions in Deep Learning



Polyhedral constraints



Regularization

Sparsification

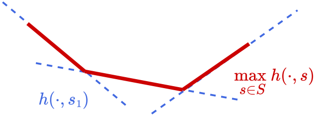
Sorting operations

$$a_{\sigma(1)} \leq a_{\sigma(2)} \leq \ldots \leq a_{\sigma(n)}$$

Median

Quantiles

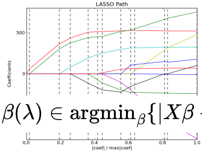
Max value functions



Min max problems

Robust learning

Solution paths



$$\beta(\lambda) \in \operatorname{argmin}_\beta \{|X\beta - Y| + \lambda|\beta|_1\}$$

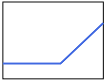
Bi-level optimization

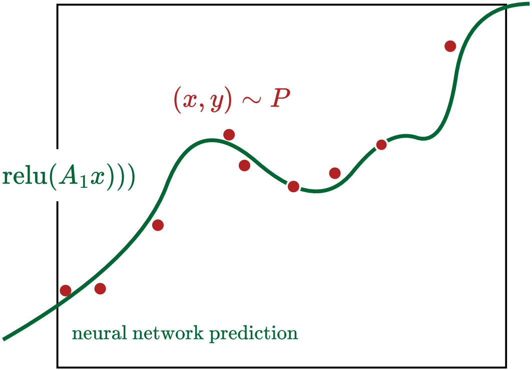
Hyperparameter selection

Neural network training

$$\min_{w \in \mathbb{R}^p} \mathbb{E}_{(x,y)\sim P}\left[(h(w,x) - y)^2\right]$$

$$h(w,x) = \mathrm{relu}(A_\ell \mathrm{relu}(A_{\ell-1} \ldots \mathrm{relu}(A_1 x)))$$


relu



Bi-level optimization

$$\min_{w \in \mathbb{R}^p} F(w, y)$$

$$y \in \operatorname*{argmin}_{\theta \in \mathcal{C}} g(w, \theta).$$

Ex: Hyperparameter selection

$$\min_{\lambda \in \mathbb{R}^p} \quad \mathrm{Criterion}(\beta(\lambda))$$

$$\beta(\lambda) \in \mathrm{argmin}_\beta \{|X\beta - Y| + \lambda|\beta|_1\}$$

Outline

Observation: a gap between nonsmooth opt. notions and practice in ML

- First-order methods, "gradient methods" are used on nonsmooth functions in practice, thanks to automatic differentiation libraries.
- The classical notions for nonsmooth functions (**Clarke subgradient**) do not explain this practice.

Outline

Observation: a gap between nonsmooth opt. notions and practice in ML

- First-order methods, "gradient methods" are used on nonsmooth functions in practice, thanks to automatic differentiation libraries.
- The classical notions for nonsmooth functions (**Clarke subgradient**) do not explain this practice.

Solution: Nonsmooth calculus with Conservative derivatives

We focus on generalized derivatives called **Conservative derivatives**, which justifies automatic differentiation.

We propose **two extensions**

- Differentiation under nonsmooth expectation $\rightarrow$ stochastic methods.
- Nonsmooth Implicit differentiation $\rightarrow$ gradient methods for bi-level problems.

Outline

Observation: a gap between nonsmooth opt. notions and practice in ML

- First-order methods, "gradient methods" are used on nonsmooth functions in practice, thanks to automatic differentiation libraries.
- The classical notions for nonsmooth functions (**Clarke subgradient**) do not explain this practice.

Solution: Nonsmooth calculus with Conservative derivatives

We focus on generalized derivatives called **Conservative derivatives**, which justifies automatic differentiation.

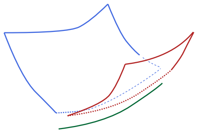
We propose **two extensions**

- Differentiation under nonsmooth expectation → stochastic methods.
- Nonsmooth Implicit differentiation → gradient methods for bi-level problems.

Output: Analysis of nonsmooth first-order algorithms.

Using **ODE approaches**, we show the convergence of nonsmooth stochastic optimization algorithms **as implemented in practice**.

