Optimization Methods*

November 22, 2019

1 Gradient Descent Methods

1.1 Gradient Descent for Unconstrained Problems

We consider the problem of finding a minimum of a function f, hence solving

$$\min_{x \in \mathbb{R}^d} f(x)$$

where $f : \mathbb{R}^d \mapsto \mathbb{R}$ is a smooth function.

The minimum is not necessarily unique. In the general case, f might exhibit local minima, in which case the proposed algorithms are not expected to find a global minimizer of the problem. In this tour, we restrict our attention to convex function, so that the methods will converge to a global minimizer.

The simplest method is the gradient descent, that computes

$$x^{(k+1)} = x^{(k)} - \tau_k \nabla f\left(x^{(k)}\right)$$

where $\tau_k > 0$ is a step size, and $\nabla f(x) \in \mathbb{R}^d$ is the gradient of f at the point x, and $x^{(0)} \in \mathbb{R}^d$ is an initial point.

In the convex case, if f is of class $\mathcal{C}^2$, in order to ensure convergence, the step size should satisfy

$$0 < \tau_k < \frac{2}{\sup_x \|Hf(x)\|}$$

where $Hf(x) \in \mathbb{R}^{d \times d}$ is the Hessian of f at x and $\|\cdot\|$ is the spectral operator norm (largest eigenvalue).

The following code takes $f(x) = x^2$, computes the gradient manually and applies gradient descent to get to the solution $x = 0$.

```
tau = 2e-1;                          % Step-size parameter
f = @(x)(x.^2);                         % Function to minimize
fgrad = @(x)(2.*x);                  % Gradient
```

*These are notes based on Gabriel Peyré's exceptional Numerical Tours and are for my own personal study.

```
x0 = -500;                % Initial guess
tol = 1e-6;                % Tolerance of the algorithm
err = 10000;        % Initial error
it = 1;                          % Iteration counter

while err > tol
        fgradx0 = fgrad(x0);
        x1(it+1) = x0-tau.*fgradx0;
        err = abs(fgradx0); % Error is the absolute value of the gradient
        if err > tol
                x0 = x1(it+1);
        end
        fprintf('New x = %3.3f \n',x0)
        it = it+1;
end
```

1.2 Gradient Descent in 2-D

Suppose we want to minimize the following quadratic form

$$f(x) = \frac{1}{2}\left(x_1^2 + \eta x_2^2\right)$$

where η controls the anisotropy and, hence, the difficulty of the problem.[1] Let us set $\eta = 10$.

The rationale for the code is the same as before. However, now I impose two stopping conditions, one for each coordinate. Note that the step-size parameter τ_k needs to be smaller than $2/\eta$. Figure

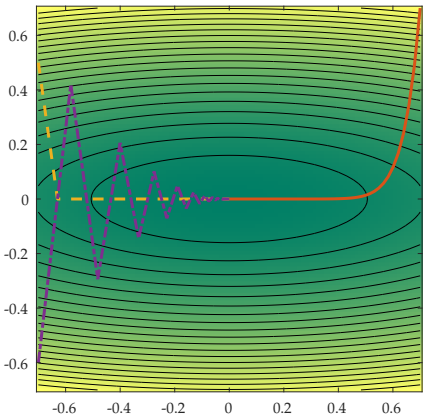

Figure 1: Visualization of Gradient Descent in $2-$D for Different Values of τ_k

[1] Anisotropy is the property of being directionally dependent, which implies different properties in different directions, as opposed to isotropy.

1.3 Gradient and Divergence of Images

Local differential operators like gradient, divergence and laplacian are the building blocks for variational image processing.

Gradient An image is a matrix $x_0 \in \mathbb{R}^N$ of $N = n \times n$ pixels. For a continuous function g, the gradient reads

$$\nabla g(s) = \left(\frac{\partial g(s)}{\partial s_1}, \frac{\partial g(s)}{\partial s_2} \right) \in \mathbb{R}^2$$

(note that here, the variable s denotes the $2-$D spatial position).

We discretize this differential operator on a discrete image $x \in \mathbb{R}^N$ using first order finite differences.

$$(\nabla x)_i = (x_{i_1,i_2} - x_{i_1-1,i_2}, x_{i_1,i_2} - x_{i_1,i_2-1}) \in \mathbb{R}^2$$

Note that for simplity we use periodic boundary conditions. Thus, we get $\nabla : \mathbb{R}^n \mapsto \mathbb{R}^{N \times 2}$. Figure 2 shows the discretized gradient in two images. The first row of figures shows a simple graph with two squares. Notice how the dx shows a vertical lines denoting changes from black into white and the other way around, while the dy figure shows horizontal lines denoting the change from black into white. The second row shows the same but for a real image.

