Derivatives of Solutions of Saddle-Point Problems

Antonin Chambolle

CEREMADE, CNRS, Univ. Paris-Dauphine / PSL, France

(joint with T. Pock)

One World Optimization Seminar, Vienna, 1st March 2021

Outline:

- Some bi-level problems involving total variation minimization;
- Homogenization: case of a loss depending on the value;
- More general losses: adjoint states;
- “Piggyback [style] algorithm” and results.

Related work

- A lot of literature on parameter learning for TV-regularization problems (for one parameter but also with varying parameters), usually for one image;
- Most papers focus on the continuous setting and then propose to solve an adjoint equation;
- Our work is more "discrete" and focuses on the algorithms (but of course, some similarities).

[Dong, Hintermüller, Rincon-Camacho 2010] ([Bredies et al 2013 for TGV]) [Kunisch-Pock 2013] [De Los Reyes, Schönlieb, Valkonen 2015], [Calatroni et al 2015-17] [Hintermüller-Rautenberg 17], [Hintermüller-et al 17], [Hintermüller-Papafistoros HNA 2019]

Starting point

Improve or learn discrete surface energies / total variations so that

- They are faithful, possibly precise approximations of the continuous T.V.;
- They behave "well" at the discrete level (isotropy, sharpness...)

0. Typical model:

Focus on problems of the form:

$$\min_{u=u^0\ \partial\Omega} \int_\Omega |Du| \quad \left(\text{or } + \int_\Omega |u-g|^2 dx \right)$$

where $u^0 \in \{0,1\}$, so that this is equivalent to finding sets E with lowest perimeter and boundary condition $\chi_E = u^0$. One expects to find (in general) sharp solutions $u \in \{0,1\}$ *a.e.*

Discretize: One minimizes in practice a convex problem of the form

$$\min_{u_i=u_i^0, i\in I^0} F_h(u_i) : u \in \mathbb{R}^{N(h)}$$

where $(u_i)_{i=1}^{N(h)}$ is supposed to be a discrete representation of u at scale $h>0$, and F_h approximates the total variation in some sense.

Typical model:

In practice, depending on the form of F_h, one can expect more or less “nice” or “precise” results (sharp, isotropic, or not...)
$\Rightarrow$ could one “learn” the “best” one (or a better one)?

Typical model:

Here the discrete problem has the form

$$\min_{u \in C_u} \sup_{w \in C_w} \langle w, Du \rangle$$

for D some discrete derivative, and where C_u and C_w are convex sets. Simpler versions include

$$\min_{u \in C_u} \sup_{w \in C_w} \langle w, Du \rangle + \frac{1}{2}\|u - g\|^2 \qquad (\text{“ROF”})$$

which is strongly convex wr u, or a “regularized” variant:

$$\min_{u \in C_u} \sup_{w \in C_w} \langle w, Du \rangle - \frac{\varepsilon}{2}\|w\|^2 + \frac{\varepsilon}{2}\|u\|^2$$

for $\varepsilon > 0$ a small parameter, which is strongly convex wr both u and w.

I. Easier case

Let us consider for a start an “easy” case. We will try to build an “*isotropic-ℓ_1*” discretization. Assume we are given a discrete $2D$ image $u_{i,j}$ on a square grid, we define a graph total variation as

$$\sum_{(i,j),(i',j')} \alpha_{(i,j),(i',j')}(u_{i',j'} - u_{i,j})^+$$

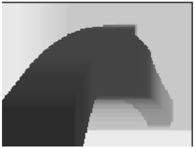
(here $x^+ = \max\{x, 0\}$). The simplest form would be:

$$\sum_{i,j} \alpha^+_{i+\frac{1}{2},j}(u_{i+1,j} - u_{i,j})^+ + \alpha^-_{i+\frac{1}{2},j}(u_{i,j} - u_{i+1,j})^+ \\ + \alpha^+_{i,j+\frac{1}{2}}(u_{i,j+1} - u_{i,j})^+ + \alpha^-_{i,j+\frac{1}{2}}(u_{i,j} - u_{i,j+1})^+$$

which involves only horizontal/vertical directions.

