Differentiating Solution Mappings of Non-smooth Optimization Problems

Peter Ochs

Mathematical Optimization for Data Science
Saarland University

— 04.09.2023 —

joint work: Sheheryar Mehmood

Variational Problems with Regularization

$$\min_x E_\lambda(x)\,, \quad E_\lambda(x) = \underbrace{D(x)}_{\text{data term}} + \lambda \underbrace{R(x)}_{\text{regularization term}} \,.$$

- $\lambda > 0$ is a regularization parameter.

Regularization weight has a huge impact on the result!

What about learning the whole regularization term?

$$R(x, \nu, \vartheta) := \sum_{i,j} \left(\sum_{k=1}^{m} \nu_k \rho\Big(\sum_{l=1}^{L} \vartheta_{kl} (K_l x)_{ij} \Big) \right)$$

- K_l are predefined basis filters (e.g. DCT filter)
- ρ is a potential function (convex)

(This regularizer reflects a 1-hidden-layer neural network.)

Variational Problems with Regularization

What about learning the whole regularization term?

$$R(x,\nu,\vartheta) := \sum_{i,j}\left(\sum_{k=1}^{m}\nu_k\rho\Big(\sum_{l=1}^{L}\vartheta_{kl}(K_l x)_{ij}\Big)\right)$$

- K_l are predefined basis filters (e.g. DCT filter)
- ρ is a potential function (convex)

(This regularizer reflects a 1-hidden-layer neural network.)

How to find the best weights ν, ϑ?

- hand-tuning and grid search are not feasible
- sampling and regression of loss function using Gaussian processes or Random Fields (up to ≈ 200 parameters)
- **gradient based bi-level optimization** (several 100 000 parameters)

Applications in Machine Learning

Similar Problems:

- (Hyper)Parameter Learning [Domke '12], [Kunisch, Pock '13], [O. et al '16], ...
- Implicit Models [Amos, Kolter '17], [Agrawal et al. '19], [Bai, Kolter, Koltun '19], [Chen et al. '21], ...
- Meta Learning [Hospedales et al. '21], ...
- Learning to Optimize [Chen et al. '21], ...
- Sensitivity Analysis.

Bilevel optimization / parameter learning

$$\min_{\theta \in \mathbb{R}^P} \quad \mathcal{L}(x^*(\theta), \theta) \qquad \text{(upper level)}$$

$$s.t. \quad x^*(\theta) \in \arg\min_{x \in \mathbb{R}^N} E(x, \theta) \qquad \text{(lower level)}$$

- $\theta \in \mathbb{R}^P$: optimization variable **parameter (vector)**.
- $\mathcal{L} \colon \mathbb{R}^N \times \mathbb{R}^P \to \mathbb{R}$: **smooth loss function**.
- $E \colon \mathbb{R}^N \times \mathbb{R}^P \to \overline{\mathbb{R}}$: parametric (energy) **minimization** problem. (convex for each $\theta \in \mathbb{R}^P$)
- $x^* \colon \mathbb{R}^P \to \mathbb{R}^N$ is a selection of the **solution mapping** of E.

Bilevel optimization / parameter learning

$$\min_{\theta \in \mathbb{R}^P} \quad \mathcal{L}(x^*(\theta), \theta) \qquad \text{(upper level)}$$

$$s.t. \quad x^*(\theta) \in \arg\min_{x \in \mathbb{R}^N} E(x, \theta) \qquad \text{(lower level)}$$

- $\theta \in \mathbb{R}^P$: optimization variable **parameter (vector)**.
- $\mathcal{L} \colon \mathbb{R}^N \times \mathbb{R}^P \to \mathbb{R}$: **smooth loss function**.
- $E \colon \mathbb{R}^N \times \mathbb{R}^P \to \overline{\mathbb{R}}$: parametric (energy) **minimization** problem. (convex for each $\theta \in \mathbb{R}^P$)
- $x^* \colon \mathbb{R}^P \to \mathbb{R}^N$ is a selection of the **solution mapping** of E.

