10-725/36-725: Convex Optimization **Fall 2016**

Lecture 8: September 26

Lecturer: Lecturer: Ryan Tibshirani Scribes: Scribes: Ramakumar Pasumarthi, Maruan Al-Shedivat

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

8.1 Subgradient Method (contd.)

Consider a convex f, with $dom(f) \in \mathbb{R}^n$, but is not necessarily differentiable. This motivates us to consider using a gradient descent-like update method, which uses subgradients instead.

Update step:

$$x^(k) = x^{(k-1)} - t_k.g^{(k-1)}, k = 1, 2, 3, ...$$

where, $g^{(k-1)} \in \partial f(x^{(k-1)})$.

Subgradient method is not necessarily a descent method, so we keep track of best iterate $x^{(k)}$ among $x^{(0)}, ...x^{(k)}$ so far:

$$f(x^{(k)}) = \min_{i=0,..,k} f(x^{(i)})$$

8.1.1 Choice of step size

Step sizes are pre-specified, unlike in gradient descent.

- Fixed step size: $t_k = t \forall k$
- Diminishing step sizes: step sizes that meet the following condition

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

  This essentially says that, the step sizes go to zero, but not too fast.

8.1.2 Convergence Analysis

Assume that f convex, $dom(f) = \mathbb{R}^n$, and also that f is Lipschitz continuous with constant $G > 0$, i.e., $|f(x) - f(y)| \leq G||x - y||_2 \forall x, y$

Theorem 8.1 *For a fixed step size t, subgradient method satisfies*

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

Theorem 8.2 *For diminishing step sizes, subgradient method satisfies*

$$\lim_{k\to\infty} f(x_{best}^{(k)}) = f^*$$

Proof: These can be proven using the following property of subgradients:

$$||x^{(k)} - x^*||^2 \leq ||x^{(k-1)} - x^*||^2 - 2tk(f(x^{(k-1)}) - f(x^*)) + t_k^2||g^{(k-1)}||_2^2$$

The first term of the RHS can be expanded using the same inequality. Doing so iteratively, we get

$$||x^{(k)} - x^*||^2 \leq ||x^{(0)} - x^*||^2 - \sum_{i=1}^{k} 2ti(f(x^{(i-1)}) - f(x^*)) + \sum_{i=1}^{k} t_i^2||g^{(i-1)}||_2^2$$

$0 \leq$ RHS. Using this and substituting $R^2 = ||x^{(0)} - x^*||^2$, and sending the second term to LHS, we get

$$\sum_{i=1}^{k} 2ti(f(x^{(i-1)}) - f(x^*)) \leq R^2 + \sum_{i=1}^{k} t_i^2||g^{(i-1)}||_2^2$$

Using $f(x_{best}^{(k)}) - f(x^*) \leq (f(x^{(i-1)}) - f(x^*)$, and $||g^{(i-1)}||_2^2 \leq G$ (from the Lipschitz condition), we get

$$(f(x_{best}^{(k)}) - f(x^*)) \sum_{i=1}^{k} 2t_i \leq R^2 + G^2 \sum_{i=1}^{k} t_i^2$$

$$f(x_{best}^{(k)}) - f(x^*) \leq \frac{R^2 + G^2 \sum_{i=1}^{k} t_i^2}{\sum_{i=1}^{k} 2t_i}$$

From this, both theorem 1 and 2 will follow. In case of theorem 2, note that RHS tends to 0, and that LHS is always non-negative. This leads to the equality in theorem 2. The above basic inequality will be useful later in proving the convergence rate to be $O(1/\epsilon^2)$. ■

8.1.3 Convergence Rate

Subgradient method can be shown to have convergence rate of $O(1/\epsilon^2)$, which is slower than $O(1/\epsilon)$ observed for gradient descent.

Using the basic inequality shown before, for a fixed step size t, we have

$$f(x_{best}^{(k)}) - f^* \leq \frac{R^2}{2kt} + \frac{G^2 t}{2}$$

To bound this value to be less than ϵ, we set each term to be less than $\epsilon/2$. This gives $t = \epsilon/G$, and $k = R^2/t.\epsilon = R^2G^2/\epsilon^2$.

Hence the convergence rate is $O(1/\epsilon^2)$.

