

Problem Set 3: Total Variation by Subgradient, Proximal, and Splitting Methods

ECE-491 Machine Learning for Medical Imaging
Fall 2026

Assigned: Week 7 **Due:** start of class, Week 9

Instructions. Submit a single PDF containing your derivations, figures, and discussion, together with the code that generated every figure. Use Python and DeepInverse 0.5 (<https://deepinv.org>, `pip install deepinv`); a laptop CPU is enough. Each problem starts with a short DeepInverse walkthrough. Use DeepInverse for the imaging physics, the finite differences, and the metrics, but implement yourself any algorithm a part asks you to implement.

The common problem. All three problems solve

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} F(\mathbf{x}), \quad F(\mathbf{x}) = \frac{1}{2} \|\mathbf{Ax} - \mathbf{y}\|_2^2 + \lambda \text{TV}(\mathbf{x}),$$

with $\mathbf{A} = \mathbf{I}$ (denoising), $\mathbf{A} = \mathbf{MF}$ (MRI), and $\mathbf{A}$ = the Radon transform (CT). Let $\mathbf{D}$ map an image to its vertical and horizontal forward differences, so that $(\mathbf{Dx})[n] \in \mathbb{R}^2$ is the discrete gradient at pixel n . The isotropic total variation is

$$\text{TV}(\mathbf{x}) = \|\mathbf{Dx}\|_{1,2} = \sum_n \|(\mathbf{Dx})[n]\|_2,$$

which is convex but not differentiable. Each problem compares the subgradient method, proximal gradient descent (PGD), and ADMM, and asks which of their steps become expensive as $\mathbf{A}$ loses structure.

Problem 1 (TV Denoising). Denoise a two-photon fluorescence microscopy image ($\mathbf{A} = \mathbf{I}$). The TV denoiser you build here is reused in Problem 2.

DeepInverse walkthrough.

```
import torch, deepinv as dinv
x = dinv.utils.load_example("FMD_TwoPhoton_MICE_R_gt_12_avg50.png", grayscale=True)
y = dinv.physics.Denoising(dinv.physics.GaussianNoise(sigma=0.1))(x)
psnr = dinv.metric.PSNR()          # psnr(y, x) is about 20 dB

tv = dinv.optim.TVPrior()
Dx = tv.nabla(x)                   # D x, shape (1,1,512,512,2)
tv.nabla_adjoint(Dx)               # D^T
tv.fn(x)                           # TV(x)
tv.grad(x)                         # a subgradient of TV at x
tv.prox(y, gamma=0.1)             # prox of 0.1*TV at y (DeepInverse's TV denoiser)
```

- (a) *Subgradients.* A vector $\mathbf{g}$ is a subgradient of a convex function h at $\mathbf{v}$, written $\mathbf{g} \in \partial h(\mathbf{v})$, if $h(\mathbf{w}) \geq h(\mathbf{v}) + \mathbf{g}^\top (\mathbf{w} - \mathbf{v})$ for all $\mathbf{w}$.
- (i) Show that for $h = \|\cdot\|_2$, $\partial h(\mathbf{v}) = \{\mathbf{v}/\|\mathbf{v}\|_2\}$ if $\mathbf{v} \neq \mathbf{0}$, and $\partial h(\mathbf{0}) = \{\mathbf{g} : \|\mathbf{g}\|_2 \leq 1\}$. (Hint: Cauchy–Schwarz.)
- (ii) Using the rule $\partial(h \circ \mathbf{D})(\mathbf{x}) = \mathbf{D}^\top \partial h(\mathbf{Dx})$, show that $\mathbf{D}^\top \mathbf{q}$ is a subgradient of TV at $\mathbf{x}$, where $\mathbf{q}[n] = (\mathbf{Dx})[n]/\|(\mathbf{Dx})[n]\|_2$ if $(\mathbf{Dx})[n] \neq \mathbf{0}$ and $\mathbf{q}[n] = \mathbf{0}$ otherwise. (This is `tv.grad`.) Write down a subgradient of F .

