Towards Differentiation of Solution Mappings of Non-smooth Optimization Problems

Peter Ochs

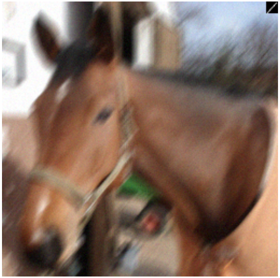
Mathematical Optimization Group
University of Tübingen

— 06.09.2021 —

Inverse Problem:

forward model $\mathcal{A}$

inverse problem



clean image $\boldsymbol{h}$

observed image $\boldsymbol{f}$

- The degradation process is modeled as follows

$$\boldsymbol{f} \approx \mathcal{A}(\boldsymbol{h}) + \mathcal{N} \qquad \mathcal{N}\text{: additive pixel-wise noise}$$

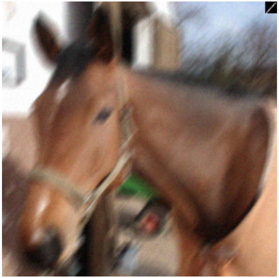
- **Goal: Reconstruct h**

Image Deblurring: of an image f (using Total Variation)

$$\min_{\boldsymbol{u}} \frac{1}{2}\|\mathcal{A}\boldsymbol{u} - \boldsymbol{f}\|^2 + \lambda\|\mathcal{D}\boldsymbol{u}\|_{2,1}$$

- $\mathcal{A}$ convolution / blurr operator; $\mathcal{D}$ discrete derivative operator

clean image



blurry image

reconstruction

Regularization weight has a huge impact on the result!

Variational Problems with Regularization

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

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

How to find the best regularization weight λ?

- hand-tuning
- grid search

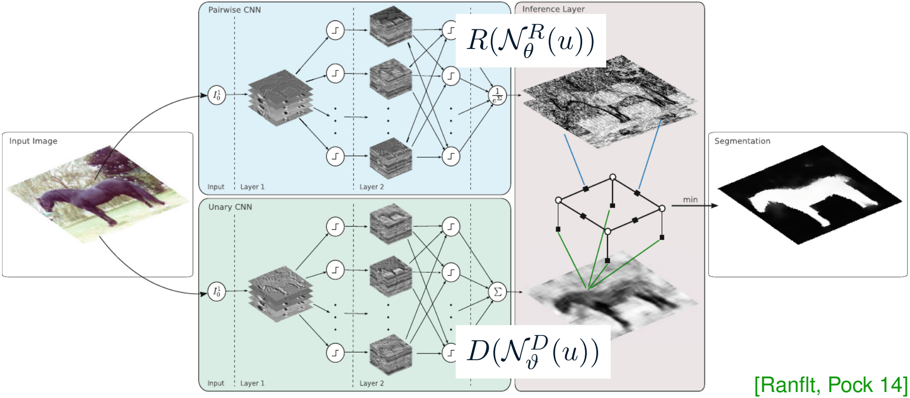
What about learning the whole regularization term?

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

- 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.)

Deep Neural Network with Inference Layer/Variational Model:



[Ranflt, Pock 14]

$\rightsquigarrow$ **Resurge of variational models as layers in deep learning.**

What about learning the whole regularization term?

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

- 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)

Bilevel optimization / parameter Learning

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

- $\theta \in \mathbb{R}^P$: 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

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

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

- $\theta \in \mathbb{R}^P$: 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 $\nabla_\theta \mathcal{L}(x^*(\theta^{(k)}),\theta^{(k)})$.

Computation of $\nabla_\theta \mathcal{L}(x^*(\theta^{(k)}), \theta^{(k)})$.

- If loss function $\mathcal{L}$ and solution mapping $x^*(\theta)$ are differentiable:

$$\boldsymbol{\nabla}_\theta \mathcal{L}(x^*(\theta^{(k)}), \theta^{(k)}) = \left(\frac{\partial x^*}{\partial \theta}(\theta^{(k)}) \right)^\top \nabla_x \mathcal{L}(x^*, \theta^{(k)}) + \nabla_\theta \mathcal{L}(x^*, \theta^{(k)})$$

- Actually, we are interested in the **directional derivative of** $x^*(\theta)$ **at** $\theta^{(k)}$ **in direction** $v^{(k)} := \nabla_x \mathcal{L}(x^*, \theta^{(k)})$:

$$\left(\frac{\partial x^*}{\partial \theta}(\theta^{(k)}) \right)^\top v^{(k)}$$

Computation of $\nabla_\theta \mathcal{L}(x^*(\theta^{(k)}), \theta^{(k)})$.