An easy case

Clearly, if all the α's are 1, this is an "ℓ_1" discretization of the total variation, which in a continuum limit would approximate the anisotropic functional $\int |\partial_1 u| + |\partial_2 u|$, and produces block artefacts. On the other hand, it is very easy and fast to optimize (graph cuts, or horizontal/vertical splitting...)



An easy case: homogenization

The isotropy can be improved by "homogenization". In practice, the idea is to use periodic oscillating weights $\alpha^{\pm}$ which produce, in the continuum limit, an "effective surface tension" given by an exact "cell formula", defined for $\nu \in \mathbb{R}^2$,

$$\phi(\nu) = \min_u \Bigg\{ \sum_{(i,j)\in Y} \alpha^+_{i+\frac{1}{2},j}(u_{i+1,j} - u_{i,j})^+ + \alpha^-_{i+\frac{1}{2},j}(u_{i,j} - u_{i+1,j})^+ \\ + \alpha^+_{i,j+\frac{1}{2}}(u_{i,j+1} - u_{i,j})^+ + \alpha^-_{i,j+\frac{1}{2}}(u_{i,j} - u_{i,j+1})^+ \; : \\ u_{i,j} - \nu \cdot \begin{pmatrix} i \\ j \end{pmatrix} \; Y\text{-periodic} \Bigg\}$$

where here Y is a periodicity cell of the form $\{1,\dots,n\} \times \{1,\dots,m\}$. (Typically, $m = n = 2, 3, 4\dots$)

An easy case: homogenization

... and one would be interested in solving:

$$\min_{(\alpha)} \mathcal{L}(\alpha) := \frac{1}{2} \sum_{i=1}^{k} |\phi(\nu_i) - 1|^2$$

where the "loss" $\mathcal{L}$ depends on α through the dependence of $\phi(\cdot)$ on α and ν_i are a set of given directions.
So one needs to estimate $\nabla_{(\alpha)}\phi(\nu_i)$, for each direction ν_i.

Derivative of the energy

In our case the minimal energy $\phi(\nu)$ can be found by solving a saddle-point problem:

$$\phi(\nu) = \min_{u \in C_i(\nu)} \sup_{w \in C_w} \langle D(\alpha)u, w \rangle$$

Derivative of the energy

In our case the minimal energy $\phi(\nu)$ can be found by solving a saddle-point problem:

$$\phi(\nu) = \min_{u \in C_i(\nu)} \sup_{w \in C_w} \langle D(\alpha)u, w\rangle - \frac{\varepsilon}{2}\|w\|^2 + \frac{\varepsilon}{2}\left\|u - \binom{i}{j}\cdot\nu\right\|^2$$

which we regularize in order to have a unique solution $(u(D), w(D))$ for a given discrete derivative operator D.

Derivative of the energy

Thanks to the regularization one easily sees that

- $D \mapsto (u(D), w(D))$ is continuous and
- $D \mapsto \phi(\nu) =: \mathcal{E}_\nu(D)$ is $C^{1,1}$.

Indeed:

$$\sup_{w \in C_w} \langle Du, w \rangle - \frac{\varepsilon}{2}\|w\|^2$$

- is convex with $(1/\varepsilon)$-Lipschitz gradient with respect to Du
- is convex with (C/ε)-Lipschitz gradient with respect to D in a neighborhood of D, for $C > \|u(D)\|^2$.
- so its $\inf_u$ has Hessian bounded from above.

Derivative of the energy

Thanks to the regularization one easily sees that

- $D \mapsto (u(D), w(D))$ is continuous and
- $D \mapsto \phi(\nu) =: \mathcal{E}_\nu(D)$ is $C^{1,1}$.