- (b) *The subgradient method* $\mathbf{x}_{k+1} = \mathbf{x}_k - t_k \mathbf{g}_k$, $\mathbf{g}_k \in \partial F(\mathbf{x}_k)$. Take $h(x) = |x|$, a constant step t , and $x_0 = t/2$. Show that the iterates alternate between $\pm t/2$ forever. Why does the majorization-minimization argument of Problem Set 2 not apply?
- (c) *The subgradient method in practice*. Sweep λ on a log scale using `tv.prox(y, gamma=lam)`, plot PSNR against λ , and use the best value λ^* from here on. Compute a reference optimum F^* from `dinv.models.TVDenoiser(crit=1e-10, n_it_max=5000)(y, ths=lam)`. Implement the subgradient method from $\mathbf{x}_0 = \mathbf{y}$ and run 300 iterations with $t = 0.05$, $t_k = 0.5/\sqrt{k+1}$, and $t_k = 1/(k+1)$. Because $F(\mathbf{x}_k)$ need not decrease, plot the best value so far, $(\min_{j \leq k} F(\mathbf{x}_j) - F^*)/F^*$, against k on log-log axes. Which step rule works best, and why? (Hint: $\frac{1}{2}\|\mathbf{x} - \mathbf{y}\|_2^2$ is strongly convex.)
- (d) *PGD on the dual*. PGD on F itself is pointless: one step of size 1 returns $\text{prox}_{\lambda \text{TV}}(\mathbf{y})$, the very thing we want. Instead, apply PGD to the dual.

- (i) Using that $\|\cdot\|_2$ is its own dual norm (Problem Set 2), show that

$$\text{TV}(\mathbf{x}) = \max_{\mathbf{p} \in B} \langle \mathbf{p}, \mathbf{D}\mathbf{x} \rangle, \quad B = \{\mathbf{p} : \|\mathbf{p}[n]\|_2 \leq 1 \text{ for every } n\}.$$

- (ii) Swap min and max in

$$\min_{\mathbf{x}} \max_{\mathbf{p} \in B} \frac{1}{2}\|\mathbf{x} - \mathbf{y}\|_2^2 + \lambda \langle \mathbf{D}^\top \mathbf{p}, \mathbf{x} \rangle$$

and minimize over $\mathbf{x}$ in closed form. Show that $\mathbf{x} = \mathbf{y} - \lambda \mathbf{D}^\top \mathbf{p}$, and that the dual problem is $\min_{\mathbf{p} \in B} \frac{1}{2}\|\mathbf{y} - \lambda \mathbf{D}^\top \mathbf{p}\|_2^2$.

- (iii) This dual objective has a gradient that is Lipschitz with constant $8\lambda^2$. Show that the prox of the constraint is the projection $\mathbf{p}[n] \leftarrow \mathbf{p}[n]/\max(1, \|\mathbf{p}[n]\|_2)$, and write out the PGD iteration on $\mathbf{p}$.
- (e) *ADMM*. Split $\mathbf{w} = \mathbf{D}\mathbf{x}$. Scaled-form ADMM with penalty $\rho > 0$ iterates
- $$\begin{aligned} \mathbf{x}_{k+1} &= \arg \min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x} - \mathbf{y}\|_2^2 + \frac{\rho}{2}\|\mathbf{D}\mathbf{x} - \mathbf{w}_k + \mathbf{u}_k\|_2^2, \\ \mathbf{w}_{k+1} &= \arg \min_{\mathbf{w}} \lambda\|\mathbf{w}\|_{1,2} + \frac{\rho}{2}\|\mathbf{D}\mathbf{x}_{k+1} - \mathbf{w} + \mathbf{u}_k\|_2^2, \\ \mathbf{u}_{k+1} &= \mathbf{u}_k + \mathbf{D}\mathbf{x}_{k+1} - \mathbf{w}_{k+1}. \end{aligned}$$
- (i) Show that the $\mathbf{x}$ -update solves $(\mathbf{I} + \rho \mathbf{D}^\top \mathbf{D})\mathbf{x} = \mathbf{y} + \rho \mathbf{D}^\top(\mathbf{w}_k - \mathbf{u}_k)$. Solve it with a few conjugate-gradient iterations, warm-started at $\mathbf{x}_k$.
- (ii) Show that the $\mathbf{w}$ -update acts pixel by pixel as the *group soft-threshold* $\mathbf{w}[n] = \max(0, 1 - \tau/\|\mathbf{v}[n]\|_2)\mathbf{v}[n]$, with $\mathbf{v} = \mathbf{D}\mathbf{x}_{k+1} + \mathbf{u}_k$ and $\tau = \lambda/\rho$. (Hint: the minimizer points along $\mathbf{v}[n]$; find its length.)
- (f) *Comparison*. Implement dual PGD as a function `tv_denoise(v, lam, n_iter, p0)` returning $(\mathbf{x}, \mathbf{p})$ (you will reuse it), and ADMM with $\rho \in \{1, 4, 16\}$ and 5 CG iterations per $\mathbf{x}$ -update (`dinv.optim.linear.conjugate_gradient`). Run 300 iterations of each. Plot $(F(\mathbf{x}_k) - F^*)/F^*$ against iteration and against wall time, together with the subgradient runs, and show the reconstructions with their PSNRs. Rank the methods. How does ρ affect ADMM?