Gradient based optimization requires $\nabla_\theta \mathcal{L}(x^*(\theta^{(k)}), \theta^{(k)})$,

i.e., in particular, the sensitivity of the solution mapping $\frac{\partial x^*}{\partial \theta}$.

$$\min_{\theta \in \mathbb{R}^P} \quad \mathcal{L}(x^*(\theta), \theta) \qquad \text{(upper level)}$$

$$s.t. \quad x^*(\theta) \in \arg\min_{x \in \mathbb{R}^N} E(x, \theta) \qquad \text{(lower level)}$$

- $\theta \in \mathbb{R}^P$: optimization variable **parameter (vector)**.
- $\mathcal{L}\colon \mathbb{R}^N \times \mathbb{R}^P \to \mathbb{R}$: **smooth loss function**.
- $E\colon \mathbb{R}^N \times \mathbb{R}^P \to \overline{\mathbb{R}}$: parametric (energy) **minimization** problem. (convex for each $\theta \in \mathbb{R}^P$)
- $x^*\colon \mathbb{R}^P \to \mathbb{R}^N$ is a selection of the **solution mapping** of E.

Gradient based optimization requires $\nabla_\theta \mathcal{L}(x^*(\theta^{(k)}), \theta^{(k)})$,

i.e., in particular, the sensitivity of the solution mapping $\frac{\partial x^*}{\partial \theta}$.

Outline:	E	**type of bilevel algorithm**
(I) smooth setting	smooth	gradient
(II) partly smooth setting	structured non-smooth	gradient
(III) non-smooth setting	non-smooth	generalized derivative

(I) Smooth Setting: Strategies for differentiation

In the following: $x^*(\theta) \in \arg\min_{x\in\mathbb{R}^N} E(x,\theta)$ **(E smooth)**

Goal: Compute $\frac{\partial x^*}{\partial \theta}(\theta)$

In the following: $x^*(\theta) \in \arg\min_{x\in\mathbb{R}^N} E(x,\theta)$ **(E smooth)**

Goal: Compute $\frac{\partial x^*}{\partial \theta}(\theta)$

Strategies:

- Implicit differentiation. (requires E strictly convex)
- Unrolling an algorithm. *(use automatic differentiation)*
- Unrolling of a fixed point equation.

Warning:

- Even for smooth E, the solution mapping $x^*(\theta)$ is often non-smooth.
- Solution mapping multivalued, when $\arg\min_{x\in\mathbb{R}^N} E(x,\theta)$ is not unique.

(I) Implicit Differentiation (ID)

- The optimality condition is $\nabla_x E(x, \theta) = 0$.
- This implicitly defines $x^*(\theta)$ (**implicit function theorem**).
- Let (x^*, θ) be such that $\nabla_x E(x^*, \theta) = 0$, then, if [...] we have

$$\frac{\partial x^*}{\partial \theta}(\theta) = -\left(\frac{\partial^2 E}{\partial x^2}(x^*, \theta)\right)^{-1} \frac{\partial^2 E}{\partial \theta \partial x}(x^*, \theta)\,.$$

- The optimality condition is $\nabla_x E(x, \theta) = 0$.
- This implicitly defines $x^*(\theta)$ (**implicit function theorem**).
- Let (x^*, θ) be such that $\nabla_x E(x^*, \theta) = 0$, then, if [...] we have

$$\frac{\partial x^*}{\partial \theta}(\theta) = -\Big(\frac{\partial^2 E}{\partial x^2}(x^*, \theta)\Big)^{-1} \frac{\partial^2 E}{\partial \theta \partial x}(x^*, \theta)\,.$$

Disadvantages:

- Requires twice differentiability of E.
- Requires several approximations: x^* and $(\frac{\partial^2 E}{\partial x^2})^{-1}$ (solve linear system).
- Unstable for badly conditioned $\frac{\partial^2 E}{\partial x^2}$.
- Requires estimation (and storing) $\frac{\partial^2 E}{\partial x^2}$.