Indeed:

$$\sup_{w \in C_w} \langle Du, w \rangle - \frac{\varepsilon}{2} \|w\|^2$$

- is convex with $(1/\varepsilon)$-Lipschitz gradient with respect to Du
- is convex with (C/ε)-Lipschitz gradient with respect to D in a neighborhood of D, for $C > \|u(D)\|^2$.
- so its $\inf_u$ has Hessian bounded from above.
- symetrically (taking first $\inf_u$ then $\sup_w$) one gets a bound from below.

Derivative of the energy

Then, computing the differential is quite standard (one can for instance estimate $\mathcal{E}_\nu(D + tL)$, t small, from above and below using the optimal values u_t, w_t, and pass to the limit...) and one finds

$$\nabla_D \mathcal{E}_\nu(D) = w(D) \otimes u(D)$$

So here one just needs to solve (with some precision) the saddle point to evaluate the derivative from the optimal solutions (u, w). Then one can implement a gradient descent and optimize the main criterion $\mathcal{L}(\alpha)$.

Derivative of the energy

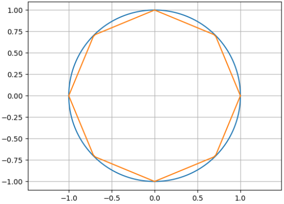
Then, computing the differential is quite standard (one can for instance estimate $\mathcal{E}_\nu(D + tL)$, t small, from above and below using the optimal values u_t, w_t, and pass to the limit...) and one finds

$$\nabla_D \mathcal{E}_\nu(D) = w(D) \otimes u(D)$$

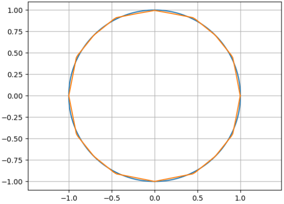
So here one just needs to solve (with some precision) the saddle point to evaluate the derivative from the optimal solutions (u, w). Then one can implement a gradient descent and optimize the main criterion $\mathcal{L}(\alpha)$.

Possible extension: smooth only wr u or w. Then the energy will still be either semi-concave or semi-convex and one can evaluate the (sub/super)gradient in the same way.

