

Variational Methods for Image Restoration

Arthur Leclaire



MVA Introduction à l'Imagerie Numérique
October, 13th, 2026

Plan

Imaging Inverse Problems

Optimization for Inverse Problems

Proximal Splitting

Metrics for Inverse Problems

Inverse problem with additive noise:

$$v = \mathcal{A}u_0 + w$$

where

- $u_0 \in \mathbf{R}^d$ is the clean image to recover, $u_0 : \Omega \rightarrow \mathbf{R}^c$
- $\mathcal{A} : \mathbf{R}^d \rightarrow \mathbf{R}^m$
- w is a noise

In many cases, the degradation operator $\mathcal{A}$ can be approximated with a linear operator A , and the noise model w is assumed to be Gaussian.

But, there are also inverse problems with non-linear $\mathcal{A}$ and non-Gaussian noise (e.g. Poisson noise).

Classical Inverse Problems

Application	Forward model	Notes
Denoising [58]	$A = I$	I is the identity matrix
Deconvolution [58, 59]	$\mathcal{A}(\mathbf{x}) = \mathbf{h} * \mathbf{x}$	$\mathbf{h}$ is a known blur kernel and $*$ denotes convolution. When $\mathbf{h}$ is unknown the reconstruction problem is known as blind deconvolution.
Superresolution [60, 61]	$A = SB$	S is a subsampling operator (identity matrix with missing rows) and B is a blurring operator corresponding to convolution with a blur kernel
Inpainting [62]	$A = S$	S is a diagonal matrix where $S_{i,i} = 1$ for the pixels that are sampled and $S_{i,i} = 0$ for the pixels that are not.
Compressive Sensing [63, 64]	$A = SF$ or $A =$ Gaussian or Bernoulli ensemble	S is a subsampling operator (identity matrix with missing rows) and F discrete Fourier transform matrix.
MRI [3]	$A = SFD$	S is a subsampling operator (identity matrix with missing rows), F is the discrete Fourier transform matrix, and D is a diagonal matrix representing a spatial domain multiplication with the coil sensitivity map (assuming a single coil acquisition with Cartesian sampling in a SENSE framework [65]).
Computed tomography [58]	$A = R$	R is the discrete Radon transform [66].
Phase Retrieval [67–70]	$\mathcal{A}(\mathbf{x}) = A\mathbf{x} ^2$	$ \cdot $ denotes the absolute value, the square is taken elementwise, and A is a (potentially complex-valued) measurement matrix that depends on the application. The measurement matrix A is often a variation on a discrete Fourier transform matrix.

(source: [Ongie et al., 2020])

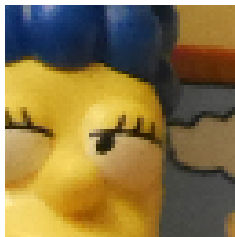
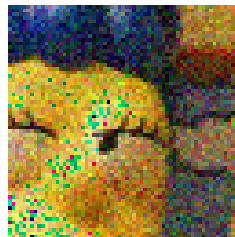
Gaussian denoising

Let's start with the case $A = \text{Id}$, i.e. **image denoising**:

$$v = u_0 + w \quad \text{where} \quad w \sim \mathcal{N}(0, \sigma^2 \text{Id}).$$

We want to estimate u_0 from a single realization of v ...

→ We need to add some prior knowledge on the solution (e.g. regularity assumption).

 u_0  v

Deblurring

- A spatially invariant blur can be modeled by a convolution operator $Au = k * u$
- Several types of blur exist (motion, defocus)
- **Non-blind deblurring** consists in recovering u_0 from

$$v = k * u_0 + w,$$

where $w \sim \mathcal{N}(0, \sigma^2 \text{Id})$.

- We won't tackle blind deblurring here.



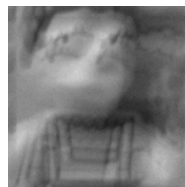
Isotropic blur



Motion blur



Original



Blurred

Example of motion blur

Super-Resolution

Super-resolution consists in finding another version of v at higher resolution.

This is an inverse problem corresponding to the subsampling operator with stride $s \in \mathbb{N}^*$:

$$u_{\downarrow s}(x, y) = u(sx, sy).$$

In practice, we often apply an (*anti-aliasing*) filter before subsampling.

With prefiltering, we obtain the operator

$$Au = (k * u)_{\downarrow s}.$$

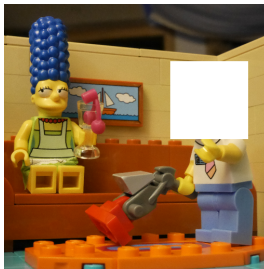
Super-resolution consists in recovering u_0 from

$$v = (k * u_0)_{\downarrow s} + w.$$

The degraded image v is defined on a subgrid of stride s .

Inpainting

Inpainting consists in filling missing regions in images



The degradation operator then writes

$$Au = u1_{\omega}$$

where $\omega \subset \Omega$ is the set of known pixels and $\Omega \setminus \omega$ the mask.

Ill-posed Inverse Problems

We wish to recover u_0 from

$$v = Au_0 + w.$$

The problem is said ill-posed when A is not invertible or with unstable inverse.

Example : For deblurring, $Au = k * u$, we can invert A directly in Fourier domain:

Ill-posed Inverse Problems

We wish to recover u_0 from

$$v = Au_0 + w.$$

The problem is said ill-posed when A is not invertible or with unstable inverse.

Example : For deblurring, $Au = k * u$, we can invert A directly in Fourier domain:

$$u = \mathcal{F}^{-1} \left(\frac{\hat{v}}{\hat{k}} \right) = \mathcal{F}^{-1} \left(\hat{u}_0 + \frac{\hat{w}}{\hat{k}} \right) \longrightarrow \text{but noise explodes !}$$

Ill-posed Inverse Problems

We wish to recover u_0 from

$$v = Au_0 + w.$$

The problem is said ill-posed when A is not invertible or with unstable inverse.

Example : For deblurring, $Au = k * u$, we can invert A directly in Fourier domain:

$$u = \mathcal{F}^{-1} \left(\frac{\hat{v}}{\hat{k}} \right) = \mathcal{F}^{-1} \left(\hat{u}_0 + \frac{\hat{w}}{\hat{k}} \right) \longrightarrow \text{but noise explodes !}$$

When the problem is ill-posed, there may be multiple solutions or erroneous solutions.

It is thus useful to adopt an *a priori* on the solution, e.g. imposing some kind of regularity.

Plan

Imaging Inverse Problems

Optimization for Inverse Problems

Proximal Splitting

Metrics for Inverse Problems

Image Restoration by Optimization

We will therefore try to solve

$$F(u) = \frac{1}{2} \|Au - v\|_2^2 + \lambda R(u)$$

where

- $f(u) = \frac{1}{2} \|Au - v\|_2^2$ is called **data-fidelity**
- $\lambda R(u)$ is called the **regularization** ($\lambda \geq 0$ is a parameter.)

