20 Total Variation and Image Processing

Lab Objective: *Minimizing an energy functional is equivalent to solving the resulting Euler-Lagrange equations. We introduce the method of steepest descent to solve these equations, and apply this technique to a denoising problem in image processing.*

The Gradient Descent method

Consider an energy functional $J[u]$, defined over a collection of admissible functions $u : \Omega \subset \mathbb{R}^n \to \mathbb{R}$, with the form

$$J[u] = \int_\Omega L(x, u, \nabla u)\, dx$$

where $L = L(x, u, \nabla u)$ is a function $\mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$. A standard result from the calculus of variations states that a minimizing function u^* satisfies the Euler-Lagrange equation

$$L_u - \sum_{i=1}^n \frac{\partial L_{u_{x_i}}}{\partial x_i} = L_u - \nabla \cdot L_{\nabla u} = L_u - \operatorname{div}(L_{\nabla u}) = 0. \tag{20.1}$$

where $L_{\nabla u} = \nabla' L = [L_{x_1}, \ldots, L_{x_n}]^\intercal$.

This equation is typically an elliptic PDE, possessing boundary conditions associated with restrictions on the class of admissible functions u. To more easily compute (20.1), we consider a related parabolic PDE,

$$\begin{aligned} u_t &= -(L_u - \operatorname{div} L_{\nabla u}), \quad t > 0, \\ u(x, 0) &= u_0(x), \quad t = 0. \end{aligned} \tag{20.2}$$

A steady state solution of (20.2) does not depend on time, and thus solves the Euler-Lagrange equation. It is often easier to evolve an initial guess using (20.2), and stop whenever its steady state is well-approximated, than to solve (20.1) directly.

Example 20.1. Consider the energy functional

$$J[u] = \int_\Omega \|\nabla u\|^2\, dx.$$

1

The minimizing function u^* satisfies the Euler-Lagrange equation

$$-\text{div}\,\nabla u = -\Delta u = 0.$$

The gradient descent flow is the well-known heat equation

$$u_t = \Delta u.$$

■

The Euler-Lagrange equation could equivalently be described as $\Delta u = 0$, leading to the PDE $u_t = -\Delta u$. Since the backward heat equation is ill-posed, it would not be helpful in a search for the steady-state.

Let us take the time to make (20.2) more rigorous. We recall that

$$\begin{aligned}
\delta J(u; h) &= \left.\frac{d}{dt}J(u + \epsilon h)\right|_{\epsilon=0}, \\
&= \int_\Omega (L_u(u) - \text{div}\,L_{\nabla u}(u))h\,dx, \\
&= \langle L_u(u) - \text{div}\,L_{\nabla u}(u), h\rangle_{L^2(\Omega)},
\end{aligned}$$

for each u and each admissible perturbation h. Then using the Cauchy-Schwarz inequality,

$$|\delta J(u; h)| \leq \|L_u(u) - \text{div}\,L_{\nabla u}(u)\| \cdot \|h\|$$

with equality iff $h = \alpha(L_u(u) - \text{div}\,L_{\nabla u}(u))$ for some $\alpha \in \mathbb{R}$. This implies that the "direction" $h = L_u(u) - \text{div}\,L_{\nabla u}(u)$ is the direction of steepest ascent and maximizes $\delta J(u; h)$. Similarly,

$$h = -(L_u(u) - \text{div}\,L_{\nabla u}(u))$$

points in the direction of steepest descent, and the flow described by (20.2) tends to move toward a state of lesser energy.

Minimizing the area of a surface of revolution

The area of the surface obtained by revolving a curve $y(x)$ about the x-axis is

$$A[y] = \int_a^b 2\pi y\sqrt{1 + (y')^2}\,dx.$$

To minimize the functional A over the collection of smooth curves with fixed end points $y(a) = y_a$, $y(b) = y_b$, we use the Euler-Lagrange equation

$$\begin{aligned}
0 &= 1 - y\frac{y''}{1 + (y')^2}, \\
&= 1 + (y')^2 - yy'',
\end{aligned} \tag{20.3}$$