(I) Unrolling / Automatic Differentiation (AD)

- Approximate by a fixed $n \in \mathbb{N}$: (where $x^{(0)}$ is some fixed initialization)

$$x^*(\theta) \approx x^{(n+1)} := \mathcal{A}^{(n+1)}(x^{(0)}, \theta) = \mathcal{A} \circ \ldots \circ \mathcal{A}(x^{(0)}, \theta)$$

- Evaluate the chain rule by reverse mode AD (backpropagation):

$$\frac{\partial \mathcal{A}^{(n+1)}}{\partial \theta}(x^{(0)}, \theta) = \frac{\partial \mathcal{A}}{\partial x}(x^{(n)}, \theta)\frac{\partial \mathcal{A}^{(n)}}{\partial \theta}(x^{(0)}, \theta) + \frac{\partial \mathcal{A}}{\partial \theta}(x^{(n)}, \theta) = \ldots \text{unroll} \ldots$$

(I) Unrolling / Automatic Differentiation (AD)

- Approximate by a fixed $n \in \mathbb{N}$: (where $x^{(0)}$ is some fixed initialization)

$$x^*(\theta) \approx x^{(n+1)} := \mathcal{A}^{(n+1)}(x^{(0)}, \theta) = \mathcal{A} \circ \ldots \circ \mathcal{A}(x^{(0)}, \theta)$$

- **Algorithmic formulation:** Set $J_\theta^{(1)} := \frac{\partial \mathcal{A}}{\partial \theta}(x^{(0)}, \theta)$ and for $n = 1, 2, \ldots$:

$$J_\theta^{(n+1)} = \frac{\partial \mathcal{A}}{\partial x}(x^{(n)}, \theta) J_\theta^{(n)} + \frac{\partial \mathcal{A}}{\partial \theta}(x^{(n)}, \theta)$$

- Approximate by a fixed $n \in \mathbb{N}$: (where $x^{(0)}$ is some fixed initialization)

$$x^*(\theta) \approx x^{(n+1)} := \mathcal{A}^{(n+1)}(x^{(0)}, \theta) = \mathcal{A} \circ \ldots \circ \mathcal{A}(x^{(0)}, \theta)$$

- **Algorithmic formulation:** Set $J_\theta^{(1)} := \frac{\partial \mathcal{A}}{\partial \theta}(x^{(0)}, \theta)$ and for $n = 1, 2, \ldots$:

$$J_\theta^{(n+1)} = \frac{\partial \mathcal{A}}{\partial x}(x^{(n)}, \theta) J_\theta^{(n)} + \frac{\partial \mathcal{A}}{\partial \theta}(x^{(n)}, \theta)$$

Advantages:

- Output is **unambiguous** (unique), even in case $\operatorname{argmin} E$ is multi-valued.
- After the algorithm $\mathcal{A}$ and $n \in \mathbb{N}$ are fixed, the approach is exact.
- All iterations depend on the same parameter $\rightsquigarrow$ possibly truncate backprop.
- Easy implementation using standard AD packages.
- For E strongly convex, we have convergence rates for

$$J_\theta^{(n+1)} \xrightarrow{n \to \infty} \frac{\partial x^*}{\partial \theta}(\theta)$$

Accelerated convergence for accelerated algorithms! [Mehmood, O. '20]

Disadvantages: Store all intermediate iterates (in reverse mode).