Image Restoration by Optimization

We will therefore try to solve

$$F(u) = \frac{1}{2} \|Au - v\|_2^2 + \lambda R(u)$$

where

- $f(u) = \frac{1}{2} \|Au - v\|_2^2$ is called **data-fidelity**
- $\lambda R(u)$ is called the **regularization** ($\lambda \geq 0$ is a parameter.)

We denote

$$\inf F = \inf \{ F(u) , u \in \mathbf{R}^d \}$$

and

$$\text{Argmin } F = \{ u \in \mathbf{R}^d , F(u) = \inf F \}.$$

The problem $\text{Argmin}_{u \in \mathbf{R}^\Omega} F(u)$ is very high-dimensional, and we need efficient algorithms.

Image Restoration by Optimization

We will therefore try to solve

$$F(u) = \frac{1}{2} \|Au - v\|_2^2 + \lambda R(u)$$

where

- $f(u) = \frac{1}{2} \|Au - v\|_2^2$ is called **data-fidelity**
- $\lambda R(u)$ is called the **regularization** ($\lambda \geq 0$ is a parameter.)

We denote

$$\inf F = \inf \{ F(u) , u \in \mathbf{R}^d \}$$

and

$$\text{Argmin } F = \{ u \in \mathbf{R}^d , F(u) = \inf F \}.$$

The problem $\text{Argmin}_{u \in \mathbf{R}^\Omega} F(u)$ is very high-dimensional, and we need efficient algorithms.

For differentiable F , we can always consider simple gradient descent.

Image Restoration by Optimization

We will therefore try to solve

$$F(u) = \frac{1}{2} \|Au - v\|_2^2 + \lambda R(u)$$

where

- $f(u) = \frac{1}{2} \|Au - v\|_2^2$ is called **data-fidelity**
- $\lambda R(u)$ is called the **regularization** ($\lambda \geq 0$ is a parameter.)

We denote

$$\inf F = \inf \{ F(u) , u \in \mathbf{R}^d \}$$

and

$$\operatorname{Argmin} F = \{ u \in \mathbf{R}^d , F(u) = \inf F \}.$$

The problem $\operatorname{Argmin}_{u \in \mathbf{R}^\Omega} F(u)$ is very high-dimensional, and we need efficient algorithms.

For differentiable F , we can always consider simple gradient descent.

Example: The gradient of $f(u) = \frac{1}{2} \|Au - v\|_2^2$ is

Image Restoration by Optimization

We will therefore try to solve

$$F(u) = \frac{1}{2} \|Au - v\|_2^2 + \lambda R(u)$$

where

- $f(u) = \frac{1}{2} \|Au - v\|_2^2$ is called **data-fidelity**
- $\lambda R(u)$ is called the **regularization** ($\lambda \geq 0$ is a parameter.)

We denote

$$\inf F = \inf \{ F(u) , u \in \mathbf{R}^d \}$$

and

$$\text{Argmin } F = \{ u \in \mathbf{R}^d , F(u) = \inf F \}.$$

The problem $\text{Argmin}_{u \in \mathbf{R}^\Omega} F(u)$ is very high-dimensional, and we need efficient algorithms.

For differentiable F , we can always consider simple gradient descent.

Example: The gradient of $f(u) = \frac{1}{2} \|Au - v\|_2^2$ is $\nabla f(u) = A^T(Au - v)$.

Image Restoration by Optimization

We will therefore try to solve

$$F(u) = \frac{1}{2} \|Au - v\|_2^2 + \lambda R(u)$$

where

- $f(u) = \frac{1}{2} \|Au - v\|_2^2$ is called **data-fidelity**
- $\lambda R(u)$ is called the **regularization** ($\lambda \geq 0$ is a parameter.)

We denote

$$\inf F = \inf \{ F(u) , u \in \mathbf{R}^d \}$$

and

$$\text{Argmin } F = \{ u \in \mathbf{R}^d , F(u) = \inf F \}.$$

The problem $\text{Argmin}_{u \in \mathbf{R}^\Omega} F(u)$ is very high-dimensional, and we need efficient algorithms.

For differentiable F , we can always consider simple gradient descent.

Example: The gradient of $f(u) = \frac{1}{2} \|Au - v\|_2^2$ is $\nabla f(u) = A^T(Au - v)$.

If $Au = k * u$ (periodic convolution), then $A^T u =$

Image Restoration by Optimization

We will therefore try to solve

$$F(u) = \frac{1}{2} \|Au - v\|_2^2 + \lambda R(u)$$

where

- $f(u) = \frac{1}{2} \|Au - v\|_2^2$ is called **data-fidelity**
- $\lambda R(u)$ is called the **regularization** ($\lambda \geq 0$ is a parameter.)

We denote

$$\inf F = \inf \{ F(u) , u \in \mathbf{R}^d \}$$

and

$$\text{Argmin } F = \{ u \in \mathbf{R}^d , F(u) = \inf F \}.$$

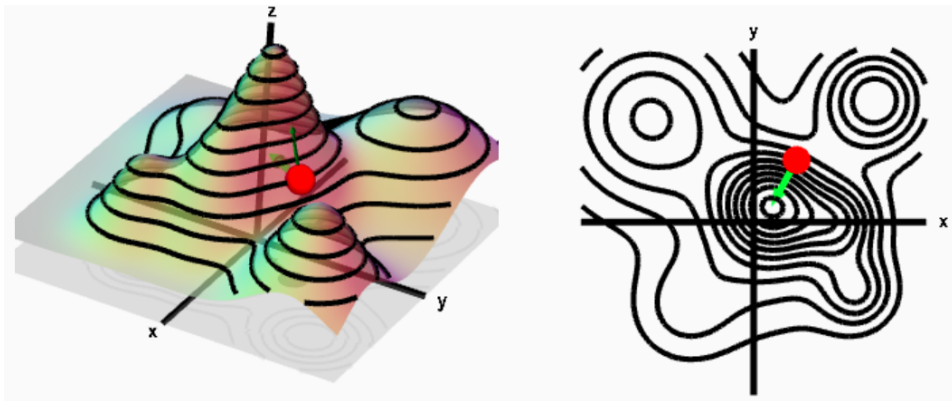
The problem $\text{Argmin}_{u \in \mathbf{R}^\Omega} F(u)$ is very high-dimensional, and we need efficient algorithms.

For differentiable F , we can always consider simple gradient descent.

Example: The gradient of $f(u) = \frac{1}{2} \|Au - v\|_2^2$ is $\nabla f(u) = A^T(Au - v)$.

If $Au = k * u$ (periodic convolution), then $A^T u = \tilde{k} * u$ with $\tilde{k}(\mathbf{x}) = \overline{k(-\mathbf{x})}$.