Nonsmooth nonconvex optimization



Classical concepts vs machine learning practice

A glance at the differentiable setting

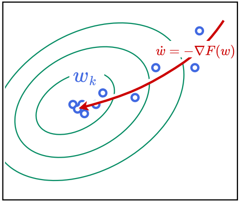
Gradient method, $\alpha_k < 0$, $\alpha_k \to 0$

$$w_{k+1} = w_k - \alpha_k \nabla F(w_k).$$

1. **Descent mechanism**: the method approximates gradient curves

$$\dot{w}(t) = -\nabla F(w(t))$$

- F decreases along w
- $w(t)$ accumulates around **critical points** $\{w \ : \ 0 = \nabla F(w)\}$



A glance at the differentiable setting

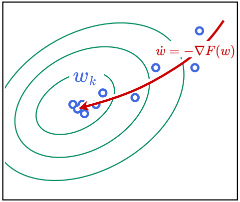
Gradient method, $\alpha_k < 0$, $\alpha_k \to 0$

$$\boxed{w_{k+1} = w_k - \alpha_k \nabla F(w_k).}$$

1. **Descent mechanism**: the method approximates gradient curves

$$\dot{w}(t) = -\nabla F(w(t))$$

- F decreases along w
- $w(t)$ accumulates around **critical points** $\{w \ : \ 0 = \nabla F(w)\}$



2. ∇F can be computed by **calculus** rules: $\nabla(f+g) = \nabla f + \nabla g$, $\mathrm{Jac}(u \circ v) = (\mathrm{Jac}\, u \circ v)\, \mathrm{Jac}\, v \ \ldots$

A glance at the differentiable setting

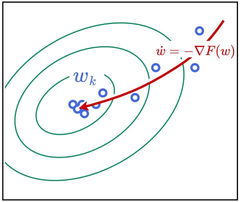
Gradient method, $\alpha_k < 0$, $\alpha_k \to 0$

$$\boxed{w_{k+1} = w_k - \alpha_k \nabla F(w_k).}$$

1. **Descent mechanism**: the method approximates gradient curves

$$\dot{w}(t) = -\nabla F(w(t))$$

- F decreases along w
- $w(t)$ accumulates around **critical points** $\{w \ : \ 0 = \nabla F(w)\}$



2. ∇F can be computed by **calculus** rules: $\nabla(f+g) = \nabla f + \nabla g$, $\mathrm{Jac}(u \circ v) = (\mathrm{Jac}\, u \circ v)\, \mathrm{Jac}\, v \ \ldots$

What if F is nonsmooth?

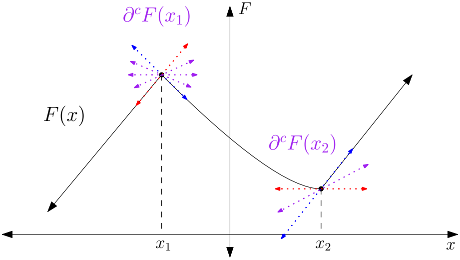
The Clarke subgradient: a gradient for nonsmooth functions

Let $F : \mathbb{R}^n \to \mathbb{R}$ be locally Lipschitz, differentiable on diff_F of full Lebesgue measure.

Clarke subgradient ∂^c

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

(extends to Jacobians)



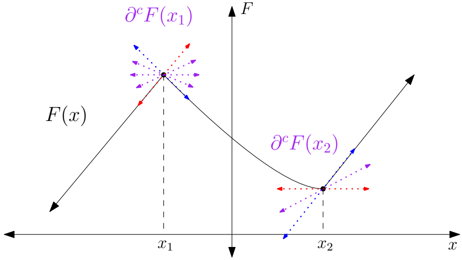
The Clarke subgradient: a gradient for nonsmooth functions

Let $F : \mathbb{R}^n \to \mathbb{R}$ be locally Lipschitz, differentiable on diff_F of full Lebesgue measure.

Clarke subgradient ∂^c

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

(extends to Jacobians)



$\partial^c F$ is graph-closed, locally bounded, convex-valued.

$\to$ existence of the continuous time dynamics $\dot{w} \in -\partial^c F(w)$.

$\to$ subgradient method $w_{k+1} \in w_k - \alpha_k \partial^c F(w_k)$ as ODE discretization.

Descent along curves

Do we have descent along $\dot{w} \in -\partial^c F(w)$?

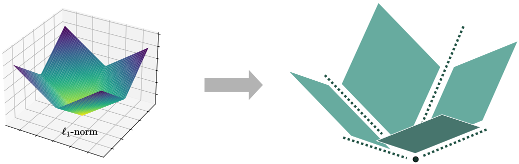
Not in general. There exist Lipschitz functions F such that $\partial^c F = B(0,1)$ everywhere.

But often in practice!