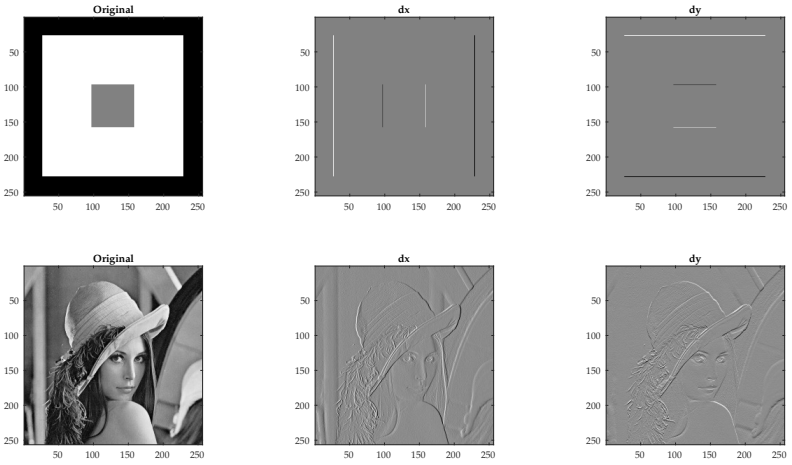

Figure 2: Gradient Descent in a 2-D Image

Magnitude The magnitude is just defined as the Euclidean norm, i.e. $\|\nabla g\| = \sqrt{\left(\frac{\partial g(s)}{\partial s_1}\right)^2 + \left(\frac{\partial g(s)}{\partial s_2}\right)^2}$. The magnitude is larger closer to the edges of the image. Figure 3 shows the Euclidean norm or the magnitude for the image. This highlights borders where changes of color or transparency are sharper.

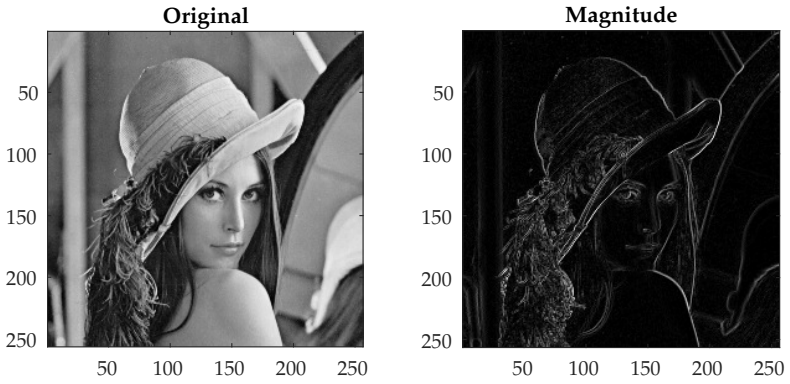

Figure 3: Magnitude in a 2-D Image

Divergence Operator The divergence operator maps vector field to images. For continuous vector fields $v(s) \in \mathbb{R}^2$, it is defined as

$$\mathrm{div}(v)(s) = \frac{\partial v_1(s)}{\partial s_1} + \frac{\partial v_2(s)}{\partial s_2} \in \mathbb{R}$$

(note that here, the variable s denotes the 2-D spatial position). It is minus the adjoint of the gadient, i.e. $\mathrm{div} = -\nabla^*$.

It is discretized, for $v = (v_1, v_2)$ as

$$\mathrm{div}(v)_i = v^1_{i_1+1,i_2} - v^1_{i_1,i_2} + v^2_{i_1,i_2+1} - v^2_{i_1,i_2}$$

It is a local measure of its *"outgoingness"*, i.e. the extent to which there is more of the field vectors exiting an infinitesimal region of space than entering it. A point at which the flux is outgoing has positive divergence, and is often called a *"source"* of the field. A point at which the flux is directed inward has negative divergence, and is often called a *"sink"* of the field. The greater the flux of field through a small surface enclosing a given point, the greater the value of divergence at that point. A point at which there is zero flux through an enclosing surface has zero divergence.

Laplacian Operator The Laplacian operator is defined as $\Delta = \mathrm{div} \circ \nabla = -\nabla^* \circ \nabla$. That is, the divergence of the gradient and denotes the rate at which the average value of an image g over spheres centered at s deviates from $g(s)$ as the radius of the spheres shrinks towards 0. Basically it is the sum of the second order partial derivatives.

Figure 4 shows the differences between the gradient and the Laplacian operators in our sample image.