- If loss function $\mathcal{L}$ and solution mapping $x^*(\theta)$ are differentiable:

$$\nabla_\theta \mathcal{L}(x^*(\theta^{(k)}), \theta^{(k)}) = \left(\frac{\partial x^*}{\partial \theta}(\theta^{(k)}) \right)^\top \nabla_x \mathcal{L}(x^*, \theta^{(k)}) + \nabla_\theta \mathcal{L}(x^*, \theta^{(k)})$$

- Actually, we are interested in the **directional derivative of** $x^*(\theta)$ **at** $\theta^{(k)}$ **in direction** $v^{(k)} := \nabla_x \mathcal{L}(x^*, \theta^{(k)})$:

$$\left(\frac{\partial x^*}{\partial \theta}(\theta^{(k)}) \right)^\top v^{(k)}$$

$\rightsquigarrow$ **Study $\frac{\partial x^*}{\partial \theta}$ or, more generally,**

sensitivity of the solution mapping.

Strategies for differentiation:

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

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

Strategies:

- Implicit differentiation. (requires E strictly convex)
- Unrolling an algorithm. *(use automatic differentiation)*
- Implicit differentiation or unrolling of a fixed point equation. [O. et al. '16]

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.

Implicit Differentiation: (widely used approach; constraints via KKT)

- 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

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

Disadvantages: (reasons why we avoid this approach)

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

Unrolling an algorithm: $\mathcal{A}^{(n+1)}(x^{(0)},\theta) \to x^*(\theta)$ for $n \to \infty$

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

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

$\rightsquigarrow$ derivative by reverse mode AD (backpropagation)

- **Advantages:**
  - Output is **unambiguous** (unique), also if $\operatorname{argmin} E$ is multi-valued.
  - After the algorithm $\mathcal{A}$ is fixed, the approach is exact.
  - All iterations depend on the same parameter $\rightsquigarrow$ truncate backprop.
  - For E strongly convex, we have convergence rates for

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

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

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

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

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

- Implicit function theorem:

$$\frac{\partial x^*}{\partial \theta} = \left(I - \frac{\partial \mathcal{A}}{\partial x^*} \right)^{-1} \frac{\partial \mathcal{A}}{\partial \theta}$$

- Using

$$\left(I - \frac{\partial \mathcal{A}}{\partial x^*} \right)^{-1} \overset{\|\frac{\partial \mathcal{A}}{\partial x^*}\|<1}{=} \sum_{n=0}^{\infty} \left(\frac{\partial \mathcal{A}}{\partial x^*} \right)^n \frac{\partial \mathcal{A}}{\partial \theta} \approx \sum_{n=0}^{n_0} \left(\frac{\partial \mathcal{A}}{\partial x^*} \right)^n \frac{\partial \mathcal{A}}{\partial \theta}$$

  we obtain **AD with intermediate variables replaced by the optimum.** [O. et al. '16]

  $\rightsquigarrow$ **accelerates convergence rate** [Mehmood, O. '20].

- Requires to store only x^*.

Strategies for differentiation if E is non-smooth:

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

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

Strategies:

- Strategies above after smoothing E.
  - $\rightsquigarrow$ often instable or requires significant smoothing
  - $\rightsquigarrow$ no approximation bounds
- Weak differentiation of iterative algorithms. [Deledalle et al. '14]
  (using Rademacher Theorem; show update map is Lipschitz)
  - $\rightsquigarrow$ yields only weak derivatives
  - $\rightsquigarrow$ requires "Subgradient Descent" methods for bilevel problem
  - $\rightsquigarrow$ convergence of derivative sequence unknown

Strategies for differentiation if E is non-smooth: (continued)