Stratification of “usual” (definable) functions

Key idea: “Simple” compositional structure leads to a **stratified** landscape

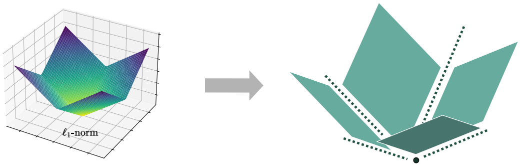
Piecewise affine functions: functions involving affine constraints and functions $(\text{if}, \text{else}, +, \cdot, \leq, \geq)$ decompose into affine pieces



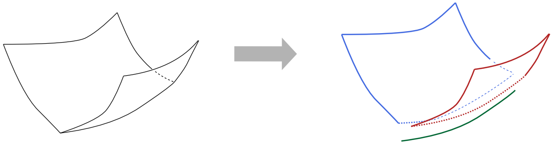
Stratification of "usual" (definable) functions

Key idea: "Simple" compositional structure leads to a **stratified** landscape

Piecewise affine functions: functions involving affine constraints and functions $(\mathrm{if}, \mathrm{else}, +, \cdot, \leq, \geq)$ decompose into affine pieces

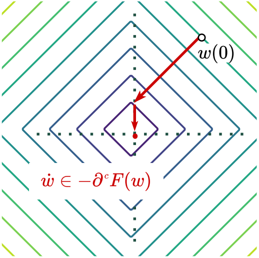


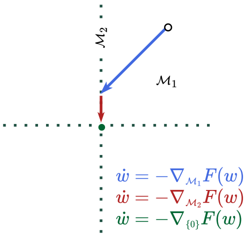
Definable functions: functions involving elementary operations $(\mathrm{if}, \mathrm{else}, +, \cdot, \times, \exp, \log)$, decompose into **smooth manifolds**



Usual functions are path-differentiable

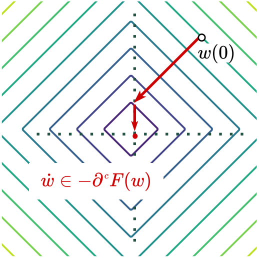
Subgradient flow $\approx$ Gradient flows along stratification manifolds

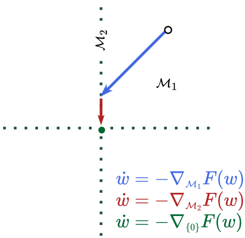




Usual functions are path-differentiable

Subgradient flow ≈ Gradient flows along stratification manifolds





Path differentiability (descent along curves) Valadier 1989

F (locally Lipschitz) is called **path-differentiable** if for all absolutely continuous curve γ, for almost all t,

$$(F \circ \gamma)'(t) = \langle \partial^c F(\gamma(t)), \dot{\gamma}(t) \rangle$$

In this case F **decreases along subgradient curves**

Definable functions are path differentiable Bolte et al. 2007, Davis et al. 2019

- The Clarke subgradient provides descent for path-differentiable functions.

  → Can we easily compute elements of the Clarke subgradient? In particular, does **automatic differentiation** output **Clarke subgradients**?

- Functions implemented in practice are definable, hence they are path-differentiable.

  → What can we say outside these functions, for instance for **expectations** $F(w) = \mathbb{E}_{\xi \sim P}[f(w, \xi)]$?

What is automatic differentiation?

Auto-differentiation libraries (PyTorch, Tensorflow) differentiate programs

$$\mathrm{relu}(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{else.} \end{cases} \quad \xrightarrow[\text{autodiff}]{} \quad \mathrm{relu}'(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{else.} \end{cases}$$

What is automatic differentiation?

Auto-differentiation libraries (PyTorch, Tensorflow) differentiate programs

$$\mathrm{relu}(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{else.} \end{cases} \quad \xrightarrow[\text{autodiff}]{} \quad \mathrm{relu}'(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{else.} \end{cases}$$

What about compositions, e.g. neural nets?

What is automatic differentiation?

Auto-differentiation libraries (PyTorch, Tensorflow) differentiate programs

$$\mathrm{relu}(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{else.} \end{cases} \quad \underset{\text{autodiff}}{\longrightarrow} \quad \mathrm{relu}'(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{else.} \end{cases}$$

What about compositions, e.g. neural nets?

Automatic differentiation **applies the chain rule to Clarke derivatives**.

$$\text{if} \quad f(w) = g_r \circ g_{r-1} \circ \ldots \circ g_1(w)$$

$$\begin{aligned} \text{then} \quad \mathrm{autodiff}_w\, f(w) &\in \partial^c g_r(g_{r-1} \circ \ldots \circ g_1(w))^T \\ &\times \mathrm{Jac}^c\, g_{r-1}(g_{r-2} \circ \ldots \circ g_1(w)) \times \ldots \times \mathrm{Jac}^c_w\, g_1(w). \end{aligned}$$

What is automatic differentiation?