The Steepest Descent



https://mathinsight.org/directional_derivative_gradient_introduction

Descent Lemma

Let $f : \mathbf{R}^d \rightarrow \mathbf{R}$ be differentiable with L -Lipschitz gradient. Then, for any $x, y \in \mathbf{R}^d$,

$$\begin{aligned} f(y) &= f(x) + \int_0^1 \nabla f(x + t(y - x)) \cdot (y - x) dt \\ &= f(x) + \nabla f(x) \cdot (y - x) + \int_0^1 (\nabla f(x + t(y - x)) - \nabla f(x)) \cdot (y - x) dt \\ &\leq f(x) + \nabla f(x) \cdot (y - x) + \int_0^1 \|\nabla f(x + t(y - x)) - \nabla f(x)\| \|y - x\| dt \\ &\leq f(x) + \nabla f(x) \cdot (y - x) + \int_0^1 Lt \|y - x\|^2 dt \\ &\leq f(x) + \nabla f(x) \cdot (y - x) + \frac{L}{2} \|y - x\|^2. \end{aligned}$$

Consequence: If we choose $\tau \in [0, \frac{2}{L}]$, then

$$f(x - \tau \nabla f(x)) \leq f(x) - \tau \left(1 - \frac{\tau L}{2}\right) \|\nabla f(x)\|^2 \leq f(x).$$

Gradient Descent

We consider here the gradient descent method:

$$x_{n+1} = x_n - \tau_n \nabla f(x_n) ,$$

where $\tau_n > 0$ is a sequence of step sizes.

- For $\tau_n = \tau$ constant, we speak of fixed step size.
- We speak of optimal step size if, at each iteration n , we choose

$$\tau_n \in \underset{t \in \mathbf{R}}{\operatorname{Argmin}} f(x_n - t \nabla f(x_n)).$$

The descent lemma gives that for f differentiable with L -Lipschitz gradient and $\tau \leq \frac{2}{L}$,

$$f(x_{n+1}) \leq f(x_n)$$

Thus, if f is lower bounded, $f(x_n)$ converges.

Convexity and Minimum

The function $f : \mathbf{R}^d \rightarrow \mathbf{R}$ is convex if for all $x, y \in \mathbf{R}^d$,

$$\forall t \in (0, 1), \quad f((1-t)x + ty) \leq (1-t)f(x) + t f(y).$$

It is said strictly convex if the inequality is strict.

If f is convex and differentiable, one can show that for any $x, y \in \mathbf{R}^d$,

$$f(y) \geq f(x) + \nabla f(x) \cdot (y - x).$$

Consequence : If f is convex and differentiable, then

$$x \in \text{Argmin } f \iff \nabla f(x) = 0.$$

The argmin is unique as soon as f is strictly convex.

Strong Convexity

We say that f is α -convex (with $\alpha \in \mathbf{R}$) if $f - \frac{\alpha}{2} \|\cdot\|^2$ is convex.

When $\alpha > 0$, we say that f is **strongly convex**.

Remark : The convexity and the gradient Lipschitz constant can be read on the Hessian:

If $A, B \in \mathbf{R}^{d \times d}$ are symmetric, we write $A \succeq B$ if $A - B$ is semi-definite positive, i.e.

$$\forall x \in \mathbf{R}^d, \quad Ax \cdot x \geq Bx \cdot x.$$

For $f : \mathbf{R}^d \rightarrow \mathbf{R}$ of class $\mathcal{C}^2$,

∇f is L -Lipschitz iff $\forall x \in \mathbf{R}^d, -L\text{Id} \preceq \nabla^2 f(x) \preceq L\text{Id}$.
i.e. $\forall x$ the eigenvalues of $\nabla^2 f(x)$ have moduli $\leq L$.

f is α -convex iff $\forall x \in \mathbf{R}^d, \nabla^2 f(x) \succeq \alpha\text{Id}$
i.e. $\forall x$ the eigenvalues of $\nabla^2 f(x)$ are all $\geq \alpha$.

Convergence Guarantees, Convex Case

See e.g. [Dossal et al., 2024]

Theorem

Let $f : \mathbf{R}^d \rightarrow \mathbf{R}$ be convex differentiable with ∇f L -Lipschitz. Assume that $\text{Argmin } f$ is non-empty. Let $\tau \in (0, \frac{2}{L})$, $x_0 \in \mathbf{R}^d$ and (x_n) the sequence defined by

$$x_{n+1} = x_n - \tau \nabla f(x_n) .$$

Then (x_n) converges towards an element of $\text{Argmin } f$.

Theorem

Let $f : \mathbf{R}^d \rightarrow \mathbf{R}$ be differentiable and α -strongly convex with L -Lipschitz gradient. (Thus $\frac{1}{L} \leq \frac{1}{\alpha}$.) Then there exists a unique $x_* \in \text{Argmin } f$, and for $0 < \tau \leq \frac{1}{L}$, we have

$$\|x_n - x_*\|^2 \leq (1 - \tau\alpha)^n \|x_0 - x_*\|^2 .$$

Optimization for Inverse Problems

Back to our problem: we wish to minimize

$$F(u) = f(u) + g(u)$$

with $f(u) = \frac{1}{2}\|Au - v\|^2$ and $g(u) = \lambda R(u)$, $\lambda > 0$.

If f and g are differentiable, then we can use plain gradient descent.

Good news: By automatic differentiation you need only coding $F(u)$...

Optimization for Inverse Problems

Back to our problem: we wish to minimize

$$F(u) = f(u) + g(u)$$

with $f(u) = \frac{1}{2} \|Au - v\|^2$ and $g(u) = \lambda R(u)$, $\lambda > 0$.

If f and g are differentiable, then we can use plain gradient descent.

Good news: By automatic differentiation you need only coding $F(u)$...

But ! in order to avoid instability problems, you'd better know what F does...
(for example, useful to have an idea of the Lipschitz constant of ∇F)

Optimization for Inverse Problems

Back to our problem: we wish to minimize

$$F(u) = f(u) + g(u)$$

with $f(u) = \frac{1}{2}\|Au - v\|^2$ and $g(u) = \lambda R(u)$, $\lambda > 0$.

If f and g are differentiable, then we can use plain gradient descent.

Good news: By automatic differentiation you need only coding $F(u)$...

But ! in order to avoid instability problems, you'd better know what F does...
(for example, useful to have an idea of the Lipschitz constant of ∇F)

EXERCISE: Let $k : \Omega \rightarrow \mathbf{R}_+$ such that $\sum_{\mathbf{x} \in \Omega} k(\mathbf{x}) = 1$

What is the Lipschitz constant of the gradient of $f(u) = \frac{1}{2}\|k * u - v\|^2$?

Explicit Regularizations

We define the discrete derivatives of u by

$$\nabla u(x, y) = \begin{pmatrix} \partial_1 u(x, y) \\ \partial_2 u(x, y) \end{pmatrix} \quad \text{avec} \quad \begin{cases} \partial_1 u(x, y) = d_1 * u(x, y) = u(x+1, y) - u(x, y) \\ \partial_2 u(x, y) = d_2 * u(x, y) = u(x, y+1) - u(x, y) \end{cases}.$$

We define **Tychonov regularization** by

$$\|\nabla u\|_2^2 = \sum_{\mathbf{x} \in \Omega} \|\nabla u(\mathbf{x})\|^2 = \sum_{\mathbf{x} \in \Omega} |\partial_1 u(\mathbf{x})|^2 + |\partial_2 u(\mathbf{x})|^2.$$

We define the **total variation** by

$$\text{TV}(u) = \|\nabla u\|_1 = \sum_{\mathbf{x} \in \Omega} \|\nabla u(\mathbf{x})\| = \sum_{\mathbf{x} \in \Omega} \sqrt{|\partial_1 u(\mathbf{x})|^2 + |\partial_2 u(\mathbf{x})|^2}.$$

We will see that Tychonov reg. promotes smooth images,
while TV promotes piecewise smooth images (*cartoon* images).

