Optimization Methods*

November 20, 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

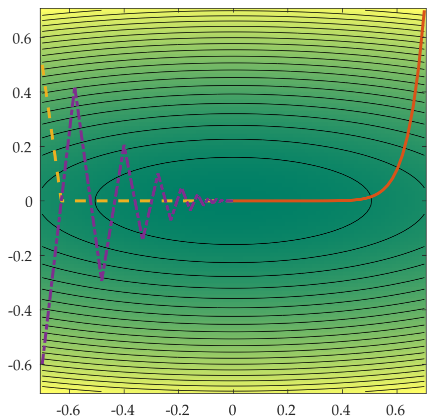

Figure 1: Visualization of Gradient Descent in 2−D for Different Values of τ_k

[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.

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 on an image. The left-hand side shows the first component of the gradient, while the right-hand side one shows the second component of the gradient.

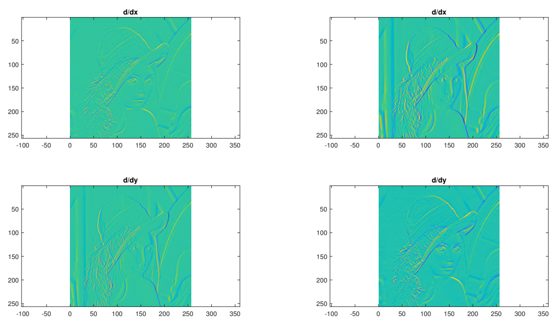

Figure 2: Gradient Descent in a 2-D Image

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 spacial position). It is minus the adjoint of the gadient, i.e. $\mathrm{div} = -\nabla^*$.

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

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