Auto-differentiation libraries (PyTorch, Tensorflow) differentiate programs

$$\mathrm{relu}(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{else.} \end{cases} \quad \xrightarrow[\text{autodiff}]{} \quad \mathrm{relu}'(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{else.} \end{cases}$$

What about compositions, e.g. neural nets?

Automatic differentiation **applies the chain rule to Clarke derivatives**.

$$\text{if} \quad f(w) = g_r \circ g_{r-1} \circ \ldots \circ g_1(w)$$

$$\text{then} \quad \mathrm{autodiff}_w\, f(w) \in \partial^c g_r(g_{r-1} \circ \ldots \circ g_1(w))^T \times \mathrm{Jac}^c\, g_{r-1}(g_{r-2} \circ \ldots \circ g_1(w)) \times \ldots \times \mathrm{Jac}^c_w\, g_1(w).$$

But do calculus rules apply to Clarke derivatives?

No.

- (Sum rule) $\partial^c(f+g) \subsetneq \partial^c f + \partial^c g$
- (Composition rule) $\mathrm{Jac}^c(F \circ G) \subsetneq \mathrm{conv}\, \mathrm{Jac}^c F(G)\, \mathrm{Jac}^c G$

Formal differentiation in machine learning.

Yet, autodiff is used extensively in machine learning:

Formal differentiation in machine learning.

Yet, autodiff is used extensively in machine learning:

Example 1. Stochastic methods. To minimize $F = \mathbb{E}_{\xi \sim P}[f(\cdot, \xi)]$ we may sample $\nabla_w f(\cdot, \xi)$, $\xi \sim P$ to have a noisy estimate of

$$\mathbb{E}_{\xi \sim P}[\nabla_w f(\cdot, \xi)] = \nabla F \qquad \text{(Differentiation under integral)}$$

$\rightarrow$ Practice: f is nonsmooth, $\mathrm{autodiff}_w\, f(w, \xi)$ is sampled instead of $\nabla_w f(w, \xi)$.

Formal differentiation in machine learning.

Yet, autodiff is used extensively in machine learning:

Example 1. Stochastic methods. To minimize $F = \mathbb{E}_{\xi \sim P}[f(\cdot, \xi)]$ we may sample $\nabla_w f(\cdot, \xi)$, $\xi \sim P$ to have a noisy estimate of

$$\mathbb{E}_{\xi \sim P}[\nabla_w f(\cdot, \xi)] = \nabla F \qquad \text{(Differentiation under integral)}$$

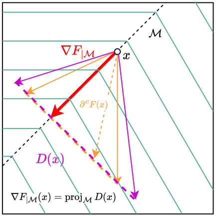
→ Practice: f is nonsmooth, $\mathrm{autodiff}_w\, f(w, \xi)$ is sampled instead of $\nabla_w f(w, \xi)$.

Example 2. Implicit differentiation. $H(x, y) = 0$, H continuously differentiable. How to differentiate y w.r.t. x?

$$\frac{\partial y}{\partial x} = -\left[\frac{\partial H}{\partial y}\right]^{-1} \frac{\partial H}{\partial x} \qquad \text{(Implicit differentiation)}$$

→ Practice: H is **optimality conditions**, hence nonsmooth. ∂H is replaced by $\mathrm{autodiff}\, H$.

Nonsmooth calculus
with conservative derivatives



Conservative gradients, Bolte & Pauwels (2021)

Let $F : \mathbb{R}^n \to \mathbb{R}$ be locally Lipschitz, and let $D : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ be a set-valued map. D is a **conservative gradient** for F if

- It generates descent trajectories

For all absolutely continuous curve $\gamma : [0,1] \to \mathbb{R}^n$,

$$\frac{\mathrm{d}}{\mathrm{d}t}(F \circ \gamma)(t) = \langle v, \dot{\gamma}(t) \rangle, \quad \forall v \in D(\gamma(t)),$$

for almost all $t \in [0,1]$.

(extends to Jacobians)

F decreases along $\dot{\gamma} \in -D(\gamma)$

Conservative gradients, Bolte & Pauwels (2021)

Let $F : \mathbb{R}^n \to \mathbb{R}$ be locally Lipschitz, and let $D : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ be a set-valued map. D is a **conservative gradient** for F if

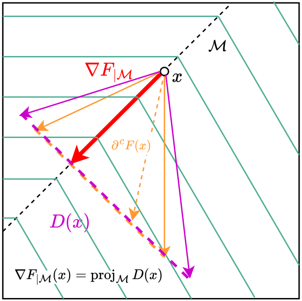
- It generates descent trajectories

For all absolutely continuous curve $\gamma : [0,1] \to \mathbb{R}^n$,

$$\frac{\mathrm{d}}{\mathrm{d}t}(F \circ \gamma)(t) = \langle v, \dot\gamma(t)\rangle, \quad \forall v \in D(\gamma(t)),$$

for almost all $t \in [0,1]$.