Tychonov denoising

Let us minimize

$$F(u) = \frac{1}{2} \|u - v\|^2 + \lambda R(u)$$

with **Tychonov regularization** $R(u) = \|\nabla u\|_2^2$.

Tychonov denoising

Let us minimize

$$F(u) = \frac{1}{2} \|u - v\|^2 + \lambda R(u)$$

with Tychonov regularization $R(u) = \|\nabla u\|_2^2$.

We have $\nabla R(u) = 2\nabla^T \nabla u$.

Tychonov denoising

Let us minimize

$$F(u) = \frac{1}{2} \|u - v\|^2 + \lambda R(u)$$

with Tychonov regularization $R(u) = \|\nabla u\|_2^2$.

We have $\nabla R(u) = 2\nabla^T \nabla u$. As F is convex,

$$u \in \operatorname{Argmin} F \iff \nabla F(u) = 0 \iff u - v + 2\lambda \nabla^T \nabla u = 0 \iff u = (I + 2\lambda \nabla^T \nabla)^{-1} v$$

Tychonov denoising

Let us minimize

$$F(u) = \frac{1}{2} \|u - v\|^2 + \lambda R(u)$$

with **Tychonov regularization** $R(u) = \|\nabla u\|_2^2$.

We have $\nabla R(u) = 2\nabla^T \nabla u$. As F is convex,

$$u \in \operatorname{Argmin} F \iff \nabla F(u) = 0 \iff u - v + 2\lambda \nabla^T \nabla u = 0 \iff u = (I + 2\lambda \nabla^T \nabla)^{-1} v$$

For $p : \Omega \rightarrow \mathbf{R}^2$, $\nabla^T p$ is given by

$$\nabla^T p(x, y) = p_1(x-1, y) - p_1(x, y) + p_2(x, y-1) - p_2(x, y).$$

Actually, $\operatorname{div}(p) := -\nabla^T p$ is a discrete divergence and $\Delta u := -\nabla^T \nabla u$ is a discrete Laplacian.

Explicit Solution: Wiener filtering

Theorem

Let $v \in \mathbb{C}^\Omega$ and $\lambda > 0$. The function $F : \mathbb{C}^\Omega \rightarrow \mathbf{R}_+$ defined by

$$\forall u \in \mathbb{C}^\Omega, \quad F(u) = \frac{1}{2} \|u - v\|_2^2 + \lambda \|\nabla u\|_2^2$$

has a minimum attained at a unique $u_* \in \mathbb{C}^\Omega$, which is given in Fourier domain:

$$\forall (\xi, \zeta) \in \Omega, \quad \hat{u}_*(\xi, \zeta) = \frac{\hat{v}(\xi, \zeta)}{1 + 2\lambda \hat{L}(\xi, \zeta)}$$

where $\hat{L}(\xi, \zeta) = |\hat{d}_1(\xi, \zeta)|^2 + |\hat{d}_2(\xi, \zeta)|^2 = 4 \left(\sin^2 \left(\frac{\pi \xi}{M} \right) + \sin^2 \left(\frac{\pi \zeta}{N} \right) \right)$.

Remarks:

- d_1, d_2 are the kernel derivatives, e.g. $d_1 = \delta_{(-1,0)} - \delta_{(0,0)}$. So $\hat{L}$ is the kernel of $-\Delta$ filter.
- The theorem adapts for deblurring with Tychonov regularization:

$$\forall (\xi, \zeta) \in \Omega, \quad \hat{u}_*(\xi, \zeta) = \frac{\overline{\hat{k}(\xi, \zeta)} \hat{v}(\xi, \zeta)}{|\hat{k}(\xi, \zeta)|^2 + 2\lambda \hat{L}(\xi, \zeta)}$$

Link with an evolution model

The gradient descent on

$$F(u) = \frac{1}{2} \|u - v\|_2^2 + \lambda \|\nabla u\|_2^2$$

writes as

$$u_{n+1} - u_n = -\tau(u_n - v) + 2\lambda\tau\Delta u_n .$$

The sequence (u_n) converges to u_* as soon as $\tau < \frac{2}{L}$ with $L = \|I + 2\lambda\nabla^T\nabla\| = 1 + 16\lambda$.

Link with an evolution model

The gradient descent on

$$F(u) = \frac{1}{2} \|u - v\|_2^2 + \lambda \|\nabla u\|_2^2$$

writes as

$$u_{n+1} - u_n = -\tau(u_n - v) + 2\lambda\tau\Delta u_n.$$

The sequence (u_n) converges to u_* as soon as $\tau < \frac{2}{L}$ with $L = \|I + 2\lambda\nabla^T\nabla\| = 1 + 16\lambda$.

If we drop the data-fidelity... then gradient descent on $u \mapsto \|\nabla u\|_2^2$ gives

$$u_{n+1} - u_n = 2\tau\Delta u_n$$

This is a discretization of the heat equation $\partial_t u = c\Delta u$ with initial condition u_0 .

Smoothed Total Variation

What if we want to minimize

$$F(u) = \frac{1}{2} \|u - v\|_2^2 + \lambda \text{TV}(u)?$$

Problem: The total variation is **not differentiable**.

Smoothed Total Variation

What if we want to minimize

$$F(u) = \frac{1}{2} \|u - v\|_2^2 + \lambda \text{TV}(u)?$$

Problem: The total variation is **not differentiable**.

To find the true solution to this problem, one can use primal-dual optimization (not covered here).

Smoothed Total Variation

What if we want to minimize

$$F(u) = \frac{1}{2} \|u - v\|_2^2 + \lambda \text{TV}(u)?$$

Problem: The total variation is **not differentiable**.

To find the true solution to this problem, one can use primal-dual optimization (not covered here).

A simple (but cheap...) alternative: consider a smoothed variant: For $\varepsilon > 0$, let

$$\text{TV}_\varepsilon(u) = \sum_{(x,y) \in \Omega} \sqrt{\varepsilon^2 + \partial_1 u(x,y)^2 + \partial_2 u(x,y)^2}.$$

One can see that

$$\nabla \text{TV}_\varepsilon(u) = \nabla^T \left(\frac{\nabla u}{\sqrt{\varepsilon^2 + \|\nabla u\|_2^2}} \right).$$

And one can show that $\nabla \text{TV}_\varepsilon$ is $\frac{8}{\varepsilon}$ -Lipschitz.

We can thus minimize F by gradient descent with $\tau < \frac{2}{1 + \frac{8\lambda}{\varepsilon}}$.