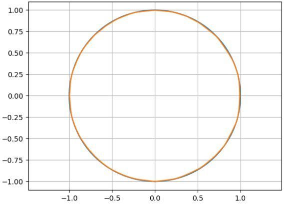
Application / Results



2×2 periodicity cell

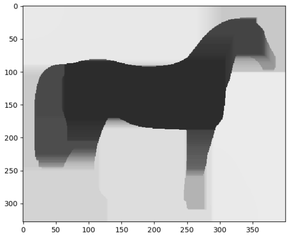


4×4 cell

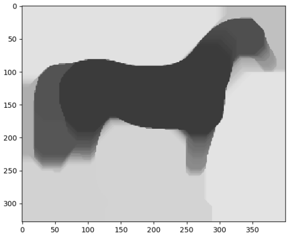


8×8

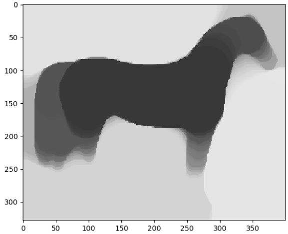
Application / Results



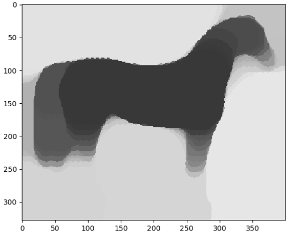
ℓ_1-TV



2×2

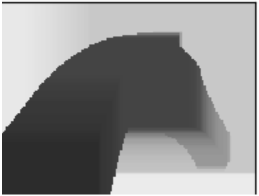


4×4

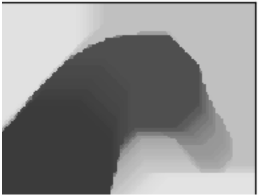


8×8

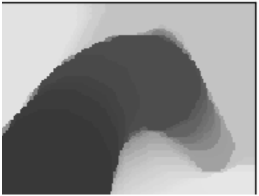
Application / Results



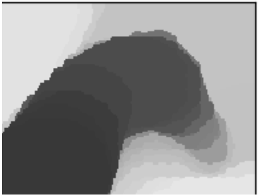
ℓ_1-TV



2×2

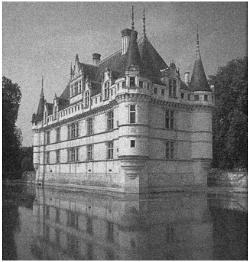


4×4



8×8

Application / Results: denoising



Noisy

ℓ_1-TV

2×2

8×8

Application / Results: denoising

Noisy

ℓ_1-TV

2×2

8×8

II. More general losses

Up to now the loss was of the form $\mathcal{L}(D) = \ell(\mathcal{E}(D))$.
Now, what about a more general loss $\mathcal{L}(D) := \ell(u(D), w(D))$ for (u, w) the saddle-point?
Typical example is $\min_D \ell(u(D))$, with u solving

$$\min_u |Du|_1 + \frac{\|u - g\|_2^2}{2},$$

D is in a class of operator realizing a consistent discretization of the total variation, and ℓ measures the discrepancy between $u(D)$ and a "true solution" (for instance, exact continuous solution), *cf* C-Pock 2020/21.

General model

So the idea is to consider a generic bilinear saddle-point problem:

$$\min_x \sup_y g(x) + \langle Kx, y\rangle - f^*(y)$$

with g, f^* strongly convex so that $(x(K), y(K))$ is uniquely defined (and continuous). How to differentiate wr K?

General model

So the idea is to consider a generic bilinear saddle-point problem:

$$\min_x \sup_y g(x) + \langle Kx, y \rangle - f^*(y)$$

with g, f^* strongly convex so that $(x(K), y(K))$ is uniquely defined (and continuous). How to differentiate wr K?

Classical method (now): Implement a 1st order algorithm to approximate $u(K)$ with some u^n, $n \geq 1$. Then "unroll" the iterations $(u^0, \dots, u^n)$ and use automatic differentiation and back-propagation to estimate $\nabla_K u^n$.
Issues: strange dependence on u^0, and difficult if the problem is large or requires too many iterations (costs a lot of memory).

General model

So the idea is to consider a generic bilinear saddle-point problem:

$$\min_x \sup_y g(x) + \langle Kx, y \rangle - f^*(y)$$

with g, f^* strongly convex so that $(x(K), y(K))$ is uniquely defined (and continuous). How to differentiate wr K?

Classical method (now): Implement a 1st order algorithm to approximate $u(K)$ with some u^n, $n \geq 1$. Then "unroll" the iterations $(u^0, \ldots, u^n)$ and use automatic differentiation and back-propagation to estimate $\nabla_K u^n$.
Issues: strange dependence on u^0, and difficult if the problem is large or requires too many iterations (costs a lot of memory).

Alternative: *even more* **classical method**: sensivity analysis.

Sensivity analysis

In a first step we assume that f, g are as smooth as needed. Given K and a perturbation L we consider for t small the saddle-point $(x_t, y_t) := (x(K+tL), y(K+tL))$. We start from the stationarity conditions:

$$\begin{cases} (K+tL)^* y_t + \nabla g(x_t) = 0 \\ -(K+tL)x_t + \nabla f^*(y_t) = 0 \end{cases}$$

to find that, after some easy computation,

$$\begin{cases} K^*\eta + D^2 g(x_0)\cdot\xi = -L^* y_0 \\ -K\xi + D^2 f^*(y_0)\cdot\eta = Lx_0 \end{cases}$$

where $\xi = \lim_{t\to 0}(x_t - x_0)/t$ and $\eta = \lim_{t\to 0}(y_t - y_0)/t$.