2 Newton's Method for Unconstrained Problems

Let us start with a highly anisotropic function, the Rosenbrock function given by

$$g(x) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$$

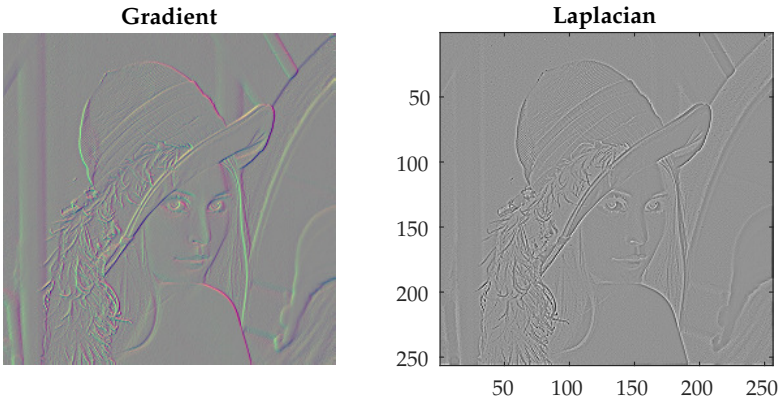

Figure 4: Gradient, and Laplacian Operators

where at $x^* = (1,1)$ the function reaches its minimum with $g(x^*) = 0$. Figure 5 shows the behavior of the function on a $3-$D grid.

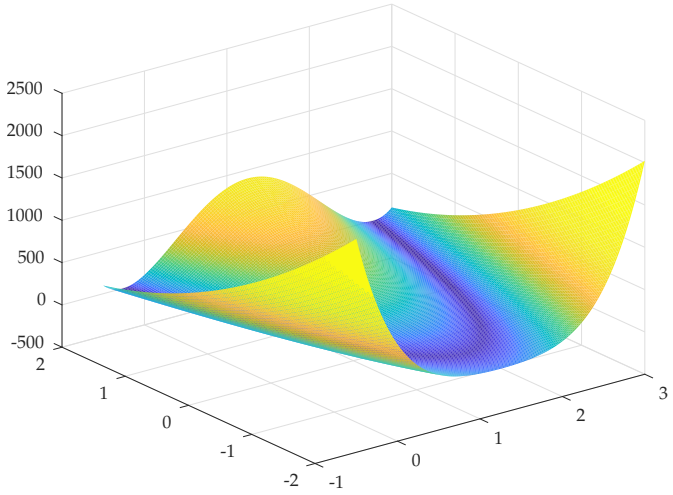

Figure 5: Rosenbrock Function

If we would use gradient descent methods that only use first order gradient information about g, we would not be able to efficiently minimize the function because of the high anisotropy. Defining the gradient of g as

$$\nabla g(x) = \left(\frac{\partial g(x)}{\partial x_1}, \frac{\partial g(x)}{\partial x_2} \right) = \left(2(x_1 - 1) + 400x_1(x_1^2 - x_2), 200(x_2 - x_1^2) \right) \in \mathbb{R}^2$$

The Hessian matrix can be computed as

$$\mathcal{H}g(x) = \begin{pmatrix} \frac{\partial^2 g(x)}{\partial x_1^2} & \frac{\partial^2 g(x)}{\partial x_1 \partial x_2} \\ \frac{\partial^2 g(x)}{\partial x_1 \partial x_2} & \frac{\partial^2 g(x)}{\partial x_2^2} \end{pmatrix} = \begin{pmatrix} 2 + 400(x_1^2 - x_2) + 800x_1^2 & -400x_1 \\ -400x_1 & 200 \end{pmatrix} \in \mathbb{R}^{2\times 2}$$

The Newton descent method starting from some $x^{(0)} \in \mathbb{R}^2$ is computed by iterating on

$$x^{(\ell+1)} = x^{(\ell)} - \left[\mathcal{H}g\left(x^{(\ell)}\right) \right]^{-1} \nabla g\left(x^{(\ell)}\right)$$

Intuition of Newton's Method in 1 Dimension Suppose we have a single variable function $f(x)$, the method tries to find the roots of $f'(x)$ by constructing a sequence $x^{(\ell)}$ from an initial guess $x^{(0)}$ that converges towards some value x^* satisfying $f'(x^*) = 0$. Taking a second-order Taylor expansion $f_T(x)$ of $f(x)$ around $x^{(\ell)}$:

$$f_T(x) = f_T(x^{(\ell)} + \Delta x) \approx f(x^{(\ell)}) + f'(x^{(\ell)})\Delta x + \frac{1}{2}f''(x^{(\ell)})\Delta x^2$$

We want to pick Δx such that $x^{(\ell)} + \Delta x$ is a stationary point of f. Using the Taylor expansion, we can solve for Δx corresponding to the root of the expansion's derivative:

$$\Delta x = -\frac{f'(x^{(\ell)})}{f''(x^{(\ell)})}$$

Provided the Taylor expansion approximation is fairly accurate, then incrementing by Δx should yield a point close enough to an actual stationary point of f. This point is given by

$$x^{(\ell+1)} = x^{(\ell)} + \Delta x = x^{(\ell)} - \frac{f'(x^{(\ell)})}{f''(x^{(\ell)})}$$

For a two-dimensional problem, we just replace the first derivative by the gradient $\nabla f(x)$ and the reciprocal of the second derivative by the inverse of the Hessian matrix $\left[\mathcal{H}g\left(x^{(\ell)}\right)\right]^{-1}$.