(extends to Jacobians)



F decreases along $\dot\gamma \in -D(\gamma)$

Conservative gradients, Bolte & Pauwels (2021)

Let $F : \mathbb{R}^n \to \mathbb{R}$ be locally Lipschitz, and let $D : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ be a set-valued map.
D is a **conservative gradient** for F if

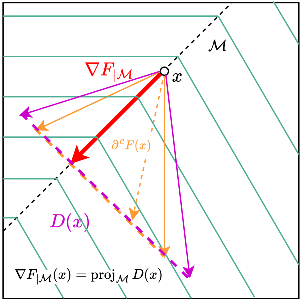
- It generates descent trajectories

For all absolutely continuous curve $\gamma : [0,1] \to \mathbb{R}^n$,

$$\frac{\mathrm{d}}{\mathrm{d}t}(F \circ \gamma)(t) = \langle v, \dot\gamma(t) \rangle, \quad \forall v \in D(\gamma(t)),$$

for almost all $t \in [0,1]$.

(extends to Jacobians)



F decreases along $\dot\gamma \in -D(\gamma)$

- Existence of the flow $\dot\gamma \in -D(\gamma)$:

D is graph-closed, nonempty (convex) valued, locally bounded.

Conservative calculus

- **Clarke subgradient of path diff.** If F is path differentiable, then $\partial^c F$ is conservative.

- **Sum rule** Let D_f, D_g be conservative gradients for f and g, $D_f + D_g$ is conservative gradient for $f + g$.

- **Chain rule** Let J_u, J_v be conservative Jacobians for u and v. $J_u(v)J_v$ is conservative Jacobian for $u \circ v$. $\rightarrow$ automatic differentiation outputs conservative Jacobians.

- **Calculus to regularity** If F has a conservative gradient, then it is path differentiable.

Conservative calculus

- **Clarke subgradient of path diff.** If F is path differentiable, then $\partial^c F$ is conservative.

- **Sum rule** Let D_f, D_g be conservative gradients for f and g, $D_f + D_g$ is conservative gradient for $f + g$.

- **Chain rule** Let J_u, J_v be conservative Jacobians for u and v. $J_u(v)J_v$ is conservative Jacobian for $u \circ v$. $\rightarrow$ automatic differentiation outputs conservative Jacobians.

- **Calculus to regularity** If F has a conservative gradient, then it is path differentiable.

An extension of the conservative calculus:
Integral rule of conservative gradients

"Differentiating" under integral/expectation $F = \mathbb{E}_{\xi \sim P}[f(\cdot, \xi)]$

Integral rule: motivation in stochastic optimization

Stochastic minimization

Consider

$$F(w) := \mathbb{E}_{\xi \sim P}[f(\cdot, \xi)]$$

Under mild conditions, one can differentiate under $\mathbb{E}$:

$$\nabla F = \mathbb{E}_{\xi \sim P}[\nabla_w f(\cdot, \xi)]$$

First-order sampling: Sample $\xi \sim P$, $\nabla_w f(w, \xi) \approx \nabla F(w)$
$\rightarrow$ Stochastic gradient method: $w_{k+1} = w_k - \alpha_k \nabla_w f(w_k, \xi_k)$

Integral rule: motivation in stochastic optimization

Stochastic minimization

Consider

$$F(w) := \mathbb{E}_{\xi \sim P}[f(\cdot, \xi)]$$

Under mild conditions, one can differentiate under $\mathbb{E}$:

$$\nabla F = \mathbb{E}_{\xi \sim P}[\nabla_w f(\cdot, \xi)]$$

First-order sampling: Sample $\xi \sim P$, $\nabla_w f(w, \xi) \approx \nabla F(w)$
$\rightarrow$ Stochastic gradient method: $w_{k+1} = w_k - \alpha_k \nabla_w f(w_k, \xi_k)$

In practice, $f(\cdot, \xi)$ is nonsmooth, and

$$\partial^c F \subsetneq \mathbb{E}_{\xi \sim P}[\partial^c_w f(\cdot, \xi)]$$

But we have access to a **conservative gradient** of $f(\cdot, \xi)$, $D(\cdot, \xi)$, e.g., **autodiff**.

Question: What is the expectation $\mathbb{E}_{\xi \sim P}[D(\cdot, \xi)]$?

Integral rule of conservative gradient

Theorem (Bolte, L., Pauwels 2022)

If $D(\cdot, \xi)$ is conservative gradient for $f(\cdot, \xi)$,
then $\mathbb{E}_{\xi \sim P}[D(\cdot, \xi)]$ is a conservative gradient for F.