Denoising Examples



Noisy
PSNR = 19.93



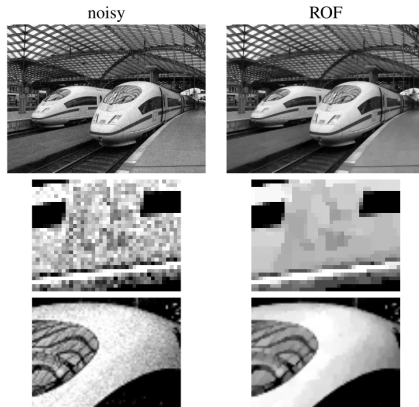
Tychonov denoising
PSNR = 25.89



TV_ϵ denoising
PSNR = 27.21

Another TV Denoising Example

Here is another example of denoising, with the true TV, from [Louchet and Moisan, 2014]:



Extension to Other Inverse Problems

- The previous methodology extends to more complicated inverse problems.
(For example, deblurring: see the practical session.)
- For many inverse problems, it is straightforward to compute the gradient of the data-fidelity.
- So gradient descent is tractable as soon as you use a convex differentiable regularization.
NB: With Tychonov or TV_ϵ regularization, we have all hypotheses needed for convergence.

Extension to Other Inverse Problems

- The previous methodology extends to more complicated inverse problems.
(For example, deblurring: see the practical session.)
- For many inverse problems, it is straightforward to compute the gradient of the data-fidelity.
- So gradient descent is tractable as soon as you use a convex differentiable regularization.
NB: With Tychonov or TV_ϵ regularization, we have all hypotheses needed for convergence.
- Remark: If you don't know $\text{Lip}(\nabla F)$, you may resort to *backtracking* strategies.
(It remains provably convergent as soon as you know that ∇F is Lipschitz!)

Extension to Other Inverse Problems

- The previous methodology extends to more complicated inverse problems.
(For example, deblurring: see the practical session.)
- For many inverse problems, it is straightforward to compute the gradient of the data-fidelity.
- So gradient descent is tractable as soon as you use a convex differentiable regularization.
NB: With Tychonov or TV_ϵ regularization, we have all hypotheses needed for convergence.
- Remark: If you don't know $\text{Lip}(\nabla F)$, you may resort to *backtracking* strategies.
(It remains provably convergent as soon as you know that ∇F is Lipschitz!)
- If you want to consider non-differentiable regularizations, you may use other optimization tools...

Plan

Imaging Inverse Problems

Optimization for Inverse Problems

Proximal Splitting

Metrics for Inverse Problems

Proximal Operator

(see e.g. [Bauschke and Combettes, 2011])

Definition

Let $g : \mathbf{R}^d \rightarrow \mathbf{R} \cup \{-\infty, +\infty\}$. The proximal operator of g is defined by

$$\forall u \in \mathbf{R}^d, \quad \text{Prox}_g(u) = \underset{z \in \mathbf{R}^d}{\text{Argmin}} \frac{1}{2} \|u - z\|_2^2 + g(z) \subset \mathbf{R}^d.$$

Proximal Operator

(see e.g. [Bauschke and Combettes, 2011])

Definition

Let $g : \mathbf{R}^d \rightarrow \mathbf{R} \cup \{-\infty, +\infty\}$. The proximal operator of g is defined by

$$\forall u \in \mathbf{R}^d, \quad \text{Prox}_g(u) = \underset{z \in \mathbf{R}^d}{\text{Argmin}} \frac{1}{2} \|u - z\|_2^2 + g(z) \subset \mathbf{R}^d.$$

Proposition

Let $g : \mathbf{R}^d \rightarrow \mathbf{R} \cup \{+\infty\}$ be a lower semi-continuous convex function such that $g \not\equiv +\infty$. Then

- $\text{Prox}_g(u)$ is single-valued, and thus defines a point $\text{Prox}_g(u) \in \mathbf{R}^d$.
- If besides g is differentiable, then $p = \text{Prox}_g(u)$ satisfies

$$p = \text{Prox}_g(u) \Leftrightarrow u = (\text{Id} + \nabla g)(p).$$

Proximal Operator

(see e.g. [Bauschke and Combettes, 2011])

Definition

Let $g : \mathbf{R}^d \rightarrow \mathbf{R} \cup \{-\infty, +\infty\}$. The proximal operator of g is defined by

$$\forall u \in \mathbf{R}^d, \quad \text{Prox}_g(u) = \underset{z \in \mathbf{R}^d}{\text{Argmin}} \frac{1}{2} \|u - z\|_2^2 + g(z) \in \mathbf{R}^d.$$

Proposition

Let $g : \mathbf{R}^d \rightarrow \mathbf{R} \cup \{+\infty\}$ be a lower semi-continuous convex function such that $g \not\equiv +\infty$. Then

- $\text{Prox}_g(u)$ is single-valued, and thus defines a point $\text{Prox}_g(u) \in \mathbf{R}^d$.
- If besides g is differentiable, then $p = \text{Prox}_g(u)$ satisfies

$$p = \text{Prox}_g(u) \Leftrightarrow u = (\text{Id} + \nabla g)(p).$$

Example

- If $g = \iota_C$ the indicator function of $C \subset \mathbf{R}^d$ closed and convex (i.e. ι_C is 0 on C and $+\infty$ outside) then Prox_g is the orthogonal projection on C .
- If $f(u) = \frac{1}{2} \|Au - v\|_2^2$, then $\text{Prox}_{\tau f}(u) = \dots$

Proximal Operator

(see e.g. [Bauschke and Combettes, 2011])

Definition

Let $g : \mathbf{R}^d \rightarrow \mathbf{R} \cup \{-\infty, +\infty\}$. The proximal operator of g is defined by

$$\forall u \in \mathbf{R}^d, \quad \text{Prox}_g(u) = \underset{z \in \mathbf{R}^d}{\text{Argmin}} \frac{1}{2} \|u - z\|_2^2 + g(z) \in \mathbf{R}^d.$$

Proposition

Let $g : \mathbf{R}^d \rightarrow \mathbf{R} \cup \{+\infty\}$ be a lower semi-continuous convex function such that $g \not\equiv +\infty$. Then

- $\text{Prox}_g(u)$ is single-valued, and thus defines a point $\text{Prox}_g(u) \in \mathbf{R}^d$.
- If besides g is differentiable, then $p = \text{Prox}_g(u)$ satisfies

$$p = \text{Prox}_g(u) \Leftrightarrow u = (\text{Id} + \nabla g)(p).$$

Example

- If $g = \iota_C$ the indicator function of $C \subset \mathbf{R}^d$ closed and convex (i.e. ι_C is 0 on C and $+\infty$ outside) then Prox_g is the orthogonal projection on C .
- If $f(u) = \frac{1}{2} \|Au - v\|_2^2$, then $\text{Prox}_{\tau f}(u) = \dots$ exercise for next week! ;-)

Proximal Gradient Descent

In order to minimize $F = f + g$, one can use the proximal gradient descent (PGD) algorithm:

$$u_{n+1} = \text{Prox}_{\tau g}(u_n - \tau \nabla f(u_n)) ,$$

where $\tau > 0$ is a fixed step size. Benefit is: g may be non-differentiable.

Proximal Gradient Descent

In order to minimize $F = f + g$, one can use the proximal gradient descent (PGD) algorithm:

$$u_{n+1} = \text{Prox}_{\tau g}(u_n - \tau \nabla f(u_n)) ,$$

where $\tau > 0$ is a fixed step size. Benefit is: g may be non-differentiable.