with the gradient descent flow given by

$$\begin{aligned}
u_t &= -1 - (y')^2 + yy'', \quad t > 0, \, x \in (a, b), \\
u(x, 0) &= g(x), \quad t = 0, \\
u(a, t) &= y_a, \quad u(b, t) = y_b.
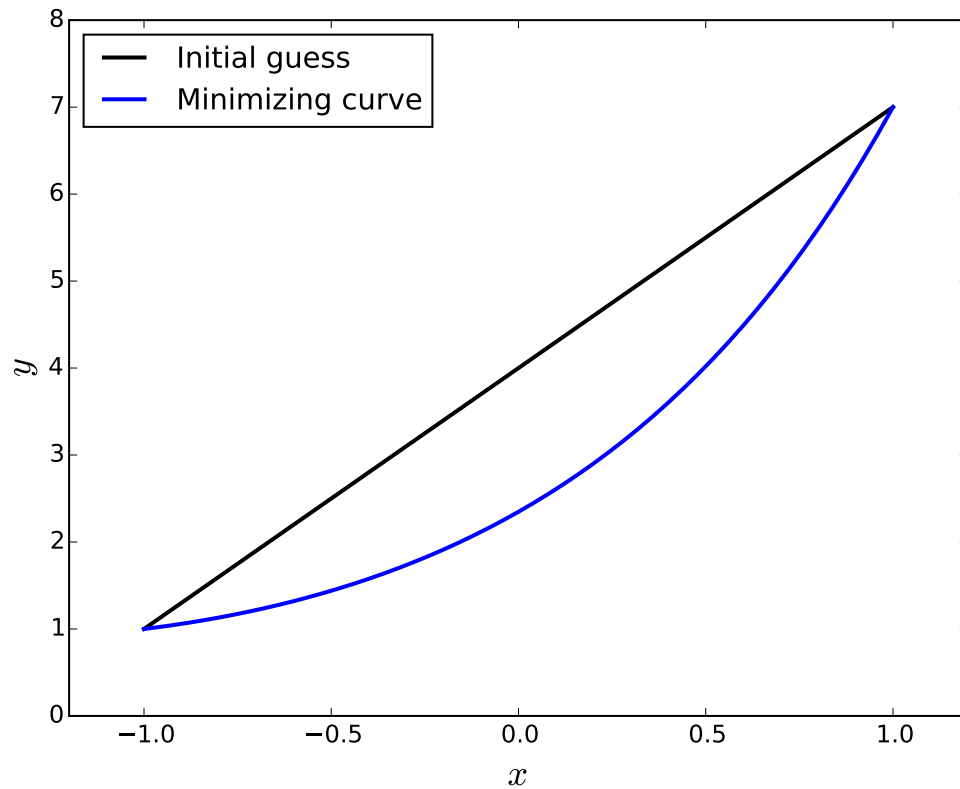\end{aligned} \tag{20.4}$$

Figure 20.1: The solution of (20.3), found using the gradient descent flow (20.4).

When solving this numerically, use the centered difference first- and second-order approximations

$$y'(x_i) = \frac{y(x_{i+1}) - y(x_{i-1})}{2\triangle x} \tag{20.5}$$

$$y''(x_i) = \frac{y(x_{i+1}) - 2y(x_i) + y(x_{i-1})}{(\triangle x)^2} \tag{20.6}$$

Problem 1. Construct a numerical solution of (20.4) using the conditions $y(-1) = 1$, $y(1) = 7$. Use 20 x steps, 250 time steps, and a final time of .2.

Most numerical schemes have a stability condition that must be satisfied. Our discretization requires that $\frac{\triangle t}{(\triangle x)^2} \leq \frac{1}{2}$. Check that this condition is satisfied.

Let the straight line connecting the endpoints serve as an initial guess, u. Iterate through the time steps, updating u according to

```
u+=dt*u_t
```

until the norm of the difference between successive iterations is less than 10^{-5}. Keep in mind the endpoints should not update.

Plot the solution that minimizes (20.4). It should match figure 20.1.

Image Processing: Denoising

A greyscale image can be represented by a scalar-valued function $u : \Omega \to \mathbb{R}$, $\Omega \subset \mathbb{R}^2$. A color image can be represented by three functions u_1, u_2, and u_3. In this lab we will work with black and white images, but total variation techniques can easily be used on more general images.