Integral rule of conservative gradient

Theorem (Bolte, L., Pauwels 2022)

If $D(\cdot, \xi)$ is conservative gradient for $f(\cdot, \xi)$,
then $\mathbb{E}_{\xi \sim P}[D(\cdot, \xi)]$ is a conservative gradient for F.

Assumptions:
1. (Measurability assumptions) ...
2. (Boundedness assumption) For all compact subset $C \subset \mathbb{R}^p$, there exists an integrable function $\kappa : S \to \mathbb{R}_+$ such that for all

$$(x, s) \in C \times S,\ \|D(x, s)\| \leq \kappa(s)$$

where for $(x, s) \in \mathbb{R}^p \times S$, $\|D(x, s)\| := \sup_{y \in D(x,s)} \|y\|$.

Main outcomes:

- **Justifies first-order sampling** in practical implementations: e.g. autodiff or implicit differentiation.
- F **(expectation) is path-differentiable**, under simple assumptions.

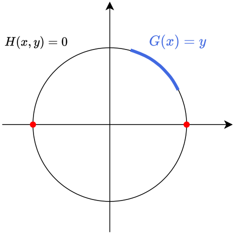
Many other applications of conservative calculus! *Differentiating max functions, ODE solutions, algorithmic recursions, function approximations*

- Pauwels, Conservative Parametric Optimality and the Ridge Method for Tame Min-Max Problems (2023)
- Marx-Pauwels, Path differentiability of ODE flows (2022)
- Bolte-Pauwels-Vaiter, Automatic differentiation of nonsmooth iterative algorithms (2022)
- Iutzeler-Pauwels-Vaiter, Derivatives of SGD (2024)
- Schechtman, The gradient's limit of a definable family of functions is a conservative set-valued field (2024)

Nonsmooth implicit differentiation formula.

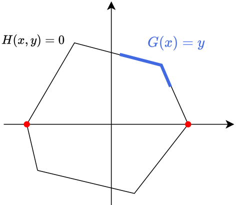
Bolte, L., Pauwels, Silveti-Falls (2021)

Setting: $y \in \mathrm{argmin}_\theta\, g(x,\theta)$ written as $H(x,y)=0$
How to differentiate y with respect to x?
Applications: Bi-level opt., optimization layers, hyperparameter selection ...



H smooth:

$$\frac{\partial G}{\partial x} = -\left[\frac{\partial H}{\partial y}\right]^{-1}\frac{\partial H}{\partial x}$$

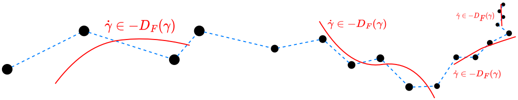


H nonsmooth, path differentiable:

$$J_G(x) = \{-B^{-1}A \mid [A\ B] \in \mathrm{Autodiff}\, H(x, G(x))\}$$

is a conservative Jacobian

Analysis of the stochastic subgradient method "as implemented practice"



Nonsmooth stochastic gradient method

We consider the problem

$$\underset{\mathbf{w}\in\mathbb{R}^p}{\text{Minimize}} \quad F(\mathbf{w}) := \mathbb{E}_{\xi\sim P}[f(\mathbf{w},\xi)],$$

We study a nonsmooth stochastic gradient method

$$w_{k+1} \in w_k - \alpha_k D(w_k, \xi_k). \tag{1}$$

For $\xi \in \mathbb{R}^m$, $D(\cdot,\xi)$ is a conservative gradient for $f(\cdot,\xi)$ $\rightarrow$ encompasses practical calculus: autodiff, implicit differentiation ...

Nonsmooth stochastic gradient method

We consider the problem

$$\underset{\mathbf{w}\in\mathbb{R}^p}{\text{Minimize}} \quad F(\mathbf{w}) := \mathbb{E}_{\xi\sim P}[f(\mathbf{w},\xi)],$$

We study a nonsmooth stochastic gradient method

$$w_{k+1} \in w_k - \alpha_k D(w_k, \xi_k). \tag{1}$$

For $\xi \in \mathbb{R}^m$, $D(\cdot,\xi)$ is a conservative gradient for $f(\cdot,\xi) \to$ encompasses practical calculus: autodiff, implicit differentiation ...

Integral rule: (1) writes

$$w_{k+1} \in w_k - \alpha_k(D_F(w_k) + \epsilon_k),$$

where $D_F = \mathbb{E}_{\xi\sim P}[D(\cdot,\xi)]$ is conservative gradient for F, ϵ_k has zero conditional mean w.r.t. w_k.