(I) Fixed Point Automatic Differentiation (FPAD)

- Replace $\arg\min E(x, \theta)$ by a fixed point equation:

$$x^*(\theta) = \mathcal{A}(x^*(\theta), \theta)$$

- Imitate $n \in \mathbb{N}$ iterations of $\mathcal{A}$ starting at $x^{(0)} := x^*$:

$$\mathcal{A}^{(n+1)}(x^*, \theta) = \mathcal{A} \circ \ldots \circ \mathcal{A}(x^*, \theta)$$

(I) Fixed Point Automatic Differentiation (FPAD)

- Replace $\arg\min E(x,\theta)$ by a fixed point equation:

$$x^*(\theta) = \mathcal{A}(x^*(\theta), \theta)$$

- Imitate $n \in \mathbb{N}$ iterations of $\mathcal{A}$ starting at $x^{(0)} := x^*$:

$$\mathcal{A}^{(n+1)}(x^*, \theta) = \mathcal{A} \circ \ldots \circ \mathcal{A}(x^*, \theta)$$

- Evaluate the chain rule by reverse mode AD (backpropagation): For $n = 1, 2, \ldots$:

$$\hat{J}_\theta^{(n+1)} = \frac{\partial \mathcal{A}}{\partial x}(x^*, \theta)\hat{J}_\theta^{(n)} + \frac{\partial \mathcal{A}}{\partial \theta}(x^*, \theta)$$

(I) Fixed Point Automatic Differentiation (FPAD)

- Replace $\arg\min E(x,\theta)$ by a fixed point equation:

$$x^*(\theta) = \mathcal{A}(x^*(\theta), \theta)$$

- Imitate $n \in \mathbb{N}$ iterations of $\mathcal{A}$ starting at $x^{(0)} := x^*$:

$$\mathcal{A}^{(n+1)}(x^*, \theta) = \mathcal{A} \circ \ldots \circ \mathcal{A}(x^*, \theta)$$

- Evaluate the chain rule by reverse mode AD (backpropagation): For $n = 1, 2, \ldots$:

$$\hat{J}_\theta^{(n+1)} = \frac{\partial \mathcal{A}}{\partial x}(x^*, \theta)\hat{J}_\theta^{(n)} + \frac{\partial \mathcal{A}}{\partial \theta}(x^*, \theta) \qquad \xrightarrow{n\to\infty} \qquad \frac{\partial x^*}{\partial \theta}(\theta)$$

$\rightsquigarrow$ **faster convergence rate of derivative sequence** [Mehmood, O. ’20].
(known in AD community [Gilbert ’92, Christianson ’94];
connection to ID via Von Neumann series.)

(I) Fixed Point Automatic Differentiation (FPAD)

- Replace $\arg\min E(x,\theta)$ by a fixed point equation:

$$x^*(\theta) = \mathcal{A}(x^*(\theta), \theta)$$

- Imitate $n \in \mathbb{N}$ iterations of $\mathcal{A}$ starting at $x^{(0)} := x^*$:

$$\mathcal{A}^{(n+1)}(x^*, \theta) = \mathcal{A} \circ \ldots \circ \mathcal{A}(x^*, \theta)$$

- Evaluate the chain rule by reverse mode AD (backpropagation): For $n = 1, 2, \ldots$:

$$\hat{J}_\theta^{(n+1)} = \frac{\partial \mathcal{A}}{\partial x}(x^*, \theta)\hat{J}_\theta^{(n)} + \frac{\partial \mathcal{A}}{\partial \theta}(x^*, \theta) \qquad \xrightarrow{n\to\infty} \qquad \frac{\partial x^*}{\partial \theta}(\theta)$$

  $\leadsto$ **faster convergence rate of derivative sequence** [Mehmood, O. '20].
  (known in AD community [Gilbert '92, Christianson '94];
  connection to ID via Von Neumann series.)