A simple approach to image processing

Here is a first attempt at denoising: given a noisy image f, we look for a denoised image u minimizing the energy functional

$$J[u] = \int_\Omega L(x, u, \nabla u)\, dx, \tag{20.7}$$

where

$$L(x, u, \nabla u) = \frac{1}{2}(u - f)^2 + \frac{\lambda}{2}|\nabla u|^2,$$
$$= \frac{1}{2}(u - f)^2 + \frac{\lambda}{2}(u_x^2 + u_y^2)^2.$$

This energy functional penalizes 1) images that are too different from the original noisy image, and 2) images that have large derivatives. The minimizing denoised image u will balance these two different costs.

Solving for the original denoised image u is a difficult inverse problem-some information is irretrievably lost when noise is introduced. However, a priori information can be used to guess at the structure of the original image. For example, here λ represents our best guess on how much noise was added to the image, and is known as a regularization parameter in inverse problem theory.

The Euler-Lagrange equation corresponding to (20.7) is

$$L_u - \text{div } L_{\nabla u} = (u - f) - \lambda \Delta u,$$
$$= 0.$$

and the gradient descent flow is

$$u_t = -(u - f - \lambda \Delta u),$$
$$u(x, 0) = f(x). \tag{20.8}$$

Let u_{ij}^n represent our approximation to $u(x_i, y_j)$ at time t_n. We will approximate u_t with a forward Euler difference, and Δu with centered differences:

$$u_t \approx \frac{u_{ij}^{n+1} - u_{ij}^n}{\triangle t},$$
$$u_{xx} \approx \frac{u_{i+1,j}^n - 2u_{ij}^n + u_{i-1,j}^n}{(\triangle x)^2},$$
$$u_{yy} \approx \frac{u_{i,j+1}^n - 2u_{ij}^n + u_{i,j-1}^n}{(\triangle y)^2}.$$

Problem 2. Let u be the grayscale image `balloons_bw.jpg`; remember to scale it when importing. Add noise using the following code.

```python
from scipy.misc import imread, imsave

imagename = 'balloons_bw.jpg'
changed_pixels=40000
# Read the image file imagename into an array of numbers, IM
# Multiply by 1. / 255 to change the values so that they are floating ↩
    point
# numbers ranging from 0 to 1.
IM = imread(imagename, flatten=True) * (1. / 255)
IM_x, IM_y = IM.shape

for lost in xrange(changed_pixels):
    x_,y_ = random_integers(1,IM_x-2), random_integers(1,IM_y-2)
    val = .1*randn() + .5
    IM[x_,y_] = max( min(val,1.), 0.)
```

Using $\triangle t = 1e-3$, $\lambda = 40$, $\triangle x = 1$, and $\triangle y = 1$, implement the numerical scheme mentioned above to obtain a denoised solution u. (So $\Omega = [0, n_x] \times [0, n_y]$, where n_x and n_y represent the number of pixels in the x and y dimensions, respectively.) Take 250 steps in time. Compare your results with Figure 20.3.

Hint: Use the function `np.roll` to compute the spatial derivatives. For example, the second derivative can be approximated at interior grid points using

```python
u_xx = np.roll(u,-1,axis=1) - 2*u + np.roll(u,1,axis=1)
```

Image Processing: Total Variation Method

We represent an image by a function $u : [0, 1] \times [0, 1] \to \mathbb{R}$. A C^1 function $u : \Omega \to \mathbb{R}$ has bounded total variation on Ω ($BV(\Omega)$) if $\int_{\Omega} |\nabla u| < \infty$; u is said to have total variation $\int_{\Omega} |\nabla u|$. Intuitively, the total variation of an image u increases when noise is added.

