Proximal Methods

Ran Wang

June 2, 2021

Chapter 1

Proximal Algorithms

1.1 Motivation

In this chapter, we briefly outline the problem of minimizing functions that are not necessarily differentiable. A typical example is the l_1-regularized problem. For example, the object might look like

$$\min_{\beta} \sum_{i=1}^{N} (y_i - x_i^t \beta)^2 + \lambda \|\beta\|_1.$$

Here, β is the parameter we want to find. Should we not have $\lambda\|\beta\|_1$, then everything is differentiable and can be solved use quasi-Newton methods, among other things. However, the absolute value function is not differentiable everywhere, which causes problems.

The first solution is by consider *sub-differentials*.Sub-differentials are defined as

$$\partial f(x) = \left\{ y \mid f(z) \geq f(x) + y^T(z - x) \text{ for all } z \in \operatorname{dom} f \right\},$$

where dom$f$ is the domain of the function. Note that if a function is differentaible then $\partial f = \{\triangledown f\}$. However, in general case, the sub-sdifferential is not a singleton.

For simplicity, we assume all the functions we discuss are subdifferentiable.

1.2 Proximal Algorithms

The proximal operator is defined as

$$\operatorname{prox}_f(v) = \operatorname*{argmin}_{x} \left(f(x) + (1/2)\|x - v\|_2^2 \right).$$

As simple as the definition might look like, it has quite some nice results. The first one is a fixed-point properties. That is, the point $x^\star$ minimizes f if and only if

$$x^\star = \mathrm{prox}_f\left(x^\star\right).$$

Proof. First we show that if x^* is the minimizer, then $x^\star = \mathrm{prox}_f\left(x^\star\right)$. Note that for any x,

$$f(x) + (1/2)\left\|x - x^\star\right\|_2^2 \geq f\left(x^\star\right) = f\left(x^\star\right) + (1/2)\left\|x^\star - x^\star\right\|_2^2,$$

and thus by definition, $x^\star = \mathrm{prox}_f\left(x^\star\right)$.

Now consider the reverse case, let $\tilde{x} = \mathrm{prox}_f(v)$. Take the subdifferential operator, we see that this is equivalent to

$$0 \in \partial f(\tilde{x}) + (\tilde{x} - v).$$

Taking $\tilde{x} = v = x^\star,$ it follows that $0 \in \partial f\left(x^\star\right),$ so $x^\star$ minimizes f. □

The second interesting, and rather surprising fact is that, the proximal operator is actually the resolvent of subdifferetial operator. More specifically,

$$\mathrm{prox}_{\lambda f} = (I + \lambda\partial f)^{-1}.$$

Proof. If $z \in (I + \lambda\partial f)^{-1}(x)$, then $0 \in \partial f(z) + (1/\lambda)(z - x)$. This implies $0 \in \partial_z\left(f(z) + (1/2\lambda)\|z - x\|_2^2\right)$. Now, since we can prove that $f(z) + (1/2\lambda)\|z - x\|_2^2$ is strongly convex, we can deduce that $z = \underset{u}{\mathrm{argmin}}\left(f(u) + (1/2\lambda)\|u - x\|_2^2\right)$. □

Finally, let us look at the case of minimzing $f + g$ where f is differentiable and g is not. A famous algorithms goes,

$$x^{k+1} := \mathrm{prox}_{\lambda^k g}\left(x^k - \lambda^k\nabla f\left(x^k\right)\right).$$

To see why, consider the fixed point version that is $x^* = \mathrm{prox}_{\lambda g}\left(x^* - \lambda\nabla f\left(x^*\right)\right)$, we show that if this is true then x is indeed the solution to

$$x^* = \mathrm{argmin}_x f(x) + g(x).$$

Proof. Note that x^* is the minimzer if and only if $0 \in \nabla f\left(x^\star\right) + \partial g\left(x^\star\right)$. Now with some straight forward computation,

$$\begin{aligned}
0 &\in \lambda\nabla f\left(x^\star\right) + \lambda\partial g\left(x^\star\right)\\
0 &\in \lambda\nabla f\left(x^\star\right) - x^\star + x^\star + \lambda\partial g\left(x^\star\right)\\
(I + \lambda\partial g)\left(x^\star\right) &\ni (I - \lambda\nabla f)\left(x^\star\right)\\
x^\star &= (I + \lambda\partial g)^{-1}(I - \lambda\nabla f)\left(x^\star\right)\\
x^\star &= \mathrm{prox}_{\lambda g}\left(x^\star - \lambda\nabla f\left(x^\star\right)\right)
\end{aligned}$$