Sensivity analysis

It follows that

$$\begin{pmatrix} \xi \\ \eta \end{pmatrix} = \begin{pmatrix} D^2 g(x_0) & K^* \\ -K & D^2 f^*(y_0) \end{pmatrix}^{-1} \begin{pmatrix} -L^* y_0 \\ L x_0 \end{pmatrix}$$

and we obtain, using $\nabla \mathcal{L}(K) \cdot L = \langle \nabla \ell(x_0, y_0), (\xi, \eta) \rangle$,

$$\nabla \mathcal{L}(K) \cdot L = \nabla \ell(x_0, y_0)^T \begin{pmatrix} D^2 g(x_0) & K^* \\ -K & D^2 f^*(y_0) \end{pmatrix}^{-1} \begin{pmatrix} -L^* y_0 \\ L x_0 \end{pmatrix}$$

Sensivity analysis

and we obtain, using $\nabla\mathcal{L}(K)\cdot L = \langle \nabla\ell(x_0,y_0),(\xi,\eta)\rangle$,

$$\nabla\mathcal{L}(K)\cdot L = \underbrace{\nabla\ell(x_0,y_0)^T \begin{pmatrix} D^2g(x_0) & K^* \\ -K & D^2f^*(y_0)\end{pmatrix}^{-1}}_{(-X,Y)^T} \begin{pmatrix} -L^*y_0 \\ Lx_0 \end{pmatrix}$$

$\rightarrow$ Introduce *adjoint states* (X,Y) so that

$$\nabla\mathcal{L}(K)\cdot L = \langle X, L^*y_0\rangle + \langle Y, Lx_0\rangle\,,$$

that is ,

$$\nabla\mathcal{L}(K) = y_0\otimes X + Y\otimes x_0$$

Adjoint states

Now, as usual, the adjoint states do not require the knowledge of L (otherwise they would be useless). They satisfy:

$$\begin{cases} D^2 g(x_0)X + K^* Y + \nabla_x \ell(x_0, y_0) = 0 \\ -KX + D^2 f^*(y_0)Y - \nabla_y \ell(x_0, y_0) = 0 \end{cases}$$

and solve the quadratic saddle-point problem:

$$\min_X \sup_Y \langle KX, Y\rangle + \frac{1}{2}\left\langle D^2 g(x_0)X, X\right\rangle - \frac{1}{2}\left\langle D^2 f^*(y_0)Y, Y\right\rangle \\ + \langle \nabla_x \ell(x_0, y_0), X\rangle + \langle \nabla_y \ell(x_0, y_0), Y\rangle$$

Naive Algorithm

Observations: a standard iterative algorithm to solve this problem would rely on iterations such as

$$X^{k+1} = (I + \tau D^2 g(x_0))^{-1}(X^k - \tau K Y^k - \tau \nabla_x \ell(x_0, y_0)),$$

requiring the knowledge of the solution (x_0, y_0) and to compute the (linear) proximity operator

$$(I + \tau D^2 g(x_0))^{-1}.$$

Yet...

Naive Algorithm

Observations: a standard iterative algorithm to solve this problem would rely on iterations such as

$$X^{k+1} = (I + \tau D^2 g(x_0))^{-1}(X^k - \tau K Y^k - \tau \nabla_x \ell(x_0, y_0)),$$

requiring the knowledge of the solution (x_0, y_0) and to compute the (linear) proximity operator

$$(I + \tau D^2 g(x_0))^{-1}.$$

Yet...

- One has that: $\nabla \mathrm{prox}_{\tau g}(x) = (I + \tau D^2 g(\mathrm{prox}_{\tau g}(x)))^{-1}$;
- One can run in parallel an algorithm for computing (x_0, y_0) and the algorithm for X, Y.

