10-725/36-725: Convex Optimization **Spring 2015**

Lecture 7: February 4

Lecturer: Ryan Tibshirani *Scribes: Rulin Chen, Dan Schwartz, Sanxi Yao*

7.1 Subgradient method

Lecture 5 covered gradient descent, an algorithm for minimizing a smooth convex function f. In gradient descent, we assume f has domain $\mathbb{R}^n$, and choose some initial $x^{(0)} \in \mathbb{R}^n$. We then iteratively take small steps in the direction of the negative gradient at the value of $x^{(k)}$ where k is the current iteration. Step sizes are chosen to be fixed and small or by backtracking line search. If the gradient, ∇f, is Lipschitz with constant $L \geq 0$, a fixed step size $t \leq 1/L$ ensures a convergence rate of $O(1/\epsilon)$ iterations to acheive error less than or equal ϵ. Gradient descent requires f to be differentiable at all $x \in \mathbb{R}^n$ and it can be slow to converge. The subgradient method removes the requirement that f be differentiable. In a later lecture, we will discuss speeding up the convergence rate.

Like in gradient descent, in the subgradient method we start at an initial point $x^{(0)} \in \mathbb{R}^n$ and we iteratively update the current solution $x^{(k)}$ by taking small steps. However, in the subgradient method, rather than using the gradient of the function f we are minimizing, we can use **any** subgradient of f at the value of $x^{(k)}$. The rule for picking the next iterate in the subgradient method is given by:

$$x^{(k)} = x^{(k-1)} - t_k \cdot g^{(k-1)}, \text{ where } g^{(k-1)} \in \partial f(x^{(k-1)})$$

Because moving in the direction of some subgradients can make our solution worse for a given iteration, the subgradient method is not necessarily a descent method. Thus, the best solution among all of the iterations is used as the final solution:

$$f(x_{best}^{(k)}) = \min_{i=0,\ldots k} f(x^{(i)})$$

7.1.1 Step size choices

There is no generic idea of backtracking in the subgradient method, so there are two ways to generically choose step sizes. The first is to choose a fixed step size, t. The second is to choose diminishing step sizes, where the step sizes should diminish but not too quickly. There are various ways to do this, but one easy way is to make the step sizes square summable but not summable:

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

An example of a square summable but not summable choice of step size is $t_k = 1/k$. This is a common choice of step size in the subgradient method.

7.1.2 Convergence analysis

Let f be a Lipschitz continuous convex function with domain $\mathbb{R}^n$ and Lipschitz constant $G > 0$:

$$|f(x) - f(y)| \leq G||x - y||_2$$

7.1.2.1 Basic Inequality

From the definition of subgradient:

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

Rewriting in terms of $x^{(0)}$:

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

Using $||x^(k) - x*||_2 \geq 0$, letting $R = ||x^(0) - x^*||_2$:

$$0 \leq R^2 - 2\sum_{i=1}^{k} t_i \left(f(x^{(i-1)}) - f(x^*)\right) + G^2 \sum_{i=1}^{k} t_i^2$$

Introducing $f(x_{best}^{(k)}) = \min_{i=0,\ldots,k} f(x^{(i)})$ and substituting for $f(x^{(i-1)}) - f(x^*)$ makes the right side larger:

$$0 \leq R^2 - 2\sum_{i=1}^{k} t_i \left(f(x_{best}^{(k)}) - f(x^*)\right) + G^2 \sum_{i=1}^{k} t_i^2$$

Rearranging gives what's called the **basic inequality**:

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

7.1.2.2 Convergence theorems

Theorem 7.1 *For fixed step size t:*

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

This follows immediately from substituting t for t_i in the basic inequality.

Note that with fixed step size, the optimal value is not acheived in the limit. Smaller fixed step sizes will be reduce the gap between f^* and $f(x_{best}^{(k)})$.

Theorem 7.2 *For diminishing step sizes:*

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

Consider square summable but not summable step sizes. Then as $k \to \infty$, the denominator on the right hand side of the basic inequality goes to ∞ but the numerator is finite. Thus the right hand side goes to 0.