Theorem (See e.g. [Dossal et al., 2024])

Let $f : \mathbf{R}^d \rightarrow \mathbf{R}$ be a convex differentiable function with L_f -Lipschitz gradient.

Let $g : \mathbf{R}^d \rightarrow \mathbf{R} \cup \{+\infty\}$ be convex l.s.c. such that $g \not\equiv +\infty$.

Assume that $F = f + g$ attains its infimum at some point in $\mathbf{R}^d$. Assume $\tau < \frac{2}{L_f}$.

Then the sequence (u_n) given by the PGD algorithm converges to some $u_* \in \text{Argmin } F$.

Proximal Gradient Descent

In order to minimize $F = f + g$, one can use the proximal gradient descent (PGD) algorithm:

$$u_{n+1} = \text{Prox}_{\tau g}(u_n - \tau \nabla f(u_n)) ,$$

where $\tau > 0$ is a fixed step size. Benefit is: g may be non-differentiable.

Theorem (See e.g. [Dossal et al., 2024])

Let $f : \mathbf{R}^d \rightarrow \mathbf{R}$ be a convex differentiable function with L_f -Lipschitz gradient.

Let $g : \mathbf{R}^d \rightarrow \mathbf{R} \cup \{+\infty\}$ be convex l.s.c. such that $g \not\equiv +\infty$.

Assume that $F = f + g$ attains its infimum at some point in $\mathbf{R}^d$. Assume $\tau < \frac{2}{L_f}$.

Then the sequence (u_n) given by the PGD algorithm converges to some $u_* \in \text{Argmin } F$.

Remarks:

- This theorem applies even for non-differentiable (or even non continuous) g .
- It generalizes the projected gradient descent algorithm: for $g = \iota_C$, PGD reads as

$$u_{n+1} = \text{Proj}_C(u_n - \tau \nabla f(u_n)) .$$

Two fundamental Examples

For any $x \in \mathbf{R}^d$, let us define

$$\|x\|_0 = |\{ i \in \{1, \dots, d\} \mid x_i \neq 0 \}|.$$

Notice that $\|\cdot\|_0$ is low on vectors having only few non-zero components.

Two fundamental Examples

For any $x \in \mathbf{R}^d$, let us define

$$\|x\|_0 = |\{ i \in \{1, \dots, d\} \mid x_i \neq 0 \}|.$$

Notice that $\|\cdot\|_0$ is low on vectors having only few non-zero components.
We will also consider a “convex approximation” of $\|\cdot\|_0$ with the ℓ^1 -norm:

$$\|x\|_1 = \sum_{i=1}^d |x_i|.$$

Two fundamental Examples

For any $x \in \mathbf{R}^d$, let us define

$$\|x\|_0 = |\{ i \in \{1, \dots, d\} \mid x_i \neq 0 \}|.$$

Notice that $\|\cdot\|_0$ is low on vectors having only few non-zero components.
We will also consider a “convex approximation” of $\|\cdot\|_0$ with the ℓ^1 -norm:

$$\|x\|_1 = \sum_{i=1}^d |x_i|.$$

Let us define

$$\forall t \in \mathbf{R}, \quad \text{HT}_\lambda(t) = t \mathbf{1}_{|t| > \sqrt{2\lambda}} \quad \text{and} \quad \text{ST}_\lambda(t) = \text{sgn}(t)(|t| - \lambda)_+.$$

We then extend HT_λ and ST_λ to vectors by applying it component-wise.

Then one can show (exercise!) that for any $x \in \mathbf{R}^d$,

- $\text{HT}_\lambda(x) \in \text{Prox}_{\lambda\|\cdot\|_0}(x)$ (hard thresholding)
- $\text{ST}_\lambda(x) = \text{Prox}_{\lambda\|\cdot\|_1}(x)$ (soft thresholding)

Thresholding in an orthogonal basis

Let $B \in \mathbf{R}^{d \times d}$ be an orthogonal matrix.

Let $q \in \{0, 1\}$ consider the regularization defined by

$$\forall u \in \mathbf{R}^d, \quad R_q(u) = \lambda \|B^T u\|_q.$$

Let us define

$$\text{HT}_\lambda^B(u) = B \text{HT}_\lambda(B^T u) \quad \text{and} \quad \text{ST}_\lambda^B(u) = B \text{ST}_\lambda(B^T u).$$

Theorem

For any $u \in \mathbf{R}^d$, we have

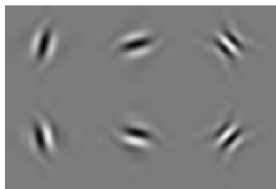
- $\text{HT}_\lambda^B(u) \in \text{Prox}_{R_0}(u)$ (hard thresholding in the basis B)
- $\text{ST}_\lambda^B(u) = \text{Prox}_{R_1}(u)$ (soft thresholding in the basis B)

NB: Both these proximal operators cancel small coefficients of u in the basis B .

A Brief Detour Into Wavelet Transforms

[Mallat, 2009]

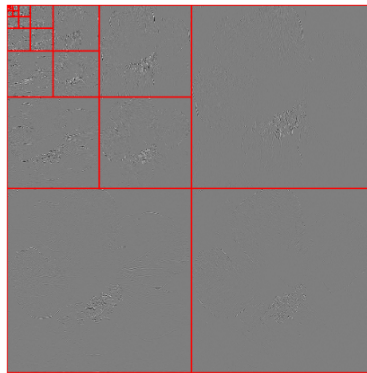
- The wavelet decomposition consists in computing coefficients in a “wavelet basis”.
- A wavelet is an oscillating function with concentrated support.
- Wavelet decomposition is analogous to Fourier transform, but wavelets have “limited” support.
- The construction of “orthogonal wavelet bases” has been a breakthrough in signal processing, especially with continuous compactly-supported functions [Daubechies, 1988], or in the discrete case with a fast practical algorithm [Mallat, 1989].



Example of 6 elements of a wavelet basis

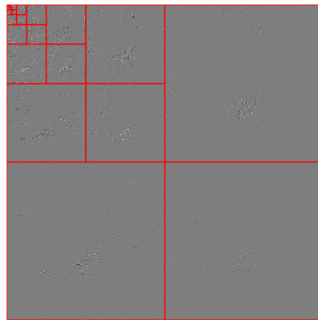
Wavelet Decomposition

- 2d wavelets are useful to represent efficiently edges (contours) in images.
- Wavelet coefficients can be seen as well-sampled multi-scale derivatives at every location.
- Regularizing with ℓ^0 or ℓ^1 norm in a wavelet basis imposes sparsity of wavelet coefficients.



Wavelet Thresholding

- Most of the image geometry is concentrated in the high wavelet coefficients.
- Thresholding wavelet coefficients keeps most of the geometry but smooths textures.
- This is helpful for denoising or compression (see JPEG2000).