- Equivalent to **AD with intermediate variables replaced by the optimum**: [O. et al. '16]

$$J_\theta^{(n+1)} = \frac{\partial \mathcal{A}}{\partial x}(x^{(n)}, \theta)J_\theta^{(n)} + \frac{\partial \mathcal{A}}{\partial \theta}(x^{(n)}, \theta)$$

(I) Fixed Point Automatic Differentiation (FPAD)

- Replace $\arg\min E(x,\theta)$ by a fixed point equation:
$$x^*(\theta) = \mathcal{A}(x^*(\theta), \theta)$$

- Imitate $n \in \mathbb{N}$ iterations of $\mathcal{A}$ starting at $x^{(0)} := x^*$:
$$\mathcal{A}^{(n+1)}(x^*, \theta) = \mathcal{A} \circ \ldots \circ \mathcal{A}(x^*, \theta)$$

- Evaluate the chain rule by reverse mode AD (backpropagation): For $n = 1, 2, \ldots$:
$$\hat{J}_\theta^{(n+1)} = \frac{\partial \mathcal{A}}{\partial x}(x^*, \theta)\hat{J}_\theta^{(n)} + \frac{\partial \mathcal{A}}{\partial \theta}(x^*, \theta) \qquad \xrightarrow{n\to\infty} \qquad \frac{\partial x^*}{\partial \theta}(\theta)$$

  $\rightsquigarrow$ **faster convergence rate of derivative sequence** [Mehmood, O. '20].

  (known in AD community [Gilbert '92, Christianson '94];
  connection to ID via Von Neumann series.)

- Equivalent to **AD with intermediate variables replaced by the optimum**: [O. et al. '16]
$$J_\theta^{(n+1)} = \frac{\partial \mathcal{A}}{\partial x}(x^{(n)}, \theta)J_\theta^{(n)} + \frac{\partial \mathcal{A}}{\partial \theta}(x^{(n)}, \theta)$$

Key advantages:

- Requires to store only x^*.
- Easy implementation using standard AD packages.

In the following: $x^*(\theta) \in \arg\min_{x\in\mathbb{R}^N} E(x,\theta)$ **(E structured non-smooth)**

Goal: Compute $\dfrac{\partial x^*}{\partial \theta}(\theta)$ **!?**

In the following: $x^*(\theta) \in \arg\min_{x\in\mathbb{R}^N} E(x,\theta)$ **(E structured non-smooth)**

Goal: Compute $\frac{\partial x^*}{\partial \theta}(\theta)$ **!?**

Strategies that compute (classic) gradients:

- Strategies above after smoothing E (ID, AD, FPAD):

$\rightsquigarrow$ often instable or requires significant smoothing
$\rightsquigarrow$ no approximation bounds

In the following: $x^*(\theta) \in \arg\min_{x\in\mathbb{R}^N} E(x,\theta)$ **(E structured non-smooth)**

Goal: Compute $\frac{\partial x^*}{\partial \theta}(\theta)$ **!?**

Strategies that compute (classic) gradients:

- Strategies above after smoothing E (ID, AD, FPAD):

$\rightsquigarrow$ often instable or requires significant smoothing
$\rightsquigarrow$ no approximation bounds

- **Unrolling a "smooth algorithm"** that solves the non-smooth problem. [O. et al. '16]
  - $\mathcal{A}^{(n+1)}(x^{(0)},\theta) \to x^*(\theta)$ for $n \to \infty$, where $\mathcal{A}$ is a smooth mapping.
  - **Idea:** Assert that iterates lie in interior of constraint set. *(e.g. Bregman Proximal Gradient Method)*

$\rightsquigarrow$ **Smoothing is controlled by the number of iterations** $n \in \mathbb{N}$.
$\rightsquigarrow$ Convergence of derivative sequence was **not** studied.
Limit requires generalized notion of derivatives.

In the following: $x^*(\theta) \in \arg\min_{x\in\mathbb{R}^N} E(x,\theta)$ **(E structured non-smooth)**

Goal: Compute $\dfrac{\partial x^*}{\partial\theta}(\theta)$ **!?**

Strategies that compute (classic) gradients:

- Strategies above after smoothing E (ID, AD, FPAD):

$\leadsto$ often instable or requires significant smoothing
$\leadsto$ no approximation bounds

- **Unrolling a "smooth algorithm"** that solves the non-smooth problem. [O. et al. '16]
  - $\mathcal{A}^{(n+1)}(x^{(0)},\theta) \to x^*(\theta)$ for $n\to\infty$, where $\mathcal{A}$ is a smooth mapping.
  - **Idea:** Assert that iterates lie in interior of constraint set. *(e.g. Bregman Proximal Gradient Method)*

$\leadsto$ **Smoothing is controlled by the number of iterations** $n \in \mathbb{N}$.
$\leadsto$ Convergence of derivative sequence was **not** studied. Limit requires generalized notion of derivatives.

