10-725: Optimization **Fall 2013**

Lecture 8: September 19

Lecturer: Ryan Tibshirani *Scribes: Qin Gao, Siheng Chen*

8.1 Subgradient method

In the last lecture, we talked about the most general method probably in the whole semester which is subgradient method which is applicable whenever you can compute subgradient which is very often. It's a very general method that's why we like it. We have a function f that is convex but not necessarily differentiable. It looks just like gradient descent we just take an initial guess and then we repeatly from that initial guess we compute the subgradient and then we update within the direction of the negative subgradient,

$$x^{(k)} = x^{(k-1)} - t_k \cdot g^{(k-1)} \tag{8.1}$$

where $g^{(k-1)}$ is the subgradient of f at $x^{(k-1)}$. It is not a descent method necessarily so we have to keep track of the best function value it has seen. We stop at some number of iterations we have kept track of that which point gives us the best function value we call that our solution or approximate solution.

8.1.1 Step size choices

In subgradient method rather than gradient decent for example we do not compute the step size adaptively based on the data we see but just fix it ahead of time. One rule is to keep constant step size. The other rule is to choose them to satisfy these conditions which we summarize as square summable but not summable,

$$\sum_{k=1}^{\infty} t_k^2 < \infty, \sum_{k=1}^{\infty} t_k = \infty \tag{8.2}$$

An example of that is just take $t_k = 1/k$. There's no analogy of backtracking in the subgradient methods.

8.1.2 Convergence analysis

Here is the results we know about subgradient methods. The first result is that if we take a constant step size then the limit of our function value at the best point convergent to something suboptimal,

$$\lim_{k\to+\infty} f(x_{best}^{(k)}) \leq f(x^*) + G^2 t/2 \tag{8.3}$$

where t is our step size and G is the Lipschitz constant of our function, where for any x and y,

$$|f(x) - f(y)| \leq G||x - y||_2 \tag{8.4}$$

More than that if we take step size is diminishing in the sense that their squares are summable but they are not summable themselves then we actually get the solution in the limit.

$$\lim_{k\to+\infty} f(x^{(k)}_{best}) = f(x^*) \tag{8.5}$$

The one thing we talked about in the end of our last lecture was that it's important to keep in mind when we take about which functions are Lipschitz that was a helpful exercise I encourage you to whenever we talk about a condition in class think about when do it hold. What function can you think of that break the assumption or meet them. What I just want to add that as a footnote was that if you look at the proof of these results, it really only needs the function to be Lipschitz on a bounded set, basically the set that all our points are within. Keep that always challenge the assumptions that we have for any of these theorems. That's a very useful thing to do. It's nice to have a mathematical result but it's more important to think about when it apply and when they don't.

8.1.3 Polyak step sizes

There's one more choice of step size which is called Polyakov step size, where the optimal value $f(x^*)$ is known,

$$t_k = \frac{f(x^{(k-1)}) - f(x^*)}{||g^{(k-1)}||_2^2}, k = 1, 2, 3, ... \tag{8.6}$$

Polyakov step sizes are nice because they give us basically the convergence rate of the subgradient method which we don't really have so far. They are impractical in the sense that when we apply them we need to know $f(x^*)$. But if we can use that step size that will give us convergence rate. In practice some people use this step size by estimating $f(x^*)$ which is maybe depending on the problem it could be a lot more easier to estimate $f(x^*)$ than estimate x^* itself which is what we are doing. When we plug this in and go through the steps we get a complexity of $1/\sqrt{k}$ in contrast with gradient decent where we get $1/k$. The translation of that is if we want to compute an estimate so that we are within ϵ of the minimized function value that we need O($1/\epsilon^2$) iterations. Comparing with gradient decent where we need $1/\epsilon$ iterations. That can be substantially smaller.

Example: find intersection of sets. We want to find the intersection of the convex and closed sets C_1 through C_n. In order to find this point we want to propose it to be a minimization problem where we define the function,

$$f(x) = \max_{i=1,...m} dist(x, C_i), \tag{8.7}$$

and minimize it,

$$\min_{x\in R^n} f(x). \tag{8.8}$$

Let's break it down into smaller pieces. Is the distance function $dist(x, C_i)$ a convex function? It first is a convex function of x and y since L_2 norm is convex. We are minimizing over one block of those variables over convex set C thus is convex. Now we are looking at the maximum of $f_i(x)$, i going through 1 to m. That's convex because that's the maximum of convex functions. So this is a convex function. If we can compute subgradient, presumably we can apply subgradient method and then we find eventually some x^* that's in the intersection of all those sets. Go through the slides if you are interested.