7.1.3 Convergence rate

Convergence rate for fixed step size: Consider the right hand side of the basic inequality with fixed step size t:

$$\frac{R^2 + G^2 \sum_{i=1}^k t^2}{2 \sum_{i=1}^k t} = \frac{R^2}{2kt} + \frac{G^2 t}{2}$$

To force the right hand side of the basic inequality to be less than ϵ, we can force both of these terms to be less than $\frac{\epsilon}{2}$:

$$\frac{R^2}{2kt} \leq \frac{\epsilon}{2}, \qquad \frac{G^2 t}{2} \leq \frac{\epsilon}{2}$$

Rearranging the second term and maximizing step size t gives $t = \frac{\epsilon}{G^2}$. Rearranging the first term to solve for k, plugging in t and minimizing k gives $k = \frac{R^2 G^2}{\epsilon^2}$. Thus for an ϵ optimal solution, we need $O(\frac{1}{\epsilon^2})$ iterations. Although the derivation shown here does not make this a tight bound, this bound turns out to be tight and this convergence rate is the best you can do for the subgradient method. Compared to gradient descent, with a convergence rate of $O(\frac{1}{\epsilon})$ this is a big difference. For example, if we take $\epsilon = 0.001$, then gradient descent will converge after $O(1000)$ iterations while the subgradient rate is $O(1 \text{ million})$!

Example: Regularized logistic regression
Given $(x_i, y_i) \in \mathbb{R}^p \times \{0, 1\}$for $i = 1, ...n$, consider logistic regression loss:

$$f(\beta) = \sum_{i=1}^n (-y_i x_i^T \beta + \log(1 + \exp(x_i^T \beta)))$$

Note that this is convex because the log-sum of exponentials is convex, and the first term is linear in β.

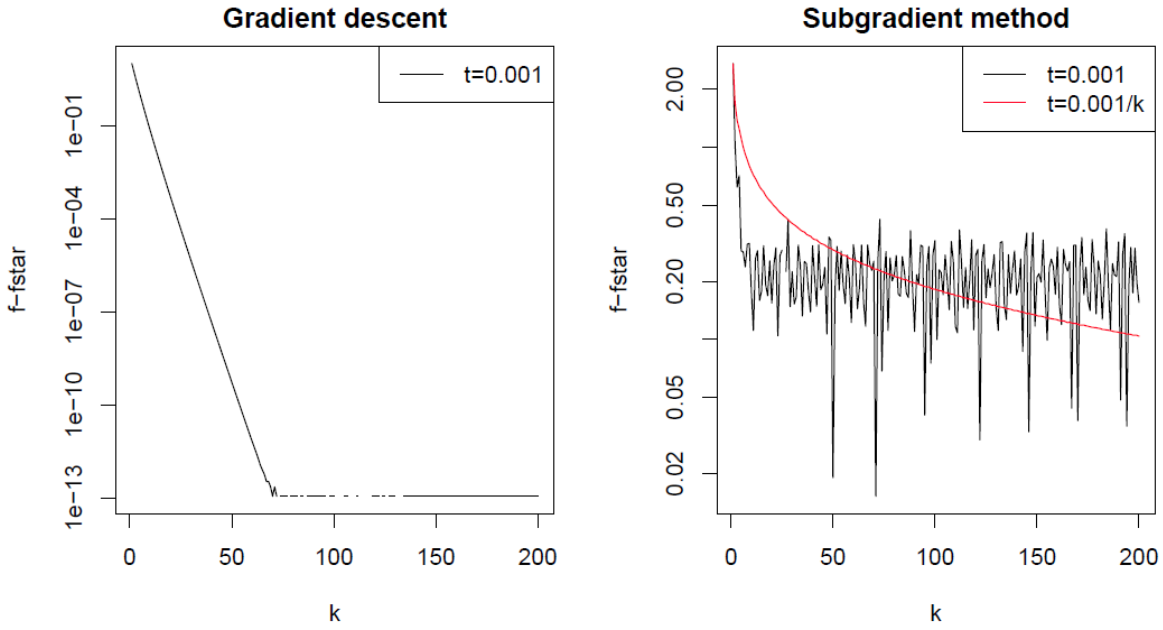
$$\begin{aligned} \nabla f(\beta) &= \sum_{i=1}^n (y_i - p_i(\beta)) x_i, \text{ where} \\ p_i(\beta) &= \exp(x_i^T \beta)/(1 + \exp(x_i^T \beta)), \ i = 1, ..., n \end{aligned}$$

Consider the regularized problem:

$$\min_{\beta \in \mathbb{R}^p} f(\beta) + \lambda P(\beta),$$

where $P(\beta) = ||\beta||_2^2$(ridge penalty) or $P(\beta) = ||\beta||_1$(lasso penalty).