Piggyback Algorithm

This is the basic idea of a "Piggyback" differentiation algorithm (*cf* Griewank-Faure 2003, designed to evaluate the derivative of fixed points with respect to some parameters). In this case, one would run in parallel, for appropriate choices of $\tau, \sigma, \theta \in [0,1]$, primal-dual iterations (*cf* [CP11]) of the form:

$$\begin{cases} x^{k+1} = \mathrm{prox}_{\tau g}(x^k - \tau K y^k) \\ X^{k+1} = \nabla \mathrm{prox}_{\tau g}(x^k - \tau K y^k) \cdot (X^k - \tau K Y^k - \tau \nabla_x \ell(x^k, y^k)) \end{cases}$$

Piggyback Algorithm

First choose starting points (x^0, y^0, X^0, Y^0), then for each $k \geq 0$:

1. $\tilde{x} = x^k - \tau K^* y^k$, $\tilde{X} = X^k - \tau(K^* Y^k + \nabla_x \ell(x^k, y^k))$;
2. compute using automatic differentiation $x^{k+1} = \mathrm{prox}_{\tau g}(\tilde{x})$, $X^{k+1} = \nabla \mathrm{prox}_{\tau g}(\tilde{x}) \cdot \tilde{X}$;
3. $\bar{x}^{k+1} := x^{k+1} + \theta(x^{k+1} - x^k)$, $\bar{X}^{k+1} := X^{k+1} + \theta(X^{k+1} - X^k)$,
4. $\tilde{y} = y^k + \sigma K \bar{x}^{k+1}$, $\tilde{Y} = Y^k + \sigma(K \bar{X}^{k+1} + \nabla_y \ell(x^k, y^k))$;
5. compute using a.d. again $y^{k+1} = \mathrm{prox}_{\sigma f^*}(\tilde{y})$, $Y^{k+1} = \nabla \mathrm{prox}_{\sigma f^*}(\tilde{y}) \cdot \tilde{Y}$;
6. return to 1.

Theoretical results

Our first result shows the method makes sense for less regular functions f^*, g:

Theorem *Assume that* g, f^* *are strongly convex and let* (x, y, X, Y) *be a fixed point of the algorithm, for which* $\nabla \mathrm{prox}_{\tau g}(x - \tau K^* y)$ *and* $\nabla \mathrm{prox}_{\tau f^*}(y + \sigma K x)$ *exist. Then* $\mathcal{L}$ *is differentiable at* K *and* $\nabla \mathcal{L}(K) = y \otimes X + Y \otimes x$.

Of course the assumptions imply that g^*, f are $C^{1,1}$. The convergence of the algorithm requires slightly more regularity:

Theorem *Assume that* g, f^* *are strongly convex, and in addition that* g^*, f *are locally* $C^{2,\alpha}$ *for some* $\alpha > 0$. *Then for* τ, σ, θ *properly chosen, the iterates* (x^k, y^k, X^k, Y^K) *converge linearly to a fixed point where the previous Thm holds.*

Why does it work?

- Relies on Moreau's identity which shows, for instance, that

$$\nabla \mathrm{prox}_{\tau g}(x+\tau p) = I - \nabla \mathrm{prox}_{\frac{1}{\tau} g^*}(x+\tau p).$$

(remember also $\nabla \mathrm{prox}_{\tau g}(x) = (I + \tau D^2 g(\mathrm{prox}_{\tau g}(x)))^{-1}$)

Why does it work?

- Relies on Moreau's identity which shows, for instance, that
$$\nabla \mathrm{prox}_{\tau g}(x+\tau p) = I - \nabla \mathrm{prox}_{\frac{1}{\tau} g^*}(x+\tau p).$$
(remember also $\nabla \mathrm{prox}_{\tau g}(x) = (I + \tau D^2 g(\mathrm{prox}_{\tau g}(x)))^{-1}$)
- Relies on Moreau-Yosida regularization, through formulas such as
$$\nabla \mathrm{prox}_{\tau g}(x) = D^2 (g^*)_{\frac{1}{\tau}} \left(\frac{x}{\tau}\right)$$
(in particular if g^* is $C^{2,\alpha}$ then $\mathrm{prox}_{\tau g}$ is $C^{1,\alpha}$).