- Unrolling a "smooth algorithm" that solves the non-smooth problem. [O. et al. '16] (*see next slides*)
  - $\rightsquigarrow$ E is non-smooth,
  - $\rightsquigarrow$ $\mathcal{A}^{(n+1)}(x^{(0)}, \theta) \to x^*(\theta)$ for $n \to \infty$,
  - $\rightsquigarrow$ update mapping $\mathcal{A}\colon \mathbb{R}^N \times \mathbb{R}^P \to \mathbb{R}^N$ is smooth.
- Implicit Differentiation of fixed point equations. [O. et al. '16] ("unrolling a smooth algorithm"-strategy but for fixed-point equation)
- Implicit Differentiation for partly smooth functions. [Vaiter et al. 16, Riis '19]
- Unrolling in the partial smoothness framework. [Riis, PhD Thesis '19]

Unrolling a "smooth algorithm":

- Key are Bregman distances D_ψ.
- Update mapping $\mathcal{B}_\psi$ of **Bregman Proximal Gradient Method** solves

$$\mathcal{B}_\psi(c(\theta)) := \operatorname*{argmin}_x \; g(x) + \langle c(\theta), x\rangle + \psi(x)$$

- c constant vector (depends on current iterate and θ);
  g is possibly non-smooth (e.g. indicator function)
- **Bregman Proximal Gradient** or **Bregman Primal Dual Algorithm**.
- idea similar to barrier approach $\rightsquigarrow$ all iterates lie in interior

Example 1: **(non-negativity constraint)**

- $X = [x \geq 0]$ and $g = \delta_X$
- entropy function $\psi(x) = x \log(x)$
- **Bregman Proximal mapping:**

$$\mathcal{B}_\psi(c(\theta)) = \exp(-c(\theta) - 1)$$

Application: **non-negative least squares problem.**

Example 2: (simplex constraint)

- Minimize $f(x)$ over the unit simplex in $\mathbb{R}^N$

$$\left\{x \in \mathbb{R}^N \middle| \sum_{i=1}^N x_i = 1 \text{ and } x_i \geq 0\right\}.$$

- Use entropy $\psi(x) = \sum_{i=1}^N x_i \log(x_i)$ to drop the non-negativity constraint.

- **Bregman proximal update mapping**:

$$(\mathcal{B}_\psi(c))_i = \frac{\exp(-c_i - 1)}{\sum_{j=1}^N \exp(-c_j - 1)} \quad \text{for } i = 1, \ldots, N\,.$$

Application: Multi-label segmentation problem or **Matrix games.**

Example 3: **(box constraint)**

- Use $\psi(x) = \frac{1}{2}((x+1)\log(x+1) + (1-x)\log(1-x))\,.$
- **Bregman proximal update mapping:**

$$\mathcal{B}_{\psi}(c) = \frac{\exp(-2c) - 1}{\exp(-2c) + 1}\,.$$

These linear functions are important thanks to convex duality.

- For instance the TV-norm can be represented as

$$\|\mathcal{D}x\|_1 = \max_y \langle \mathcal{D}x, y\rangle + \underbrace{\delta_{[-1\leq y\leq 1]}(y)}_{\text{box constraint}}\,,$$

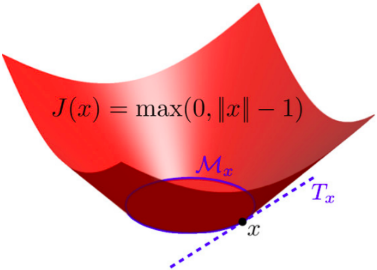
which can be employed in **Bregman Primal–Dual Algorithms**.

Differentiation Strategies for partly smooth functions [Lewis '03]

(**Definition** here in convex setting taken from [Liang, Fadili, Peyré '14])

$J \in \Gamma_0(\mathbb{R}^N)$ and $\partial J(x) \neq \emptyset$. We call J **partly smooth at x relative to $\mathcal{M} \ni x$**, if the following conditions are satisfied:

- (Smoothness) $\mathcal{M}$ is a C^2-Manifold around x and $J|_{\mathcal{M}} \in C^2$
- (Sharpness) The tangent space $\mathcal{T}_{\mathcal{M}}$ is $\mathrm{par}(\partial J(x))^{\perp}$.
- (Continuity) ∂J is continuous at x relative to $\mathcal{M}$.