Figure 7.1: Ridge problem: gradient descent; lasso problem: subgradient method. Data example with n = 1000, p = 20



You can see in figure 7.1 how the convergence rate on the ridge penalty version of the problem solved with the gradient descent method compares to the convergence rate of the subgradient method on the lasso penalty version of the problem.

7.2 Polyak Step Size

There is another choice of step size when we know the optimal value f^*. It is called Polyak step size, which minimizes the bound of $||x^{(k)} - x^*||$.

Definition

When the optimal value f^* is known, take the step size

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

This is defined as Polyak step size.

Motivation

In the first step in subgradient proof:

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

To minimize the right hand side, take the derivative with respect to t_k and let it equal 0.

$$-2(f(x^{(k-1)}) - f(x^*)) + 2t_k||g^{(k-1)}||_2^2 = 0$$

We then get the above t_k, which is how the Polyak step size is derived. With Polyak step size, the subgradient method converges to optimal value with convergence rate $O(\frac{1}{\epsilon^2})$.

Example: Intersection of Sets

Suppose we want to find $x^* \in C_1 \bigcap ... \bigcap C_m$, i.e, find a point in intersection of closed, convex sets $C_1, ..C_m$. First define $f_i(x) = \text{dist}(x, C_i), i = 1, ..., m$ and $f(x) = \max\limits_{i=1,...,m} f_i(x)$. Then the problem is to solve:

$$\min f(x)$$

Note that when $f^* = 0$, which means the sets have non-empty intersection, then $f_i(x^*) = \text{dist}(x^*, C_i) = 0$ for all i, which means $x^* \in C_i$ for all i. So:

$$x^* \in C_1 \bigcap ... \bigcap C_m$$

This problem is convex. Reason: for convex set C the distance function $\text{dist}(x, C)$ is convex, so $f_i(x)$ is convex, and the max of a set of convex functions is still convex. So f is convex.

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

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

where $P_C(x)$ is the projection of x onto C. Recall for $f(x) = \max_{i=1,...,m} f_i(x) = \max_{i=1,...,m} \text{dist}(x, C_i)$, the subgradient is

$$\partial f(x) = \text{conv}(\bigcup_{i: f_i(x) = f(x)} \partial f_i(x)) = \text{conv}(\bigcup_{C_i \text{ farthest from x}} \frac{x - P_{C_i}(x)}{||x - P_{C_i}(x)||_2})$$

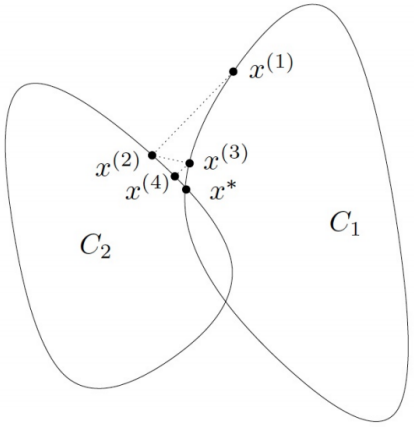
So if $f_i(x) = f(x)$ and $g_i \in \partial f_i(x)$, then $g_i \in \partial f(x)$, here $g_i = \nabla f_i(x) = \frac{x - P_{C_i}(x)}{||x - P_{C_i}(x)||_2}$. Now apply subgradient method with Polyak step size, repeat:

- find i such that C_i is farthest from x
- take $g_i = \frac{x - P_{C_i}(x)}{||x - P_{C_i}(x)||_2}$
- $x^+ = x - tg_i$, here t is Polyak step size

$$t = \frac{f(x) - f^*}{||g_i||_2} = \frac{||x - P_{C_i}(x)||_2 - 0}{1} = ||x - P_{C_i}(x)||_2$$

so $x^+ = x - ||x - P_{C_i}||_2 \cdot \frac{x - P_{C_i}(x)}{||x - P_{C_i}||_2} = P_{C_i}(x)$

Figure 7.2: A picture showing Polyak steps (in this case equivalent to alternating projections) on the problem of finding a point in the intersection of two sets



(From Boyd's lecture notes)

This is the famous alternating projection algorithm, i.e., just keep projecting back and forth.

7.3 Stochastic Subgradient Method

Stochastic methods are useful when optimizing the sum of functions instead of a single function. This happens almost every time we are doing emperical risk minimization over a number of samples.