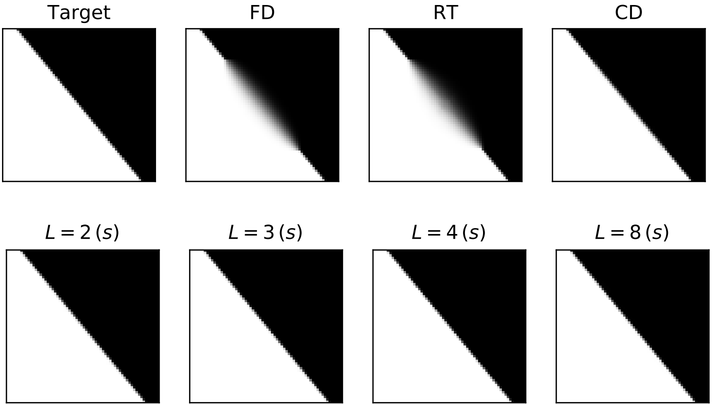
Why does it work?

- Relies on Moreau's identity which shows, for instance, that
$$\nabla \mathrm{prox}_{\tau g}(x+\tau p) = I - \nabla \mathrm{prox}_{\frac{1}{\tau}g^*}(x+\tau p).$$
(remember also $\nabla \mathrm{prox}_{\tau g}(x) = (I + \tau D^2 g(\mathrm{prox}_{\tau g}(x)))^{-1}$)
- Relies on Moreau-Yosida regularization, through formulas such as
$$\nabla \mathrm{prox}_{\tau g}(x) = D^2(g^*)_{\frac{1}{\tau}}\left(\frac{x}{\tau}\right)$$
(in particular if g^* is $C^{2,\alpha}$ then $\mathrm{prox}_{\tau g}$ is $C^{1,\alpha}$).
- The errors between the values of $\nabla \ell$, $\nabla \mathrm{prox}_{\tau g}$ at iterates and at the limit points are controlled thanks to the linear convergence of (x^k, y^k) (*cf* [CP11]). Then, (X^k, Y^k) solve a primal-dual algorithm with errors for which a (less good) linear convergence can also be proved, *cf* Rasch-C. 2020.

Remarks

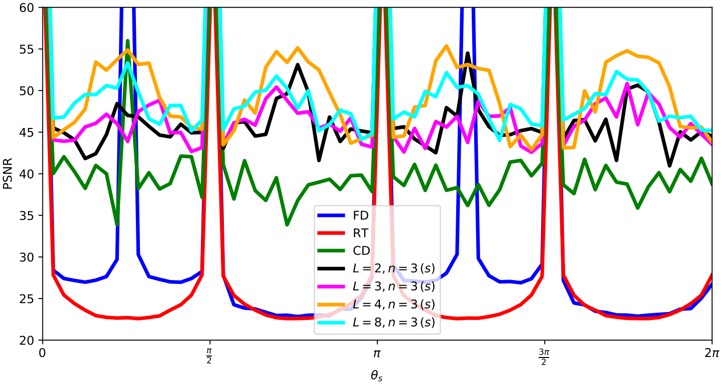
- It is not clear that one can easily drop the regularity assumptions, in addition, for the first theorem, even if prox is differentiable a.e., it is not clear that it will be differentiable precisely at some fixed point.
- Interestingly, the same adjoint states can be used to derivate with parameters in f, g ($f^*(y, \Theta)$, etc). (But this needs more regularity).

Examples



"Inpainting" of a straight line with ad hoc (top) and learned (bottom) discretizations of the total variation.

Examples



Quality of reconstruction (PSNR) as a function of the angle of the discontinuity. The learning significantly improves the isotropy.

Perspectives

- Applications to learning/classification (but too simple classification problems yield overfitting);
- Understand what is computed for totally nonsmooth/non-strongly convex problems (difficult in general, yet it seems to work).
- Compare with backpropagation? (Inpainting experiments require too many iterations for BP.)

Thank you for your attention