Problem 2 (TV MRI Reconstruction). Reconstruct a single-coil brain slice from $4 \times$ undersampled k-space, $\mathbf{A} = \mathbf{M}\mathbf{F}$ with $\mathbf{F}$ the unitary 2D DFT and $\mathbf{M}$ a 0/1 mask. DeepInverse stores complex images as two real channels; display the magnitude `x.pow(2).sum(1).sqrt()`.

DeepInverse walkthrough.

```

x = dinv.utils.load_example("demo_mini_subset_fastmri_brain_0.pt") # (1,2,320,320)
gen = dinv.physics.generator.GaussianMaskGenerator(
    (320, 320), acceleration=4, rng=torch.Generator().manual_seed(1))
mask = gen.step(batch_size=1)["mask"]
physics = dinv.physics.MRI(mask=mask, img_size=(320, 320),
    noise_model=dinv.physics.GaussianNoise(sigma=0.02))
y = physics(x)
x_zf = physics.A_adjoint(y) # zero-filled reconstruction

prior, fid = dinv.optim.TVPrior(), dinv.optim.L2() # L2 is 1/2 ||Ax - y||^2
pgd = dinv.optim.PGD(prior=prior, data_fidelity=fid, lambda_reg=lam,
    stepsize=1.0, max_iter=100)
x_hat, m = pgd(y, physics, init=x_zf, x_gt=x, compute_metrics=True)
m["psnr"], m["cost"] # per-iteration PSNR and F(x_k)
# Similarly: dinv.optim.GD(..., stepsize=[t_0, t_1, ...]) and
# dinv.optim.ADMM(..., stepsize=gamma), with init=(x_zf, 0*x_zf).

```

- (a) *PGD*. Let $f(\mathbf{x}) = \frac{1}{2} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2$. Since $\|\mathbf{A}\|_2 = 1$, the step $\gamma = 1$ is valid. Show that the gradient step is then $\mathbf{x} - \mathbf{A}^H(\mathbf{A}\mathbf{x} - \mathbf{y}) = \mathbf{F}^H((\mathbf{I} - \mathbf{M})\mathbf{F}\mathbf{x} + \mathbf{M}\mathbf{y})$, with $\mathbf{y}$ stored as full k-space (zeros where not acquired): it overwrites the acquired k-space of $\mathbf{x}$ with the data. PGD alternates this *data consistency* with the TV denoiser of Problem 1.
- (b) *ADMM*. DeepInverse's ADMM splits off a copy of $\mathbf{x}$ instead of $\mathbf{D}\mathbf{x}$:

$$\begin{aligned}
\mathbf{u}_{k+1} &= \text{prox}_{\gamma f}(\mathbf{x}_k - \mathbf{z}_k), \\
\mathbf{x}_{k+1} &= \text{prox}_{\gamma \lambda \text{TV}}(\mathbf{u}_{k+1} + \mathbf{z}_k), \\
\mathbf{z}_{k+1} &= \mathbf{z}_k + \mathbf{u}_{k+1} - \mathbf{x}_{k+1}.
\end{aligned}$$

Show that

$$\text{prox}_{\gamma f}(\mathbf{v}) = \mathbf{F}^H(\mathbf{I} + \gamma \mathbf{M})^{-1} \mathbf{F}(\mathbf{v} + \gamma \mathbf{A}^H \mathbf{y}),$$

so it costs two FFTs. What does it do at acquired and at unacquired k-space locations?