Definition

Consider minimizing the sum of m functions

$$\min \sum_{i=1}^{m} f_i(x)$$

Two examples for this are linear regression (least squares loss) $\sum_{i=1}^{n} (y_i - x_i^T \beta)$ where $f_i(\beta) = y_i - x_i^T \beta$ and logistic regression (logistic loss) $\sum_{i=1}^{n} (-y_i x_i^T \beta + \log(1 + \exp(-x_i^T \beta)))$ where $f_i(x) = -y_i x_i^T \beta + \log(1 + \exp(-x_i^T \beta))$. Both examples are convex (affine can be regarded as convex).

Sometimes m can be quite large.

Recall that $\partial \sum_{i=1}^{m} f_i(x) = \sum_{i=1}^{m} \partial f_i(x)$, and subgradient method would repeat

$$x^{(k)} = x^{(k-1)} - t_k \cdot \sum_{i=1}^{m} g_i^{(k-1)}, \qquad k = 1, 2, 3, ...$$

where $g_i^{(k-1)} \in \partial f_i(x^{(k-1)})$, while stochastic subgradient method (or incremental subgradient) repeats

$$x^{(k)} = x^{(k-1)} - t_k \cdot g_{ik}^{(k-1)}, \qquad k = 1, 2, 3, ...$$

where $i_k \in \{1, ..., m\}$ is some chosen index at iteration k. In other words, rather than computing the full subgradient, only the subgradient g_i corresponding to one of the functions f_i is used.

As a special case in which $f_i, i = 1, 2, 3, ...$ are differentiable, then $g_i^{(k-1)} = \nabla f_i(x^{(k-1)})$. This is called stochastic gradient descent.

Two methods for choosing which g_i to use

There are two rules for choosing index i_k at iteration k:

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

The randomized rule is more commonly used, because it protects against worst case behavior.

How does the stochastic subgradient method differ from the batch subgradient method? Computationally, we know m stochastic steps $\approx$ one batch step, but a major advantage is that you do not need to have all data points in memory when applying the stochastic method.

Let's consider applying the cyclic rule to smooth functions f_i with a constant step size t as a simple case. Here we cycle through a descent step on each f_i individually. After m steps, namely one cycle, the update is:

$$x^{(k+m)} = x^{(k)} - t \sum_{i=1}^{m} \nabla f_i(x^{(k+i-1)}) \qquad \text{(stochastic)}$$

Instead one batch step is

$$x^{(k+1)} = x^{(k)} - t \sum_{t=1}^{m} \nabla f_i(x^{(k)}) \qquad \text{(batch)}$$

The stochastic update after a cycle can be expressed in terms of a batch step with difference E, where

$$x^{(k+m)}(\text{stochastic}) = x^{(k)} - t \sum_{t=1}^{m} \nabla f_i(x^{(k)}) + t \sum_{i=1}^{m} (\nabla f_i^{(k-i+1)} - \nabla f_i^{(k)}) = x^{(k+1)}(\text{batch}) + E$$

with

$$E = t \sum_{i=1}^{m} (\nabla f_i^{(k-i+1)} - \nabla f_i^{(k)})$$

You can think of E as the error accumulated from evaluating the subgradient at the "wrong" point compared to the batch subgradient method (since the subgradient is calculated for each f_i after the iterate has changed rather than calculating them all before the iterate is changed). We can believe that the stochastic method still converges if $\nabla f(x)$ doesn't vary wildly with x, e.g. if $\nabla f(x)$ is Lipschitz, in which case the difference E almost vanishes.

7.3.1 Convergence of Stochastic Methods

The proofs for these bounds were not shown in class, but can be found in the references.

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

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

$$\lim_{k\to\infty} f(x_{best}^{(k)}) \leq f^* + 5m^2G^2t/2$$

The 5 in the last term is just an artefact of the proof. Since f is mG Lipschitz, this bound is very similar to the earlier bound on the batch subgradient method.

For diminishing step sizes (e.g. square summable but not summable), cyclic and randomized methods satisfy

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

Convergence Rate

Proofs of the convergence rates can also be found in the references. However, the main points in the outcome are described here.

As described above, the batch subgradient method has a convergence rate of $O(\frac{G_{batch}^2}{\epsilon^2})$ (ignoring R^2).

If we use the cyclic rule to choose the next subgradient during the stochastic subgradient method, the *iteration complexity* is $O(m\frac{(mG)^2}{\epsilon^2})$. Since m iterations of the cyclic rule are the same as 1 iteration of the batch rule, the *cyclic complexity* (i.e. the number of cycles needed for error $\leq \epsilon$), is $O(\frac{(mG)^2}{\epsilon^2})$. The bound is really the same in the cyclic rule version of the stochastic subgradient method as in the batch subgradient method since the Lipschitz constant for the sum of functions in the stochastic subgradient method is mG.