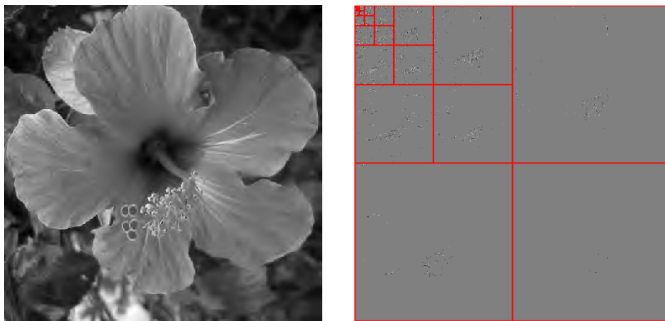
Before thresholding

Example taken from G. Peyré's numerical tours:

https://www.numerical-tours.com/python/introduction_4_fourier_wavelets/

Wavelet Thresholding

- Most of the image geometry is concentrated in the high wavelet coefficients.
- Thresholding wavelet coefficients keeps most of the geometry but smooths textures.
- This is helpful for denoising or compression (see JPEG2000).



After thresholding

Example taken from G. Peyré's numerical tours:

https://www.numerical-tours.com/python/introduction_4_fourier_wavelets/

Iterative Soft Thresholding

[Daubechies et al., 2004]

- One can solve image inverse problems with wavelet ℓ^1 regularization:

$$F(u) = \frac{1}{2} \|Au - v\|_2^2 + \lambda \|W^T u\|_1,$$

where W is the orthogonal matrix related to an orthogonal wavelet basis of $\mathbf{R}^\Omega$.

- For that, we can thus use the following PGD algorithm (known as *Iterative Soft-Thresholding*):

$$u_{n+1} = \text{ST}_\lambda^W(u_n - \tau \nabla f(u_n)) .$$

- It converges as soon as $\tau < \frac{2}{\|A^T A\|}$, with $(F(u_n))$ non-increasing.

Denoising with Various Priors

In a following session, we will discuss the denoising problem in more details.

Several solutions have been proposed for image denoising:

- Total variation denoising [Rudin et al., 1992]
- Wavelet denoising [Mallat, 2009]
- Patch-based denoising [Buades et al., 2005, Zoran and Weiss, 2011, Yu et al., 2011]
- Neural denoising [Zhang et al., 2021]

Denoising with Various Priors

In a following session, we will discuss the denoising problem in more details.

Several solutions have been proposed for image denoising:

- Total variation denoising [Rudin et al., 1992]
- Wavelet denoising [Mallat, 2009]
- Patch-based denoising [Buades et al., 2005, Zoran and Weiss, 2011, Yu et al., 2011]
- Neural denoising [Zhang et al., 2021]

Each of these solutions somehow imposes some regularity prior on the image.

This has led to the idea of “Regularization by Denoising” [Romano et al., 2017].

Plug-and-Play algorithms

[Venkatakrishnan et al., 2013]

- Plug-and-Play (PnP) algorithms are inspired by proximal splitting (which is used to minimize $F_\lambda = f + g$ where f is the data-fidelity and g a regularization term)
- The core idea of PnP is start from some splitting algorithm, for example PGD:

$$u_{k+1} = \text{Prox}_g \circ (\text{Id} - \tau \nabla f)(u_k) \quad (\text{PGD})$$

and then to replace $\text{Prox}_{\lambda g}$ by an off-the-shelf denoiser:

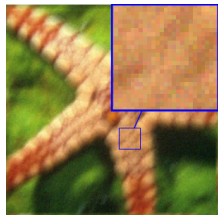
$$u_{k+1} = D_s \circ (\text{Id} - \tau \nabla f)(u_k) \quad (\text{PnP-PGD})$$

- D_s is a pre-defined denoiser acting at noise level $s > 0$.
- In practice, it works pretty well! and proves relatively stable on many inverse problems.

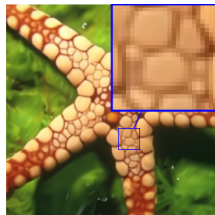
Example of Results with a PnP algorithm

[Hurault et al., 2021]

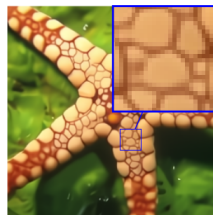
- PnP methods quickly attain impressive performance for image restoration.
- Besides, several PnP methods (not all!) can be shown provably convergent.



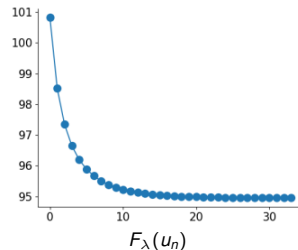
Blurred
(20.97dB)



DPIP
(29.76dB)



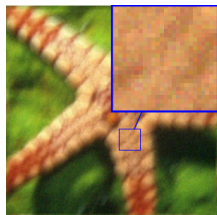
GS-PnP
(29.9dB)



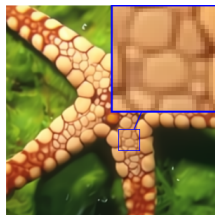
Example of Results with a PnP algorithm

[Hurault et al., 2021]

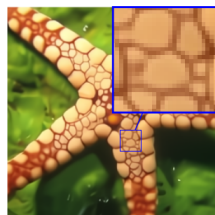
- PnP methods quickly attain impressive performance for image restoration.
- Besides, several PnP methods (not all!) can be shown provably convergent.



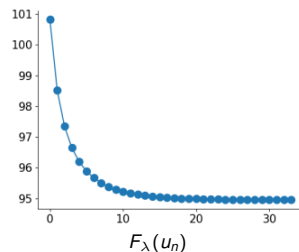
Blurred
(20.97dB)



DPIR
(29.76dB)



GS-PnP
(29.9dB)



- More generally, one may address image generation with iterations based on some denoiser.
- But for that, the denoiser's training should be adapted carefully!

Plan

Imaging Inverse Problems

Optimization for Inverse Problems

Proximal Splitting

Metrics for Inverse Problems

Euclidean metrics

- Given two images u and v of size $M \times N$ with graylevels between 0 and 255.
- Denote $\Omega = \{0, \dots, M-1\} \times \{0, \dots, N-1\}$ the pixel domain
- Mean Square Error $\downarrow$:

$$\text{MSE}(u, v) = \frac{1}{MN} \sum_{\mathbf{x} \in \Omega} (u(\mathbf{x}) - v(\mathbf{x}))^2$$

- Root Mean Square Error $\downarrow$:

$$\text{RMSE}(u, v) = \left(\frac{1}{MN} \sum_{\mathbf{x} \in \Omega} (u(\mathbf{x}) - v(\mathbf{x}))^2 \right)^{\frac{1}{2}}$$

- Peak Signal to Noise Ratio $\uparrow$:

$$\text{PSNR}(u, v) = 20 \log_{10} \left(\frac{\text{MAX}}{\text{RMSE}(u, v)} \right) \quad (\text{where MAX} = 255)$$

- Useful for inverse problems such as denoising.
- Not ideal when one hopes to generate new content.