8.1.4 Polyak step sizes

When the optimal value f^* is known, take $t_k = \frac{f(x^{(k-1)})-f^*}{|g^(k-1)|_2^2}, k = 1, 2, 3, ...$

This can be motivated from the following subgradient property,

$$||x^{(k)} - x^*||^2 \leq ||x^{(k-1)} - x^*||^2 - 2tk(f(x^{(k-1)}) - f(x^*)) + t_k^2||g^{(k-1)}||_2^2$$

Polyak step size minimizes the right-hand side. This can be seen by taking the derivative w.r.t t_k of RHS and setting it to 0.

Polyak steps can be shown to converge to optimal value, with the same converge rate: $O(1/\epsilon^2)$.

8.1.5 Example: intersection of sets

Given closed, convex sets $C_1, C_2, ..., C_m$, we want to find

$$x^* \in \cap_{i=1}^m C_i$$

To formulate this, we first define

$$f_i(x) = dist(x, C_i)$$

$$f(x) = \max_{i=1,..,m} f_i(x)$$

Solve for

$$\min_x f(x)$$

$$f(x^*) = 0 \implies x^* \in \cap_{i=1}^m C_i$$

Recall that the distance function $dist(x, C) = min_{y \in C}||y - x||_2$. The gradient

$$\nabla dist(x, C) = \frac{x - P_C(x)}{||x - P_C(x)||_2}$$

where $P_C(x)$ is the projection of x onto C.

Recall the subgradient rule for maximum of functions. The subdifferential is the convex hull of the union of subdifferentials of i, whenever i is the maximal value.

$$\partial f(x) = conv(\bigcup_{i:f_i(x)=f(x)} \partial f_i(x))$$

Consider the case when C_i is the farthest set from x. Then $f_i(x) = f(x)$, and

$$g_i = \nabla f_i(x) = \frac{x - P_{C_i}(x)}{||x - P_{C_i}(x)||_2}$$

Hence $g_i \in \partial f(x)$.

Hence, we can apply the subgradient method with Polyak size $t_k = f(x^{(k-1)})$ (as $f^* = 0$ and $||g_i||_2 = 1$).

Update step: At iteration k, with C_i being the farthest set from $x^{(k-1)}$,

$$\begin{aligned} x^{(k)} &= x^{(k-1)} - f(x^{(k-1)}) \frac{x - P_{C_i}(x)}{||x^{(k-1)} - P_{C_i}(x^{(k-1)})||_2} \\ &= P_{C_i}(x^{(k-1)}) \end{aligned}$$

For two sets, this is the famous alternating projections algorithm. We know that this algorithm has convergence rate of $O(1/\epsilon^2)$.

How do we ensure that when the solution is ϵ-optimal, the solution lies in the intersection of all the sets?

One way to solve this is, shrink all your convex sets C_i by ϵ, and then run the alternating projectionss algorithm. This will now ensure that the ϵ-optimal solution lies within each of the original sets.

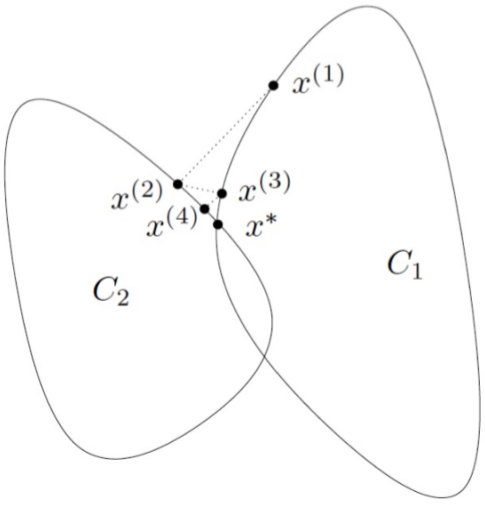

Figure 8.1: alternating projections

8.1.6 Stochastic Subgradient Method

Stochastic methods are useful for optimizing a large sum of functions instead of just a single function. For example, this is the case of the empirical risk minimization.