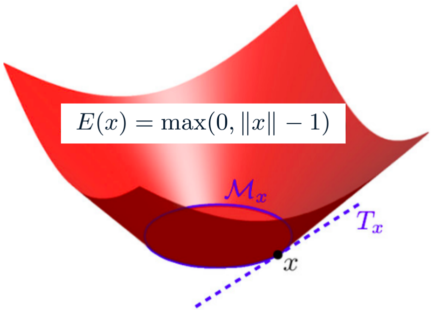
  - **FPAD variant** requires x^* to lie in the interior of the constraint set.

(II) Differentiation Strategies for Partly Smooth Functions

Definition of **partial smoothness** for convex E from [Liang, Fadili, Peyré '14] (original [Lewis '02]):

E with $\partial E(x) \neq \emptyset$ is **partly smooth at x relative to $\mathcal{M} \ni x$**, if

- (Smoothness) $\mathcal{M}$ is a C^2-Manifold around x and $E|_{\mathcal{M}} \in C^2$,
- (Sharpness) the normal space $\mathcal{N}_{\mathcal{M}}$ is $\mathrm{par}(\partial E(x))$, and
- (Continuity) ∂E is continuous at x relative to $\mathcal{M}$.



from [G. Peyré, talk "Low Complexity Regularization of Inverse Problems", 2014]

Examples: ℓ_1-norm, $\ell_{2,1}$-norm, ℓ_∞-norm, nuclear norm, TV-norm, ...

(II) Implicit Differentiation (ID) under Partial Smoothness

Consider: $x^*(\theta) \in \arg\min_{x\in\mathbb{R}^N} E(x,\theta)$

Theorem: [ID under Partial Smoothness [Lewis '02], [Vaiter et al. '17]]

Assumptions:

- E is partly smooth relative to $\mathcal{M} \times \Omega$ where $\Omega \subset \mathbb{R}^P$ is open, and
- for some (x^*, θ^*), the following **restricted positive definiteness** holds

$$\forall v \in \mathcal{T}_{x^*}\mathcal{M}\colon \qquad \langle v, \nabla^2_{\mathcal{M}} E(x^*,\theta^*)v\rangle > 0$$

and **non-degeneracy** holds

$$0 \in \operatorname{ri}\left(\partial_x E(x^*,\theta^*)\right).$$

Then the **manifold version of the Implicit Differentiation (ID) holds**:

- There exists a neighborhood Θ of θ^* and a C^1 mapping $\psi\colon \Theta \to \mathcal{M}$ such that
- $\Theta \ni \theta \mapsto \psi(\theta) = \operatorname{argmin}_x E(x,\theta)$, and
- $\Theta \ni \theta \mapsto \dfrac{\partial\psi}{\partial\theta}(\theta) = -\nabla^2_{\mathcal{M}} E(\psi(\theta),\theta)^\dagger \dfrac{\partial}{\partial\theta}\nabla_{\mathcal{M}} E(\psi(\theta),\theta).$

(II) Unrolling / Automatic Differentiation (AD) under Partial Smoothness

- **Idea:** Many algorithms have the **finite identification property** (see papers by [J. Liang]):

$$\exists n_0 \in \mathbb{N}: \quad x^{(n)} \in \mathcal{M} \text{ for all } n \geq n_0 \,,$$

  hence the update mapping $\mathcal{A}$ becomes smooth eventually.

  Examples: Forward–backward Splitting (Proximal Gradient Descent), FISTA, ...