□

1.3 Applications

Now let us return to the previous topics. Let us assume that the **negative** log-likelihood function is $f_\beta(X)$, here β is the parameter we want to estimate and X is the data, which include both the dependent and independent variables. The goal now is the minimize the following quantities.

$$f_\beta(X) + \lambda\|\beta\|_1,$$

where $\lambda > 0$ is a positive hyperparemeter, $\|\cdot\|$ is the l_1 norm, namely for any $x \in R^N$, $\|x\|_1 = \sum_{i=1}^N |x_i|$. Since the latter term is not differentiable at 0, we are in position to workout the solution as is indicated in the previous section.

Now let $w := \beta^k - \lambda^k \nabla f_{\beta^k}(X)$, where λ^k is the step-size, and β^k is the value of β at step k, and let $g : x \mapsto \lambda\|x\|_1$, we have

$$\begin{aligned}\beta^{k+1} &:= \text{prox}_{\lambda^k g}(w) \\ &= \text{argmin}_x \left(\lambda^k g(x) + \frac{1}{2}\|x-w\|_2^2\right) \\ &= \text{argmin}_x \left(\lambda\|x\|_1 + \frac{1}{2\lambda_k}\|x-w\|_2^2\right)\end{aligned}$$

Now it suffices to calculate the for each i, since none of i depends on others. For that, note that (**exercises!**) we have

$$\underset{z}{\text{argmin}}\frac{1}{2}\|\beta - z\|_2^2 + \lambda t\|z\|_1 = S_{\lambda t}(\beta)$$

where we have

$$[S_\lambda(\beta)]_i = \begin{cases} \beta_i - \lambda & \text{if } \beta_i > \lambda \\ 0 & \text{if } -\lambda \le \beta_i \le \lambda \\ \beta_i + \lambda & \text{if } \beta_i < -\lambda \end{cases}$$

Therefore we have

$$\left[\beta^{k+1}\right]_i = \begin{cases} w_i - \lambda\lambda_k & \text{if } w_i > \lambda\lambda_k \\ 0 & \text{if } -\lambda\lambda_k \le w_i \le \lambda\lambda_k \\ w_i + \lambda\lambda_k & \text{if } w_i < -\lambda\lambda_k \end{cases}$$

With that, one can easily implement the algorithm.

1.4 Improving the algorithm

Unfortunately, the algorithm itself isn't the fastest possible. One way to implement the algorithm is by backtracking line search. The basic idea of backtracking line search is to avoid computing the gradients as much as possible. Loosely speaking, this means once the gradient is computed, one tries to avoid the re-computation of the gradient by "going through the same direction several times". More specifically, backtracking means one starts by going a "large" step while shrinking the steps each time until the termination condition is met.

Specifically, let us define the following quantities: $G_t(x) = \frac{x - \mathrm{prox}_t(x - t\nabla g(x))}{t}$. To see why we would like to define this quantity, note that the updating equation

$$x^{k+1} := \mathrm{prox}_{\lambda^k g}\left(x^k - \lambda^k \nabla f\left(x^k\right)\right).$$

is equivalent to

$$x^{k+1} := x^k - \lambda^k G_{\lambda^k}\left(x^k\right).$$

Now, fix an shrinkage parameter $\beta \in (0,1)$, let the initial step size $t = 1$, then while

$$g\left(x - tG_t(x)\right) > g(x) - t\nabla g(x)^T G_t(x) + \frac{t}{2}\left\|G_t(x)\right\|_2^2.$$

Shrink t by β, i.e. $t := \beta t$.