8.1.4 Intersection of sets

Now I want to mention a even more general version of subgradient method which will handle convex functions over constraint sets. If we have a convex function we want to minimize over C which is some convex set, we could apply subgradient method and then every step project back into C that's called projected subgradient method. If we just apply the usual subgradient updates, at every iteration we must make sure we stay in C. So after we compute the update we project into C. We get the exact same convergence guarantee with this.

8.1.5 Project subgradient method

Another way to minimize over a convex set C with subgradient is write this as min over all x of $f(x)+I_C(x)$. The subgradient of $I_C(x)$ is normal cone.

There are a lot of sets that are easy to project on to and there are a lot that are hard as well. Things like the Affine image, the solution set of linear system, the nonnegative orthant (points that are componentwise nonnegative), Norm balls for the case $p = 2, \inf$ is very easy for the case $p = 1$ is possible and some simple polyhedral and simple cones.

- For nonnegative orthant, $P(x_i) = max(x_i, 0)$;
- For linear system $C = \{x : Ax = b\}$, $P_C(x_i) = X + A^T(AA^T)^{-1}(b - Ax)$ where we assumed A has full row rank;
- For basic pursuit problem, we solved it with simplex in a linear program form. Here we solve it with projected subgradient method. The problem is,

$$\min_{\beta \in R^n} ||\beta||_1 \textit{ subject to } X\beta = y. \tag{8.9}$$

  Hence the projected subgradient method repeats,

$$\begin{aligned} \beta^{(k)} &= P_C(\beta^{(k-1)} - t_{ks}^{(k-1)}) \\ &= \beta^{(k-1)} - t_k(I - X^T(XX^T)^{-1}X)s^{(k-1)} \end{aligned} \tag{8.10}$$

  where $s^{(k-1)}$ is subgradient of $||\beta^{(k-1)}||_1$, which is

$$s_i^{(k-1)} \in \begin{cases} sign(\beta^{(k-1)}) & \beta_i^{(k-1)} \neq 0; \\ [-1, 1] & \text{otherwise.} \end{cases}$$

  We then have an algorithm to solve the basis pursuit problem. There's no algorithm that generically perform better than $1/\sqrt{k}$ rate (Nesterov theorm). So there's nothing that is generically faster than subgradient method.

Theorem 8.1 *Nesterov theorem: For any $k \leq n - 1$ and starting point $x^{(0)}$, there is a function in the problem class such that any nonsmooth first-order method satisfies*

$$f(x^{(k)}) - f(x^*) \geq \frac{RG}{2(1 + \sqrt{k+1})} \tag{8.11}$$

We are going to consider function that can be composite to be part that is convex and differentiable and part that's nonsmooth but simple.

8.2 Generalized gradient descent

8.2.1 Algorithm

Suppose a objective function $f(x)$ can be decomposed as

$$f(x) = g(x) + h(x),$$

where $g(x)$ is convex and differentiable and $h(x)$ is convex and not necessarily differentiable.

If f is differentiable, gradient descent can be applied to the whole function. Since f is not differentiable, we can approximate the differentiable part $g(x)$ as

$$\hat{g}(y) = g(x) + \nabla g(x)^T(y-x) + \frac{1}{2t}||y-x||^2.$$

We then obtain the approximation of the objective function as

$$\begin{aligned} f(y) &= g(y) + h(y) \\ &\approx \hat{g}(y) + h(y) \\ &= g(x) + \nabla g(x)^T(y-x) + \frac{1}{2t}||y-x||^2 + h(y). \end{aligned}$$

The next step can be updated as:

$$\begin{aligned} x^+ &= \arg\min_y g(x) + \nabla g(x)^T(y-x) + \frac{1}{2t}||y-x||^2 + h(y) \\ &= \arg\min_y \frac{1}{2t}||x-y-t\nabla g(x)||^2 + h(y) \end{aligned}$$

Define

$$prox_{h,t}(x) = \arg\min_z \frac{1}{2t}||x-z||^2 + h(z)$$

The generalized gradient descent updates as:

$$\begin{aligned} x^+ &= prox_{h,t}(x - t\nabla g(x)) \\ &= x - tG_t(x) \qquad (8.12) \end{aligned}$$

where $G_t(x) = (x - prox_{h,t}(x - t\nabla g(x)))/t$. We claim that (8.12) is the generalized gradient descent and $G_t(x)$ is the generalized gradient.