The ODE approach

Key idea: Studying algorithms as ODE discretizations.

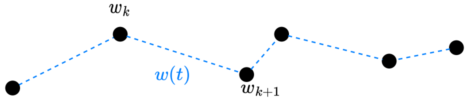
$$\frac{w_{k+1} - w_k}{\alpha_k} = -D_F(w_k) + \epsilon_k \qquad \rightsquigarrow \qquad \dot{\gamma} \in -D_F(\gamma) \tag{2}$$

The ODE approach

Key idea: Studying algorithms as ODE discretizations.

$$\frac{w_{k+1} - w_k}{\alpha_k} = -D_F(w_k) + \epsilon_k \qquad \rightsquigarrow \qquad \dot{\gamma} \in -D_F(\gamma) \tag{2}$$

Interpolated process w:



Asymptotic pseudo trajectory Benaim (1999), Benaim-Hofbauer-Sorin (2005)

$w : \mathbb{R}_+ \to \mathbb{R}^p$ is an **asymptotic pseudo trajectory** (APT) if for all $T > 0$,

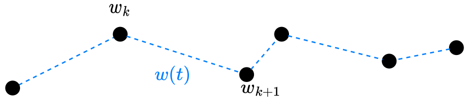
$$\lim_{t \to \infty} \inf_{\gamma \text{ solution}} \sup_{s \in [0,T]} \|w(t+s) - \gamma(s)\| = 0.$$

The ODE approach

Key idea: Studying algorithms as ODE discretizations.

$$\frac{w_{k+1} - w_k}{\alpha_k} = -D_F(w_k) + \epsilon_k \qquad \rightsquigarrow \qquad \dot{\gamma} \in -D_F(\gamma) \tag{2}$$

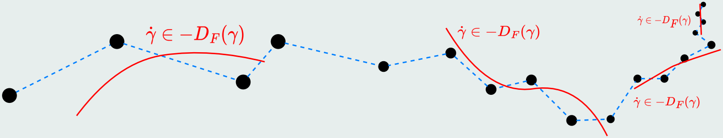
Interpolated process w:



Asymptotic pseudo trajectory Benaim (1999), Benaim-Hofbauer-Sorin (2005)

$w : \mathbb{R}_+ \to \mathbb{R}^p$ is an **asymptotic pseudo trajectory** (APT) if for all $T > 0$,

$$\lim_{t\to\infty} \inf_{\gamma \text{ solution}} \sup_{s\in[0,T]} \|w(t+s) - \gamma(s)\| = 0.$$



Convergence results

F decreases along $\dot{\gamma} \in -D_F(\gamma)$ (**conservative gradient**)
$+$ w is APT
$=$ Asymptotic descent

Convergence results

- **Essential** accumulation points w^* satisfy $0 \in D_F(w^*)$.
- Under definable assumptions on P, $\{F(w) \ : \ 0 \in D_F(w)\}$ has empty interior: $F(w_k)$ **converges** and **all accumulation points** satisfy $0 \in D_F(w^*)$.

Convergence results

F decreases along $\dot{\gamma} \in -D_F(\gamma)$ (**conservative gradient**)
$+$ w is APT
$=$ Asymptotic descent

Convergence results

- **Essential** accumulation points w^* satisfy $0 \in D_F(w^*)$.
- Under definable assumptions on P, $\{F(w) \ : \ 0 \in D_F(w)\}$ has empty interior: $F(w_k)$ **converges** and **all accumulation points** satisfy $0 \in D_F(w^*)$.

Essential accumulation point w^* is such that for all open $U \ni w^*$,

$$\limsup_{k \to \infty} \frac{\sum_{i=0}^{k} \alpha_i \mathbf{1}_{w_i \in U}}{\sum_{i=0}^{k} \alpha_i} > 0 \quad \text{a.s.}$$

Proportion of time spent around w^*

Convergence results

F decreases along $\dot{\gamma} \in -D_F(\gamma)$ (**conservative gradient**)
$+$ w is APT
$=$ Asymptotic descent

Convergence results

- **Essential** accumulation points w^* satisfy $0 \in D_F(w^*)$.
- Under definable assumptions on P, $\{F(w) \ : \ 0 \in D_F(w)\}$ has empty interior: $F(w_k)$ **converges** and **all accumulation points** satisfy $0 \in D_F(w^*)$.