- **Idea:** Many algorithms have the **finite identification property** (see papers by [J. Liang]):

$$\exists n_0 \in \mathbb{N}: \quad x^{(n)} \in \mathcal{M} \text{ for all } n \geq n_0 \,,$$

hence the update mapping $\mathcal{A}$ becomes smooth eventually.

Examples: Forward–backward Splitting (Proximal Gradient Descent), FISTA, ...

Theorem: [AD for Forward–backward Splitting [Liang et al. '14], [Mehmood, O. '22]]

Assumptions:

- Partial smoothness, restricted positive definiteness, the non-degeneracy assumption, and convergence of the iterates.

Then,

- the rate of convergence for $(x^{(n)})_{n\in\mathbb{N}}$ is actually **locally linear**.
- If, additionally, $x^{(0)}$ is close enough to x^*, then

$$\frac{\partial \mathcal{A}^{(n+1)}}{\partial \theta}(x^{(0)}, \theta) =: J_\theta^{(n)} \xrightarrow{n\to\infty} \frac{\partial \psi}{\partial \theta}(\theta)\,.$$

$\rightsquigarrow$ **Crucial issue:** Derivatives are not defined for $x^{(n)} \notin \mathcal{M}$.

- Linear convergence, if all involved derivatives are locally Lipschitz.

Remedy of Differentiation Issue:

- Since $x^* \in \mathcal{M}$, by definition, all iteration mappings in FPAD are well defined.

(II) Fixed Point Automatic Differentiation (FPAD)

Remedy of Differentiation Issue:

- Since $x^* \in \mathcal{M}$, by definition, all iteration mappings in FPAD are well defined.

Theorem: [Convergence of FPAD for Forward–backward Splitting [Mehmood, O. '22]]

Assumptions:

- Partial smoothness, restricted positive definiteness and the non-degeneracy.

Then,

- the derivative sequence converges linearly

$$\frac{\partial \mathcal{A}^{(n+1)}}{\partial \theta}(x^*, \theta) =: \hat{J}_\theta^{(n)} \xrightarrow{n \to \infty} \frac{\partial \psi}{\partial \theta}(\theta)\,.$$

(III) Non-Smooth Setting: Strategies for Generalized Differentiation

In the following: $x^*(\theta) \in \arg\min_{x\in\mathbb{R}^N} E(x,\theta)$ **(E non-smooth)**

Goal: Compute $\frac{\partial x^*}{\partial \theta}(\theta)$ **!?**

In the following: $x^*(\theta) \in \arg\min_{x \in \mathbb{R}^N} E(x, \theta)$ **(E non-smooth)**

Goal: Compute $\dfrac{\partial x^*}{\partial \theta}(\theta)$ **!?**

Strategies:

- Weak differentiation of iterative algorithms. [Deledalle et al. '14]
  (using Rademacher Theorem; show update map is Lipschitz)

$\rightsquigarrow$ yields only weak derivatives (not defined pointwise)
$\rightsquigarrow$ convergence of derivative sequence unknown

- Using generalized non-smooth derivatives. [Bolte et al. '21]

  $\rightsquigarrow$ *Some details on next slides.*

Definition: [Bolte, Pauwels '19]

A locally Lipschitz function $F\colon \mathbb{R}^N \to \mathbb{R}$ is **path differentiable**, if

- there exists a nowhere empty-valued mapping $D_F\colon \mathbb{R}^N \rightrightarrows \mathbb{R}^N$
- that has a closed graph and is locally bounded, such that
- for any absolutely continuous curve $\gamma\colon [0,1] \to \mathbb{R}^N$

$$\frac{d}{dt} F \circ \gamma(t) = A\dot{\gamma}(t)\,, \quad \forall A \in D_F(\gamma(t))\,, \quad \text{a.e. for } t \in [0,1]\,.$$

- Call D_F **conservative Jacobian** of F.

Construction idea:

$\rightsquigarrow$ By definition, **path differentiable functions admit a chain rule**.