- (c) *Comparison*. Report the PSNR of $\mathbf{x}_{\text{ZF}}$. Sweep λ with 60 iterations of PGD and pick λ^* (use `TVPrior(def_crit=1e-5)` to make the sweep fast). From $\mathbf{x}_{\text{ZF}}$, run 100 iterations of the subgradient method ($t_k = 1/\sqrt{k+1}$), PGD ($\gamma = 1$),¹ and ADMM ($\gamma \in \{1, 5\}$). Each PGD or ADMM run takes about two minutes on a CPU. Plot PSNR and $F(\mathbf{x}_k) - F^*$ against iteration and wall time, with F^* the smallest cost reached, and show $|\mathbf{x}|$, $|\mathbf{x}_{\text{ZF}}|$, $|\hat{\mathbf{x}}|$, and the mask.
- (d) *Inexact proximal steps*. `TVPrior.prox` runs up to 1000 inner iterations at every outer iteration. Write your own PGD loop whose prox is your `tv_denoise` with only $n_{\text{in}} \in \{1, 5, 20\}$ iterations, warm-started with the previous $\mathbf{p}$. Compare with the built-in PGD in cost, PSNR, and wall time. Is it worth solving the prox exactly?

Problem 3 (TV Sparse-View CT). Reconstruct a CT slice from 60 parallel-beam projections ($\mathbf{A}$ = Radon transform, normalized so that $\|\mathbf{A}\|_2 = 1$).

DeepInverse walkthrough.

```

x = dinv.utils.load_example("CT100_256x256_0.pt") # (1,1,256,256)
physics = dinv.physics.Tomography(angles=60, img_width=256, normalize=True,
    noise_model=dinv.physics.GaussianNoise(sigma=0.01))

```

¹(optional) Try `dinv.optim.FISTA`, which adds momentum to PGD.

<code>y = physics(x)</code>	<code># sinogram</code>
<code>x_fbp = physics.A_dagger(y, fbp=True)</code>	<code># filtered backprojection</code>
<code>physics.max_iter, physics.tol = 5, 1e-12</code>	<code># CG budget of ADMM's data step</code>

(a) *What changes.*

- (i) ADMM's data step still solves $(\mathbf{I} + \gamma \mathbf{A}^\top \mathbf{A})\mathbf{x} = \mathbf{v} + \gamma \mathbf{A}^\top \mathbf{y}$. Why is there no FFT shortcut for CT, as there was for MRI? The system is solved by CG, where each iteration costs one $\mathbf{A}$ and one $\mathbf{A}^\top$.
- (ii) Count the applications of $\mathbf{A}$ and $\mathbf{A}^\top$ per iteration of the subgradient method, PGD, and ADMM with m CG iterations.

(b) *Comparison.* Compute the filtered backprojection (FBP) baseline. Sweep λ with PGD as in Problem 2. From $\mathbf{x}_{\text{FBP}}$, run 60 iterations of the subgradient method ($t_k = 1/\sqrt{k+1}$), PGD ($\gamma = 1$), and ADMM ($\gamma = 1$) with $m \in \{2, 5, 20\}$. Plot PSNR against iteration and against the number of applications of $\mathbf{A}$ or $\mathbf{A}^\top$, and show $\mathbf{x}$, $\mathbf{x}_{\text{FBP}}$, and your best TV reconstruction.²

(c) *Wrap-up.* For each of the three problems, which algorithm would you choose, and why? Name the step that was cheap or expensive, and the property of $\mathbf{A}$ responsible.

²(optional) Attenuation is nonnegative. How would you add the constraint $\mathbf{x} \geq \mathbf{0}$ to PGD and ADMM, and does it help?