Consider the following minimization problem:

$$\min_x \sum_{i=1}^{m} f_i(x). \tag{8.1}$$

Stochastic subgradient method repeats the following updates:

$$x^{(k)} = x^{(k-1)} - t_k g_{i_k}^{(k-1)},\ k = 1, 2, 3, \ldots \tag{8.2}$$

where $i_k \in \{1, \ldots, m\}$ is some chosen index at iteration k, chosen by either by the *randomized* or *cyclic* rule, and $g_i^{(k-1)} \in \partial f_i(x^{(k-1)})$. The difference between this update and the subgradient method is simply that we

avoid computing the full sum $\sum_{i=1}^m g_i^{(k-1)}$ at each iteration. Also note that when each f_i, $i = 1, \ldots, m$ is differentiable, this reduces to the stochastic gradient descent (SGD).

As mentioned, there are two rules for choosing index i_k at iteration k:

- *Cyclic*: choose $i_k = 1, 2, \ldots, m, 1, 2, \ldots, m, \ldots$.
- *Randomized*: choose $\in \{1, \ldots, m\}$ uniformly at random.

The randomized rule is more commonly used, as it protects against the worst case or adversarial scenario.

How does the stochastic subgradient method differ from the batch subgradient method? Computationally, we know m stochastic steps approximately correspond to one batch step, but a major advantages is that we do not need to "touch" the entire data when applying a stochastic step.

8.1.7 Convergence of Stochastic Methods

Let f_i with $i = 1, \ldots, m$ be convex and Lipschitz with constant G. Note that $f = \sum_{i=1}^m f_i$ is Lipschitz with mG being the upper bound on its Lipschitz constant.

For fixed step sizes $t_k = t$ for every iteration k, both cyclic and randomized methods satisfy:

$$\lim_{k \to \infty} f(x_{\text{best}}^{(k)}) \leq f^\star + 5m^2G^2t/2. \tag{8.3}$$

The constant 5 in the bound is an artifact of the proof. Since f is mG-Lipschitz, this bound is similar to the bound on the batch subgradient method. For diminishing step sizes (e.g. square summable but not summable), both cyclic and randomized methods converge to the optimum in the limit.

8.1.8 Example: Stochastic Logistic Regression

Consider the following problem:

$$\min_{\beta \in \mathbb{R}^p} f(\beta) = \sum_{i=1}^n -y_i x_i^\top \beta + \log(1 + \exp(x_i^\top \beta)). \tag{8.4}$$

The gradient of the objective is $\nabla f(\beta) = \sum_{i=1}^n (\sigma_i(\beta) - y_i)$, where $\sigma_i(\beta) = \exp(x_i^\top \beta)/(1 + \exp(x_i^\top \beta))$. The gradient is not feasible to compute on every iteration when n is very large. One batch update costs $O(np)$, while one gradient update is only $O(p)$. Convergence of logistic regression using batch and stochastic methods is given on Figure 8.2. Note how the stochastic method moves towards the solution much more quickly than the batch method during early iterations, but then moves much slower as it approaches the solution. *Rule of thumb for stochastic methods:* generally thrive far from optimum, but struggle close to the optimum.

8.1.9 Improving on the Subgradient Method

It turns out not to be possible to improve over the convergence rate of the subgradient method using first-order methods to find the solution to a nonsmooth function where we are only given a weak subgradient oracle. A weak oracle for the subgradient means that at each step we are given a subgradient and do not have control over the choice of the subgradient. The following theorem of Nesterov provides a tight lower bound on the rate for such case.

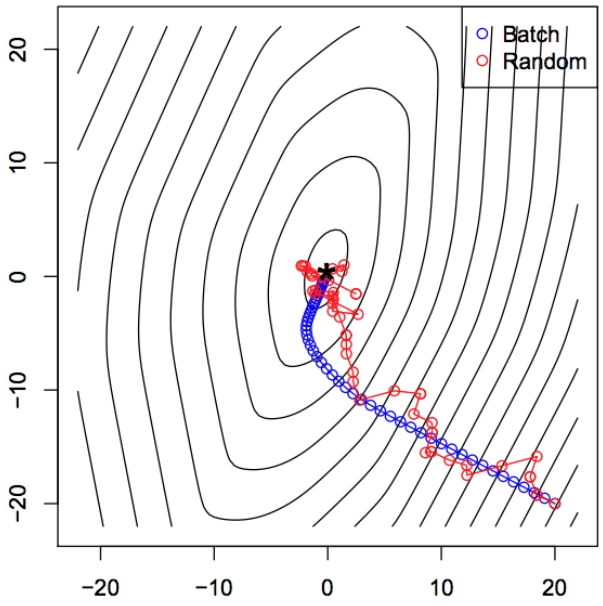

Figure 8.2: Blue: batch steps, $O(np)$. Red: stochastic steps, $O(p)$.

