10-725/36-725: Convex Optimization **Fall 2018**

Lecture 8: September 24

Lecturer: Lecturer: Ryan Tibshirani *Scribes: Scribes: Allie Chang*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

8.1 Proximal Gradient Descent

Suppose $f(x)$ is decomposable:

$$f(x) = g(x) + h(x) \tag{8.1}$$

Where g is convex, differentiable, $dom(g) = \mathbb{R}^n$ and h is convex, but not necessary differentiable. When h is differentiable we can simply compute gradient and do gradient descent. We can do quadratic approximation to

$$x^+ = argmin_z f(x) + \bigtriangledown f(x)^T (z - x) + \frac{1}{2t}||z - x||_2^2 \tag{8.2}$$

Where t represents the step size and the weight for quadratic term. If we apply this quadratic approximation to g and keep h the same, we get:

$$x^+ = argmin_z \frac{1}{2t}||z - (x - t \bigtriangledown g(x))||_2^2 + h(x) \tag{8.3}$$

The idea behind this is to stay close to gradient update for g and also make h small. This function is defined as proximal mapping. Rewrite as follows:

$$prox_t(x) = argmin_z \frac{1}{2t}||x - z||_2^2 + h(x) \tag{8.4}$$

This function has unique solution because the square term is strictly convex and $h(x)$ is convex. So proximal gradient descent is just repeat following steps:

$$x^{(k)} = prox_{t_h}(x^{(k-1)} - t_k \bigtriangledown g(x^{(k-1)})), k = 1, 2, 3, ... \tag{8.5}$$

Let $G_t(x) = \frac{x - prox_t(x - t\bigtriangledown g(x))}{t}$ be the generalized gradient, we can rewrite above equation in familiar, gradient descent way:

$$x^{(t)} = x^{(k-1)} - t_k G_{t_k}(x^{(k-1)}) \tag{8.6}$$

For many h (ex: L_1 norm)the $prox_t(.)$ has closed-form solution. Besides, $prox_t(.)$ doesn't depend on g.We tend to use this method when h is cheap. ISTA is the problem of using proximal gradient descent to solve lasso. Please refer to the slides for matrix completion example.

8.2 Backtracking Line Search

We can use backtracking to select step size for proximal gradient descent. It's similar to gradient descent, just replacing the gradient term with generalized gradient G_t. Choose $0 < \beta < 1$, for each iteration, while

$$g(x - tG_t(x)) > g(x) - t \bigtriangledown g(x)^T G_t(x) + \frac{t}{2}||G_t(x)||_2^2 \tag{8.7}$$

shrink $t = \beta t$. Otherwise do proximal gradient update.

8.3 Convergence Analysis

Proximal gradient descent with fixed step size $t \leq \frac{1}{L}$ satisfies

$$f(x^{(k)}) - f^* \leq \frac{||x^{(0)} - x^*||_2^2}{2tk} \tag{8.8}$$

Where $t = \beta/L$ when doing backtracking. So proximal gradient descent has convergence rate $O(1/k)$ or $O(1/\epsilon)$, which is the same as gradient descent. But we need to consider prox cost too.

8.4 Special cases

Proximal gradient descent is also called composite gradient descent or "generalized" gradient descent. It's call "generalized" because:

- $h = 0$: $prox_t(x) = x$, same as gradient descent
- $h = I_C$: projected gradient descent
- $g = 0$: proximal minimization algorithm

8.4.1 Projected gradient descent

We can show that when h is I_C, this becomes projected gradient descent:

$$_t(x) = argmin_z \frac{1}{2t}||x - z||_2^2 + I_C(z) = argmin_{z \in C}||x - z||_2^2 = P_C(x) \tag{8.9}$$

So after each update, it projects the solution back to set C for further updates. Notice that the distances between projections is no bigger than the distances between original values. No new convergence analysis needed.

$$x^+ = P_C(x - t \bigtriangledown g(x)) \tag{8.10}$$

8.4.2 Proximal minimization algorithm

When $g = 0$, gradient of g is also zero, so the update is just

$$x^+ = argmin_z \frac{1}{2t}||x - z||_2^2 + h(z) \tag{8.11}$$

Sounds great, but can only used when we know the prox form of h.

In practice, if we cannot evaluate $prox_t$, we can consider to approximate it if we know how to control the error.

8.5 Acceleration

Acceleration can improve the convergence rate to $O(1/\sqrt{\epsilon})$. Nesterov published a series of paper for acceleration methods. Choose initial point $x^{(0)} = x^{(-1)}$, repeat:

$$v = x^{(k-1)} + \frac{k-2}{k+1}(x^{(k-1)} - x^{(k-2)}) \tag{8.12}$$

$$x^{(k)} = prox_{t_k}(v - t_k \bigtriangledown g(v)) \tag{8.13}$$

It pushed the update using momentum from previous iterations. If $h = 0$ it is accelerated gradient method. The momentum weight start's from zero and increases as k grows, approaching to 1. As we get closer to optimality, the gradient is expected to be smaller, so does the update, by using larger momentum weights with large k, it keeps the optimization going forward without slowing down.