The total variation approach was originally introduced by Ruding, Osher, and Fatemi[1]. It was formulated as follows: given a noisy image f, we look to find a denoised image u minimizing

$$\int_{\Omega} |\nabla u(x)| \, dx \tag{20.9}$$

subject to the constraints

$$\int_{\Omega} u(x) \, dx = \int_{\Omega} f(x) \, dx, \tag{20.10}$$

$$\int_{\Omega} |u(x) - f(x)|^2 \, dx = \sigma |\Omega|. \tag{20.11}$$

---

[1]L. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms", *Physica D.*, 1992.
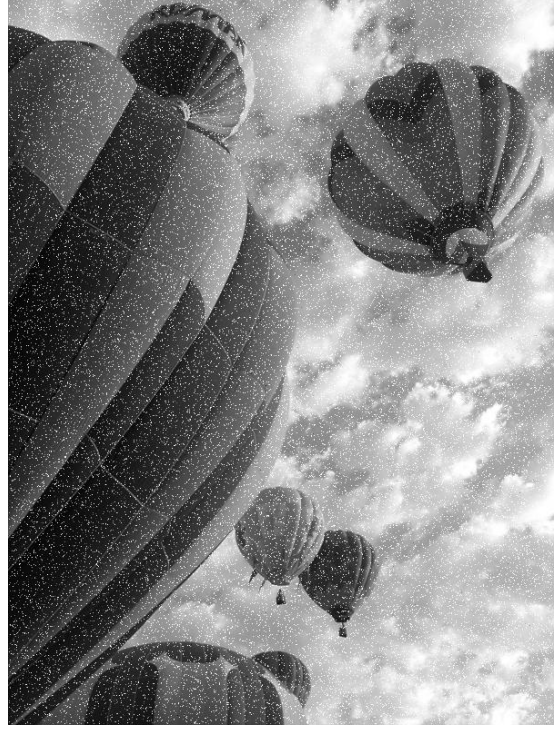
Original image   Image with white noise

Figure 20.2: Noise.

Intuitively, (20.9) penalizes fast variations in f - this functional together with the constraint (20.10) has a constant minimum of $u = \frac{1}{|\Omega|} \int_\Omega u(x)\,dx$. This is obviously not what we want, so we add a constraint (20.11) specifying how far $u(x)$ is required to differ from the noisy image f. More precisely, (20.10) specifies that the noise in the image has zero mean, and (20.11) requires that a variable σ be chosen a priori to represent the standard deviation of the noise.

Chambolle and Lions proved that the model introduced by Rudin, Osher, and Fatemi can be formulated equivalently as

$$F[u] = \min_{u \in BV(\Omega)} \int_\Omega |\nabla u| + \frac{\lambda}{2}(u - f)^2\,dx, \tag{20.12}$$

where $\lambda > 0$ is a fixed regularization parameter[2]. Notice how this functional differs from (20.7): $\int_\Omega |\nabla u|$ instead of $\int_\Omega |\nabla u|^2$. This turns out to cause a huge difference in the result. Mathematically, there is a nice way to extend F and the class of functions with bounded total variation to functions that are discontinuous across hyperplanes. The term $\int |\nabla|$ tends to preserve edges/boundaries of objects in an image.

The gradient descent flow is given by

$$u_t = -\lambda(u - f) + \frac{u_{xx}u_y^2 + u_{yy}u_x^2 - 2u_x u_y u_{xy}}{(u_x^2 + u_y^2)^{3/2}}, \tag{20.13}$$

$$u(x,0) = f(x).$$

---

[2]A. Chambolle and P.-L. Lions, "Image recovery via total variation minimization and related problems", *Numer. Math.*, 1997.

Initial diffusion-based approach   Total variation based approach

Figure 20.3: The solutions of (20.8) and (20.13), found using a first order Euler step in time and centered differences in space.

Notice the singularity that occurs in the flow when $|\nabla u| = 0$. Numerically we will replace $|\nabla u|^3$ in the denominator with $(\epsilon + |\nabla u|^2)^{3/2}$, to remove the singularity.

Problem 3. Using $\triangle t = 1e - 3, \lambda = 1, \triangle x = 1$, and $\triangle y = 1$, implement the numerical scheme mentioned above to obtain a solution u. Take 200 steps in time. Compare your results with Figure 20.3. How small should ϵ be?

Hint: To compute the spatial derivatives, consider the following:

```
u_x = (np.roll(u,-1,axis=1) -  np.roll(u,1,axis=1))/2
u_xx = np.roll(u,-1,axis=1) - 2*u + np.roll(u,1,axis=1)
u_xy = (np.roll(u_x,-1,axis=0) - np.roll(u_x,1,axis=0))/2.
```