Theorem 8.3 (Nesterov) *For any $k \leq n1$ and starting point $x^{(0)}$, there is a function in the specified problem class such that any nonsmooth first-order method satisfies the following lower bound:*

$$f(x^{(k)}) - f^\star \geq \frac{RG}{2(1+\sqrt{k+1})}. \tag{8.5}$$

Therefore, instead of trying to improve convergence rates for all nonsmooth functions, we further focus on functions of the form $f(x) = g(x) + h(x)$, where g is convex and differentiable, and h is convex, potentially nonsmooth, but "simple" in the sense defined further.

8.2 Proximal Gradient Descent

We can improve on a rather slow subgradient method by turning to proximal gradient descent, an algorithm with an improved running time and the ability to act on a decomposable objective function that may not necessarily be differentiable.

8.2.1 Decomposable Functions

Consider an objective function that is decomposable into two functions as follows:

$$f(x) = g(x) + h(x) \tag{8.6}$$

where g is a convex and differentiable function, and h is convex and possibly non-differentiable. An example for a simple h is the l_1-norm of a vector. With the proximal gradient descent method, we can achieve a convergence rate of $O(1/\epsilon)$. By adding acceleration, this can be improved to $O(1/\sqrt{\epsilon})$.

Simple gradient descent works with a convex and differentiable function f, using gradient information. One can derive gradient descent step using a quadratic approximation of the objective function, $f(x)$, by replacing $\nabla^2 f$ with $\frac{1}{t}I$:

$$x^+ = \arg\min_z f(x) + \nabla f(x)^\top (z - x) + \frac{1}{2t}\|z - x\|_2^2 \tag{8.7}$$

If f is not differentiable, but is decomposable into two functions g and h as described above, we can still use a quadratic approximation of the smooth part g to define a step towards the minimum value:

$$x^+ = \arg\min_z g(x) + \nabla g(x)^\top (z - x) + \frac{1}{2t}\|z - x\|_2^2 + h(z) \tag{8.8}$$

8.2.2 Proximal Mapping

We can re-write the update rule (8.8) in the following form:

$$x^+ = \arg\min_z \frac{1}{2t}\|z - (x - t\nabla g(x))\|_2^2 + h(z) := \text{prox}_{h,t}(x - t\nabla g(x)), \tag{8.9}$$

where we effectively defined the *prox* function. In equation (8.9), the first term is minimized when z is close to the gradient update of the smooth part g, and the second term is minimized when the value of h is as small as possible.

8.2.3 Proximal Gradient Descent

Using the prox function, we can now define an iterative procedure, called proximal gradient descent, as follows. First, choose initial $x^{(0)}$, then repeat:

$$x^{(i)} = \text{prox}_{h,t_i}(x^{(i-1)} - t_i \nabla g(x^{(i-1)})), \;\; i = 1, 2, 3, \ldots \tag{8.10}$$

This can be further re-written in the following more familiar, additive form:

$$x^{(i)} = x^{(i-1)} - t_i G_{t_i}(x^{(i-1)}), \tag{8.11}$$

where $G_{t_i}(x^{(i-1)}) = (1/t)(x^{(i-1)} - \text{prox}_{h,t_i}(x^{(i-1)} - t_i \nabla g(x^{(i-1)})))$.

Even though it may seam that we simply substituted one optimization problem with another one, the approach can be advantageous because:

- The proximal map $\text{prox}_{h,t}(\cdot)$ can be computed *analytically* for many different h functions.
- $\text{prox}_t(\cdot)$ depends only on h, and hence can be used with different g's.
- g can be a arbitrarily complicated function; all we need to do is to compute its gradient.