If we use the randomized rule to choose the next subgradient during the stochastic subgradient method, the iteration complexity is $O(\frac{(mG)^2}{\epsilon^2})$, and the cyclic complexity is $O(\frac{mG^2}{\epsilon^2})$. The complexity is thus reduced by a factor of m compared to the cyclic rule, but note that the comparison is not really fair since the complexity of the cyclic rule is a bound on worst case behavior while the complexity of the randomized rule is a bound of expected behavior. Nonetheless, in practice people use the randomized rule since it protects against the worst case behavior and m can be a big factor.

Example: Stochastic logistic regression

We want to solve the following problem:

$$\min_{\beta\in\mathbb{R}^p} f(\beta) = \sum_{i=1}^{n} \left(-y_i x_i^T\beta + \log(1+\exp(x_i^T\beta)\right)$$

The gradient is:

$$\nabla f(\beta) = \sum_{i=1}^{n}(y_i - p_i(\beta))$$

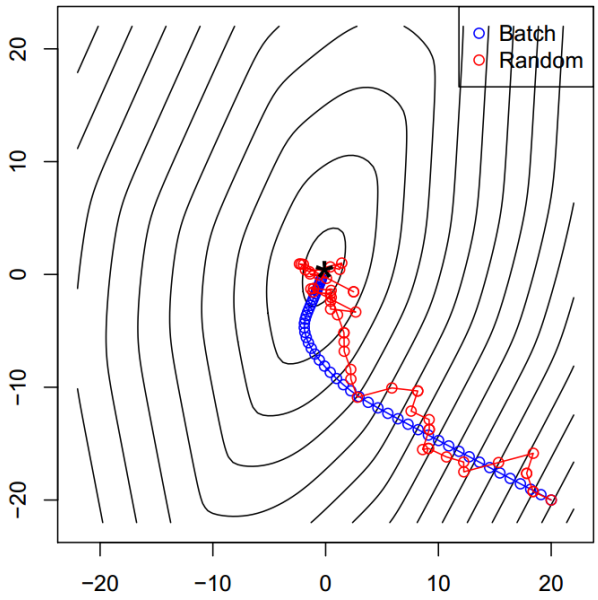
where

$$p_i(\beta) = \exp(x_i^T\beta)/(1+\exp(x_i^T)(\beta))$$

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 $O(p)$.

In general, stochastic methods tend to 'strive' during early iterations, meaning that they progress towards the solution much more quickly than the batch method, but they 'struggle' when they are close to the solution. If you just want an approximate solution, for instance in a noisy situation, then the stocahstic methods can be very advantageous. If you want an exact solution, you may not get it with stochastic methods.

Figure 7.3: Convergence of logistic regression using batch and stochastic methods. The stochastic method is using a random choice in the selection of the data point to use at each step. Note how the stochastic method moves towards the solution much more quickly than the batch method during early iterations, but then moves much more slowly than the batch method as both approach the solution. Also note that each of the stochastic steps is $O(p)$ while each of the batch steps is $O(np)$, so each of the steps in the picture is much more expensive for the batch method than for the stochastic method.



Tight bound on first-order nonsmooth methods

It turns out not to be possible to improve over the convergence rate of the subgradient method for first-order methods to find solutions to nonsmooth functions where you have a weak oracle of the subgradient. A weak oracle for the subgradient means that you know one subgradient, but you do not get to choose which subgradient you know. In this situation, there is a theorem from Nesterov giving a tight bound:

Theorem 7.3 *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*

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

Instead of trying to improve convergence rates for all nonsmooth functions, we will focus on *composite functions* of the form $f(x) = g(x)+h(x)$ where g is convex and differentiable, and h is convex and nonsmooth but "simple", i.e. its prox operator (next lecture) can be computed in closed form. For many functions, a $O(1/\epsilon)$ convergence rate can be acheived with algorithms on composite functions.

References and Further Reading

- D. Bertsekas (2010), Incremental gradient, subgradient, and proximal methods for convex optimization: a survey

- S. Boyd, Lecture notes for EE 264B, Stanford University, Spring 2010-2011
- Y. Nesterov (2004), Introductory lectures on convex optimization: a basic course, Chapter 3
- B. Polyak (1987), Introduction to optimization, Chapter 5
- L. Vandenberghe, Lecture notes for EE 236C, UCLA, Spring 2011-2012