Essential accumulation point w^* is such that for all open $U \ni w^*$,

$$\limsup_{k \to \infty} \frac{\sum_{i=0}^k \alpha_i \mathbf{1}_{w_i \in U}}{\sum_{i=0}^k \alpha_i} > 0 \quad \text{a.s.}$$

Proportion of time spent around w^*

Assumptions

- $(w_k)_{k \in \mathbb{N}}$ bounded a.s.
- $\alpha_k > 0$, $\sum \alpha_k = \infty$, $\sum \alpha_k^2 < \infty$
- $\|D(w,s)\| \leq \kappa(s)\psi(w)$, κ square integrable, ψ locally bounded.

Conclusion and perspectives

- **Conservative derivatives** provide a justification to many implementations of "gradient" method
  - nonsmooth automatic differentiation,
  - **Differentiation under integral** $\rightarrow$ nonsmooth stochastic algorithms
  - **Implicit differentiation** $\rightarrow$ can be applied to bi-level programming, optimization layers, implicit layers

Conclusion and perspectives

- **Conservative derivatives** provide a justification to many implementations of “gradient” method
  - nonsmooth automatic differentiation,
  - **Differentiation under integral** $\rightarrow$ nonsmooth stochastic algorithms
  - **Implicit differentiation** $\rightarrow$ can be applied to bi-level programming, optimization layers, implicit layers
  - many other applications: value function, differentiation of ODE flows, monotone inclusion, iterative algorithms...
- Chain rule along curves $\rightarrow$ **ODE approach** $\rightarrow$ **convergence results.**

Conclusion and perspectives

- **Conservative derivatives** provide a justification to many implementations of "gradient" method
  - nonsmooth automatic differentiation,
  - **Differentiation under integral** $\rightarrow$ nonsmooth stochastic algorithms
  - **Implicit differentiation** $\rightarrow$ can be applied to bi-level programming, optimization layers, implicit layers
  - many other applications: value function, differentiation of ODE flows, monotone inclusion, iterative algorithms...

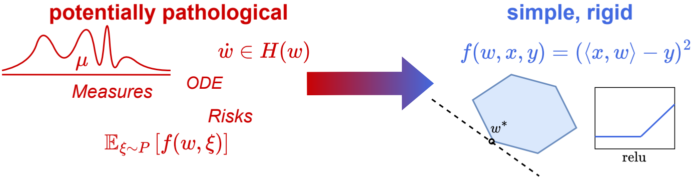
- Chain rule along curves $\rightarrow$ **ODE approach** $\rightarrow$ **convergence results.**

Importance of "rigidity" (definability):

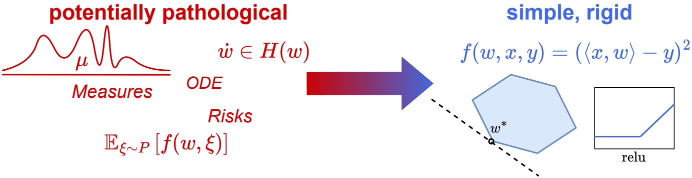
- Chain rule of simple (definable) functions
- Definable $P \rightarrow$ stronger convergence
- Avoidance of artefacts, beyond the general criticality notion $0 \in D_F$:

"For most initialization w_0 and stepsizes, accumulation points satisfy $0 \in \partial^c F(w^*)$."

A good property may persists "outside" a class of nice objects: "$\mathbb{E}_{\xi \sim P}[D(\cdot, \xi)]$ is a conservative gradient"



A good property may persists "outside" a class of nice objects:
"$\mathbb{E}_{\xi\sim P}[D(\cdot,\xi)]$ is a conservative gradient"



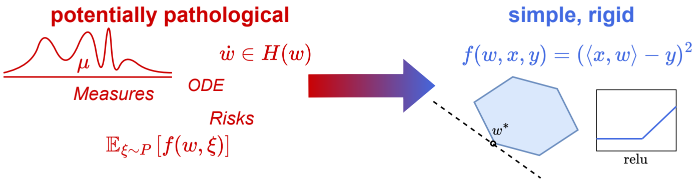
How far can the analysis benefit from this?

- Convergence theory: constant (large?) steps, complexity...
- Algorithmic extensions: constraints, biased oracle; beyond vanishing stepsizes: adaptive algorithms; non i.i.d. samples, ...

Can we design algorithms based on this nice geometry?

When nonsmoothness is "useful" (robustness, bi-level opt...).

A good property may persists "outside" a class of nice objects:
"$\mathbb{E}_{\xi\sim P}[D(\cdot,\xi)]$ is a conservative gradient"



How far can the analysis benefit from this?

- Convergence theory: constant (large?) steps, complexity...
- Algorithmic extensions: constraints, biased oracle; beyond vanishing stepsizes: adaptive algorithms; non i.i.d. samples, ...

Can we design algorithms based on this nice geometry?

When nonsmoothness is "useful" (robustness, bi-level opt...).

Thank you!!!