8.2.4 Example: Iterative Soft-Thresholding Algorithm (ISTA)

Consider the lasso problem:

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1, \tag{8.12}$$

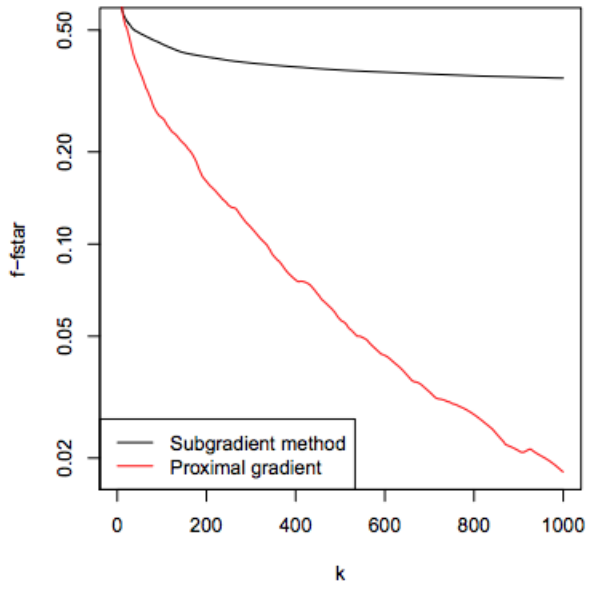

Figure 8.3: Example of proximal gradient descent (ISTA) vs. subgradient method convergence rates.

where we let $g(\beta) = \frac{1}{2}\|y - X\beta\|_2^2$ and $h(\beta) = \|\beta\|_1$. For the proximal gradient descent algorithm to work, we need to find the proximal mapping for the given objective function. We know that the proximal mapping can be computed as follows:

$$\text{prox}_{h,t}(\beta) = \arg\min_z \frac{1}{2t}\|\beta - z\|_2^2 + \|z\|_1 = S_{\lambda t}(\beta), \tag{8.13}$$

where $S_{\lambda t}(\beta)$ can be computed analytically and corresponds to the soft thresholding operator:

$$S_{\lambda t}(\beta) = \begin{cases} \beta_i - \lambda, & \beta_i > \lambda \\ 0, & -\lambda \leq \beta_i \leq \lambda \\ \beta_i + \lambda, & \beta_i < -\lambda \end{cases} \tag{8.14}$$

The gradient of $g(x)$ is $X^\top(X\beta - y)$, and therefore we arrive at the following proximal gradient descent update:

$$\beta^+ = S_{\lambda t}(\beta - tX^\top(X\beta - y)). \tag{8.15}$$

Performance of this algorithm versus the subradient on lasso is given Figure 8.3.

8.2.5 Convergence Analysis

For the objective $f(x) = g(x) + h(x)$, we assume the following:

- The function g is convex, differentiable, $\text{dom}(g) = \mathbb{R}^n$, and ∇g is L-Lipschitz continuous with $L > 0$.
- The function h is convex and its proximal map can be evaluated.

Convergence of the proximal gradient method is given by the following theorem.

Theorem 8.4 *Proximal gradient descent with fixed step size $t \leq 1/L$ satisfies*

$$f(x^{(k)}) - f^\star \leq \frac{1}{2tk}\|x^{(0)} - x^\star\|_2^2. \tag{8.16}$$

Corollary 8.5 *This implies that the proximal gradient descent has a convergence rate of $O(1/k)$ or $O(1/\epsilon)$.*

Notice that the convergence rate is similar to the convergence rate of the gradient descent. However, we should be careful as this specifies the number of *iterations* and the cost of each iteration of the proximal gradient method depends on the cost of evaluating the proximal operator.

8.2.6 Backtracking Line Search

Similar to gradient descent, backtracking line search to determine the step size for each step towards the minima. However, the search is applied only on the smooth part g of the function f.

To perform backtracking line search, first choose a shrinking parameter, $0 < \beta < 1$, and then at each iteration, start with $t = 1$, and while the following condition is true

$$g(x - tG_t(x)) > g(x) - t\nabla g(x)^\top G_t(x) + \frac{t}{2}\|G_t(x)\|_2^2 \tag{8.17}$$

shrink $t = \beta t$, where $G_t(x)$ is the generalized gradient as described in previous sections. Once the while condition is no longer true, perform the proximal gradient update With the same assumptions as those for gradient descent, we get the same convergence rate for proximal gradient descent:

$$f(x^{(k)}) - f^\star \leq \frac{1}{2t_{\min}k}\|x^{(0)} - x^\star\|_2^2, \tag{8.18}$$

where $t_{\min} = \min 1, \beta/L$.