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

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

Implicit differentiation under partial smoothness

(from [Vaiter et al. '16])

$$E(x,\theta) = F(x,\theta) + J(x)$$

Assumptions:

- F block-wise C^2
- $\bar{\theta} \notin \mathcal{H}$ (certain transition space of measure 0; non-degeneracy ass.)
- $J \in \Gamma_0$ is partly smooth at solution $x^*(\bar{\theta})$ relative to $\mathcal{M}$
- restricted positive definiteness holds at $x^*(\bar{\theta})$.

Then:

- there exists open $\mathcal{V} \ni \bar{\theta}$ and a mapping $\hat{x}\colon \mathcal{V} \to \mathcal{M}$ such that
  1. For all $\theta \in \mathcal{V}$, $\hat{x}(\theta)$ is a solution that coincides with x^* at $\bar{\theta}$
  2. $\hat{x} \in C^1(\mathcal{V})$ and for all $\theta \in \mathcal{V}$:

$$\frac{\partial \hat{x}}{\partial \theta}(\theta) = -\Big(\nabla^2_{\mathcal{M}} F(\hat{x}(\theta),\theta) + \nabla^2_{\mathcal{M}} J(\hat{x}(\theta))\Big)^{\dagger} \mathrm{P}_{T_{\hat{x}(\theta)}} \frac{\partial (\nabla F)}{\partial \theta}(\hat{x}(\theta),\theta)$$

Unrolling algorithms under partial smoothness (idea):

- E is non-smooth (but partly smooth)
- Solution lies on a smooth manifold $\mathcal{M}$ and is stable
- $\mathcal{A}^{(n+1)}(x^{(0)}, \theta) \to x^*(\theta)$ for $n \to \infty$
- requires non-degeneracy assumption for $x^*(\theta)$
- 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.

Differentiating the Value Function by using Convex Duality, AISTATS 2021 (different situation) [Mehmood, O. '21]

$$p(u) = \min_x f(x,u)$$

Compute the **derivative of the value function**:

$\nabla p(u)$ (p **can be smooth with** f **being smooth**)

- Requires f to be convex in x and u.
- Derivative is given by (convex duality; f^*: convex conjugate of f)

$$\partial p(u) = \operatorname*{argmax}_y \langle u, y\rangle - f^*(0,y)$$

⇝ **derivative by solving an optimiation problem** (dual problem).

⇝ **convergence rates of the derivative sequence** in situations where f is not strongly convex.

Conclusion

- Bilevel optimization as framework for parameter learning.
- Solve bilevel problem by gradient based algorithms.
- Strategies for computing the derivative of the solution mapping in smooth and non-smooth setup:
  - Implicit differentiation.
  - Weak differentiation of iterative algorithms.
  - Unrolling a “smooth algorithm” for a non-smooth problem.
  - Implicit Differentiation or unrolling of fixed point equations.
  - Differentiation under partial smoothness assumption.

Derivation:

$$
\begin{aligned}
\partial_\theta x^* &\approx \partial_\theta x^{(n+1)} \\
&= \partial_x x^{(n+1)} \partial_\theta x^{(n)} + \partial_\theta x^{(n+1)} \\
&= \partial_x x^{(n+1)} (\partial_x x^{(n)} \partial_\theta x^{(n-1)} + \partial_\theta x^{(n)}) + \partial_\theta x^{(n+1)} \\
&\ \vdots \\
&= \sum_{k=0}^{n+1} \Big(\prod_{j=k}^{n} \partial_x x^{(j+1)} \Big) \partial_\theta x^{(k)} \\
&= \sum_{k=0}^{n+1} \Big(\prod_{j=k}^{n} \partial_x \mathcal{A}(x^{(j)}, \theta) \Big) \partial_\theta \mathcal{A}(x^{(k-1)}, \theta) \\
&\approx \sum_{k=0}^{n+1} \Big(\prod_{j=k}^{n} \partial_x \mathcal{A}(x^*, \theta) \Big) \partial_\theta \mathcal{A}(x^*, \theta) \\
&= \sum_{k=0}^{n+1} \Big(\partial_x \mathcal{A}(x^*, \theta) \Big)^k \partial_\theta \mathcal{A}(x^*, \theta) \approx \Big(I - \partial_x \mathcal{A}(x^*, \theta) \Big)^{-1} \partial_\theta \mathcal{A}(x^*, \theta)
\end{aligned}
$$