$\rightsquigarrow$ Formalization of backpropagation used in AD packages for neural networks.

$\rightsquigarrow$ Straightforward generalization to locally Lipschitz mappings.

Theorem: [Implicit Path Differentiation (ID) [Bolte et al. '21], [Bolte, Pauwels, Silveti–Falls '23]]

Assumptions:

- $F\colon \mathbb{R}^N \times \mathbb{R}^P \to \mathbb{R}^N$ is path differentiable.
- (x^*, θ^*) satisfies the optimality condition
$$F(x^*, \theta^*) = 0\,.$$
- $D_F(x^*, \theta^*)$ is convex and $\forall [A\,, B] \in D_F(x^*, \theta^*)$, we have that A is invertible.

Then the **path differentiable version of the Implicit Differentiation (ID) holds**:

- $\exists \Theta$ neighborhood of θ^* and path differentiable $\Psi\colon \Theta \to \mathbb{R}^N$ such that
$$\forall \theta \in \Theta\colon \quad F(\Psi(\theta), \theta) = 0\,,$$
- and $\Theta \ni \theta \mapsto D_\Psi(\theta) = \{-A^{-1}B \,|\, [A\,, B] \in D_F(\Psi(\theta), \theta)\}$.

Key Disadvantage:

- Invertibility condition for all A above; Sufficient condition by strong monotonicity of F.

(III) Unrolling / Automatic Differentiation (AD)

Idea:

- Compositions of path differentiable functions are path differentiable.
- Path differentiable functions admit a chain rule.

Unrolling:

- Evaluate the chain rule. Set $\mathcal{J}_\theta^{(1)} := \{J_\theta^{(1)} \mid [A_x^{(1)}, J_\theta^{(1)}] = D_\mathcal{A}(x^{(0)}, \theta)\}$ and for $n = 1, 2, \ldots$:

$$\mathcal{J}_\theta^{(n+1)} = \{A_x^{(n)} J_\theta^{(n)} + A_\theta^{(n)} \mid [A_x^{(n)}, A_\theta^{(n)}] \in D_\mathcal{A}(x^{(n)}, \theta), J_\theta^{(n)} \in \mathcal{J}_\theta^{(n)}\}\,.$$

Theorem: [Convergence of AD under path differentiability [Bolte, Pauwels, Vaiter '22]]

Assumption:

- $\mathcal{A}$ is locally Lipschitz with unique fixed point $x^*(\theta)$, path differentiable in (x, θ),
- and $\exists \rho \in [0, 1) \colon \forall (x, \theta)$ and all $[A, B] \in D_\mathcal{A}(x, \theta)$, we have $\|A\| \le \rho$.

Then:

- The gap between $\mathcal{J}_\theta^{(n+1)}$ and $\mathrm{fix}(D_\mathcal{A}(x^*(\theta), \theta))$ vanishes as $n \to \infty$.
- Under additional assumptions on a Lipschitz gradient selection structure, the convergence is linear.

Conclusion

Bi-level optimization / parameter learning

$$
\begin{aligned}
&\min_{\theta\in\mathbb{R}^P} \quad \mathcal{L}(x^*(\theta),\theta) && \text{(upper level)}\\
&s.t. \quad x^*(\theta)\in \arg\min_{x\in\mathbb{R}^N} E(x,\theta) && \text{(lower level)}
\end{aligned}
$$

Goal: Compute $\dfrac{\partial x^*}{\partial \theta}(\theta)$ **!?**

Outline:	E	**type of bilevel algorithm**
(I) smooth setting	smooth	gradient
(II) partly smooth setting	structured non-smooth	gradient
(III) non-smooth setting	non-smooth	generalized derivative

For each part:

- Implicit differentiation
- Automatic Differentiation / Backpropagation
- Fixed Point Automatic Differentiation
- Generalized derivatives of $x^*(\theta)$ requires algorithms for non-smooth bi-level opt.