Note that:

- $prox_t$ does not depend on g at all;
- g can be very complicated as long as we can compute its gradient.

8.2.2 Example - Iterative Soft-Thresholding Algorithm (ISTA)

To select variables, we consider lasso criterion as follows:

$$f(\beta) = \frac{1}{2}||y - X\beta||^2 + \lambda||\beta||_1.$$

We then consider that $g(\beta) = \frac{1}{2}||y - X\beta||^2$ and $h(\beta) = \lambda||\beta||_1$. Prox function is:

$$\begin{aligned} prox_t(\beta) &= \arg\min_z \frac{1}{2t}||\beta - z||^2 + \lambda||z||_1 \\ &= S_{t\lambda}(\beta) \end{aligned}$$

The updating equation is:

$$\begin{aligned} \beta^{(k)} &= prox_t(\beta^{(t)} - t_k \nabla g(\beta^{(k-1)})) \\ &= S_{t\lambda}(\beta^{(t)} - t_k \nabla g(\beta^{(k-1)})) \\ &= S_{t\lambda}(\beta^{(t)} + t_k X^T(y - X\beta^{(k-1)}) \end{aligned}$$

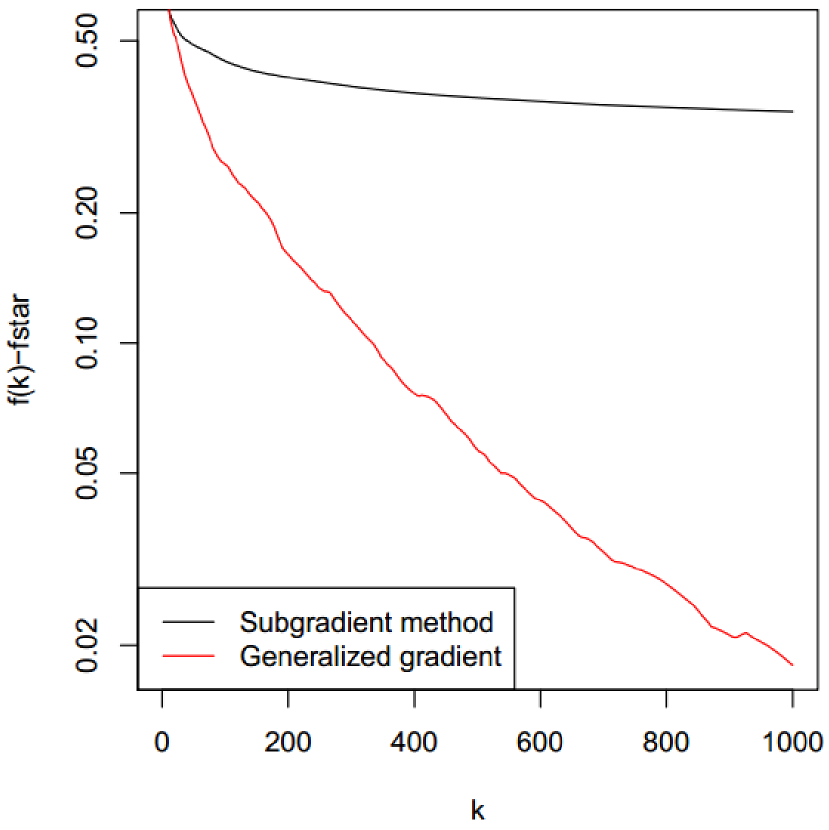

Figure 8.1: Generalized gradient (ISTA) vs subgradient descent.

8.2.3 Convergence analysis

Theorem 1 *We have $f(x) = g(x) + h(x)$, and assume that g is convex, differentiable, ∇g is Lipschitz continuous with constant $L > 0$; h is convex, $prox_t(x) = \arg\min_z\{||x - z||^2/(2t) + h(z)\}$ can be evaluated. Generalized gradient descent with fixed step size $t \leq 1/L$ satisfies*

$$f(x^{(k)}) - f(x^*) \leq \frac{||x^{(0)} - x^*||^2}{2tk} \tag{8.13}$$

Surprisingly, generalized gradient descent has convergence rate $O(1/k)$, which is same as gradient descent! But remember, this counts the number of iterations, not the number of operations