Structural similarity index measure (SSIM ↑) [Wang et al., 2004]

Between patches:

- Given two patches x, y (typically of size 8×8 or 11×11 with a Gaussian windowing)

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \in [-1, 1]$$

with:

- μ_x the pixel sample mean of x
 - μ_y the pixel sample mean of y
 - σ_x^2 the variance of x
 - σ_y^2 the variance of y
 - σ_{xy} the covariance of x and y
 - $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$ two variables to stabilize the division with weak denominator, with the range $L = 255$ or 1 and $k_1 = 0.01$ and $k_2 = 0.03$ by default.
- SSIM(x, y) is the product of three terms:

Luminance	Contrast	Structure
$l(x, y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1}$	$c(x, y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2}$	$s(x, y) = \frac{\sigma_{xy} + c_2/2}{\sigma_x\sigma_y + c_2/2}$

Structural similarity index measure (SSIM $\uparrow$) [Wang et al., 2004]

Between images:

- Given two images u and v of size $M \times N$ with graylevels between 0 and 255, define the Mean-SSIM by averaging over all patches:

$$(M)SSIM(u, v) = \text{mean}(\{SSIM(P_{\mathbf{x}}(u), P_{\mathbf{x}}(v)), \mathbf{x} + \omega \subset \Omega\})$$

where $P_{\mathbf{x}}(u)$ is the restriction of u on the patch $\mathbf{x} + \omega$.

- There are also multiscale variants.
- SSIM is not a distance, its range is $[-1, 1]$.
- SSIM is closer to a perceptual distance, especially regarding local textures.

LPIPS ↓ [Zhang et al., 2018]

LPIPS: Learned Perceptual Image Patch Similarity

- Previous works on texture synthesis [Gatys et al., 2015] and style transfer [Gatys et al., 2016]
- [Johnson et al., 2016] have shown the importance of the VGG [Simonyan and Zisserman, 2015] features for perceptual similarity between images.
- This means that intermediate features of classification CNN are useful in their own: “**a good feature is a good feature**. Features that are good at semantic tasks are also good at self-supervised and unsupervised tasks, and also provide good models of both human perceptual behavior and macaque neural activity.”

LPIPS model: Define a perceptual distance between 64×64 patches by computing a Euclidean norm between features:

$$\text{LPIPS}(u, v)^2 = \sum_{\text{layers } \ell} \frac{1}{H_\ell W_\ell} \sum_{i,j} \| \mathbf{w}_\ell \odot (F^\ell(u)_{i,j} - F^\ell(v)_{i,j}) \|^2_2$$

where for each layer ℓ , the neural response $F^\ell(u)$ is weighted by channel weights $\mathbf{w}_\ell \in \mathbf{R}^{G_\ell}$ that are learned to reproduce human evaluation of distortion between patches.

Take-home Messages

- We have introduced optimization methods for solving imaging inverse problems.
- Classical regularizations enforce some kind of image regularity prior.
(Tychonov regularization promotes smooth images; TV promotes cartoon images.)
- We have seen convergence guarantees for gradient descent.
- We have introduced the proximal operator of a convex function.
- We introduced the idea of Plug-and-Play methods for image restoration.

THANK YOU FOR YOUR ATTENTION!

References I

-  Bauschke, H. H. and Combettes, P. L. (2011).
Convex Analysis and Monotone Operator Theory in Hilbert Spaces.
Springer.
-  Buades, A., Coll, B., and Morel, J.-M. (2005).
A non-local algorithm for image denoising.
In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), volume 2, pages 60–65. IEEE.
-  Daubechies, I. (1988).
Orthonormal bases of compactly supported wavelets.
Communications on Pure and Applied Mathematics, 41(7):909–996.
-  Daubechies, I., Defrise, M., and De Mol, C. (2004).
An iterative thresholding algorithm for linear inverse problems with a sparsity constraint.
Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences, 57(11):1413–1457.
-  Dossal, C., Hurault, S., and Papadakis, N. (2024).
Optimization with first order algorithms.

References II

-  Gatys, L. A., Ecker, A. S., and Bethge, M. (2015).
Texture synthesis using convolutional neural networks.
In Advances in Neural Information Processing Systems, pages 262–270.
-  Gatys, L. A., Ecker, A. S., and Bethge, M. (2016).
Image style transfer using convolutional neural networks.
In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2414–2423.
-  Hurault, S., Leclaire, A., and Papadakis, N. (2021).
Gradient step denoiser for convergent plug-and-play.
In International Conference on Learning Representations.
-  Johnson, J., Alahi, A., and Fei-Fei, L. (2016).
Perceptual losses for real-time style transfer and super-resolution.
In Computer Vision – ECCV 2016, pages 694–711, Cham. Springer International Publishing.
-  Louchet, C. and Moisan, L. (2014).
Total variation denoising using iterated conditional expectation.
In 2014 22nd European Signal Processing Conference (EUSIPCO), pages 1592–1596. IEEE.

References III

-  [Mallat, S. \(1989\).](#)
A theory for multiresolution signal decomposition: the wavelet representation.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 11(7):674–693.
-  [Mallat, S. \(2009\).](#)
A Wavelet Tour of Signal Processing, The Sparse Way.
Academic Press, Elsevier, 3rd edition edition.
-  [Ongie, G., Jalal, A., Metzler, C. A., Baraniuk, R. G., Dimakis, A. G., and Willett, R. \(2020\).](#)
Deep learning techniques for inverse problems in imaging.
IEEE Journal on Selected Areas in Information Theory, 1(1):39–56.
-  [Romano, Y., Elad, M., and Milanfar, P. \(2017\).](#)
The little engine that could: Regularization by denoising (RED).
SIAM Journal on Imaging Sciences, 10(4):1804–1844.
-  [Rudin, L. I., Osher, S., and Fatemi, E. \(1992\).](#)
Nonlinear total variation based noise removal algorithms.
Physica D: nonlinear phenomena, 60(1-4):259–268.

References IV

-  Simonyan, K. and Zisserman, A. (2015).
Very deep convolutional networks for large-scale image recognition.
In Bengio, Y. and LeCun, Y., editors, *Proceedings of the International Conference on Learning Representations*.
-  Venkatakrisnan, S. V., Bouman, C. A., and Wohlberg, B. (2013).
Plug-and-play priors for model based reconstruction.
In *2013 IEEE Global Conference on Signal and Information Processing*, pages 945–948. IEEE.
-  Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004).
Image quality assessment: from error visibility to structural similarity.
IEEE transactions on image processing, 13(4):600–612.
-  Yu, G., Sapiro, G., and Mallat, S. (2011).
Solving inverse problems with piecewise linear estimators: From gaussian mixture models to structured sparsity.
IEEE Transactions on Image Processing, 21(5):2481–2499.

References V

-  Zhang, K., Li, Y., Zuo, W., Zhang, L., Van Gool, L., and Timofte, R. (2021). Plug-and-play image restoration with deep denoiser prior.
IEEE Transactions on Pattern Analysis and Machine Intelligence.
-  Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018). The unreasonable effectiveness of deep features as a perceptual metric.
In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
-  Zoran, D. and Weiss, Y. (2011). From learning models of natural image patches to whole image restoration.
In 2011 International Conference on Computer Vision, pages 479–486. IEEE.