10-801 Spring 2014

Advanced Optimization and Randomized Methods

Complexity Theory for Optimization

Lecturer and scribe: Aaditya Ramdas

1 Why Study Lower Bounds?

The broad topic of discussion will be complexity theory for optimization, which essentially means "the quest for finding matching lower and upper bounds".

Why is this important? When designing optimization algorithms for your own application, you often end up analyzing their "convergence rate". Say for some problem you designed an algorithm and you get a function-error convergence rate of O(Ld/T). Is this the best dependence on dimension possible and Lipschitz constant? Is this the best dependence on time-steps possible? Such questions will always arise - can I do better, or is this somehow optimal?

That's where minimax lower bounds come in. Their basic aim is usually to answer such questions precisely. They will depend on many things - the geometry of the set you are optimizing over (cube? sphere? simplex?), the kind of information you have access to (function values? gradients?), the kind of noise model for that information (noiseless/deterministic? stochastic?), the function class you are optimizing over (convex? strongly convex?), etc.

It may not be possible to go into proof details, but it is certainly possible to understand many of the main issues to solve, intelligent ideas, subtle construction arguments, connections to hypothesis testing, etc and try to appreciate their usefulness and their vast limitations, and see the road forward (and how it can help you in your research!). Out of personal choice, my emphasis will not be on proofs (since the probability of you needing to prove a lower bound is low), but on where to find these bounds, how to understand what they're saying, how to understand what they're *not* saying, and how to use that to speed up your life.

2 Oracles

One can think of complexity theory as representing a game, between you and an oracle. You get to ask the oracle certain types of questions (the nature of the oracle will determine the kind of answers you get) and this is the only kind of information you have access to. Our aim is to understand what's the minimum number of questions you can ask the oracle, before you can optimize an unknown function f. Examples of different oracles include:

1. First order deterministic oracle - each query must be a point in the space, say x, the answer you get is $f(x), g(x)$ where $g(x) \in \partial f(x)$.

2. First order stochastic oracle - each query must be a point in the space, say x, the answer you get is $\hat{f}(x), \hat{g}(x)$ where $\mathbb{E}[\hat{f}(x)] = f(x), \mathbb{E}[\hat{g}(x)] \in \partial f(x)$. (technical point: noise must have finite variance - oracle can add gaussian noise, not cauchy noise).

3. Zeroth order deterministic/stochastic oracle - similar to first order deterministic/stochastic oracles, without gradient information.

4. Zeroth order noisy comparision oracle - your query must be two points x, y, and the answer you get is $\text{sign}(f(x) - f(y))$ with a probability that increases with how large $f(x) - f(y)$ is.

3 Geometry (Error, Function, Set)

Iterative numerical algorithms can (almost) never optimize a function exactly except for in special cases. Given, that we can only approximately minimize functions, we need to have some notion of approximation error, or some measure of how well we're doing. How does one measure accuracy? There are two main choices:

1. Function error - $\epsilon_T := f(x_T) - f(x^*)$

2. Point error - $\rho_T := \|x_T - x^*\|$

I purposely left the norm out of the second equation, since you can indeed choose to measure your error in different norms. Function error is much more commonly studied than point error, but there are sometimes ways to go between the two. There are two ways in which one can state convergence rates, which are nearly equivalent:

1. Fix the time budget T (number of oracle queries, aka iterations) and calculate the best accuracy you can reach, like $\epsilon_T = O(d/T)$.

2. Fix an accuracy ϵ and calculate the number of time steps you need to reach that accuracy, like $T_\epsilon = O(d/\epsilon)$.

How quickly you can optimize a function hugely depends on the geometry of the set you are optimizing over. For example, the l_2 ball, the l_1 ball and l_∞ ball have different *sizes* - their diameters differ by $\sqrt{d}$ or d factors depending on which norm you measure the diameter in.

Of course, different functions can be optimized at different rates. The way one studies bounds is to define a *class* of functions, and ask what the convergence rate is if the oracle chose some unknown function from that class. For example, some common classes are Lipschitz non-smooth convex functions, Lipschitz non-smooth strongly convex functions, etc. Importantly, Lipschitzness and strong-convexity heavily depend on the choice of norm in which they are Lipschitz/strongly-convex.

All these details about geometry matter only to determine the exact dependence on dimension, Lipschitz constant and strong-convexity constant (measures of conditioning). Sometimes, you'll just hear people saying *subgradient methods have a* $1/\sqrt{T}$ *convergence.*

4 Eg: Smooth or nonsmooth f, FO Stochastic

This example is adapted from *Information-theoretic lower bounds on the oracle complexity of convex optimizatinon* by Agarwal-Bartlett-Ravikumar-Wainwright, IEEE Transactions on Information Theory 2011 (generalization of their easier to read NIPS paper).

Let S be a convex set in $\mathbb{R}^d$ that contains an l_∞ ball of radius r_∞. Let $\mathcal{S}_r^\infty$ be the class of all such sets.

Let f be a convex function that is L-Lipschitz with respect to the $\|.\|_\infty$ on this set S, i.e. $|f(x) - f(y)| \leq L_\infty \|x - y\|_\infty$ for all $x, y \in S$. Let $x_f^* \in \arg\min_{x \in S} f(x)$. Let $\mathcal{F}_L^\infty(S)$ be the class of all such functions.

Let O be a function from $\mathbb{R}^d$ to $\mathbb{R}^2$ such that when queried at $x \in S$, it returns $(\hat{f}(x), \hat{g}(x))$ such that $\mathbb{E}[\hat{f}(x)] = f(x), \mathbb{E}[\hat{g}(x)] \in \partial f(x)$ (and these estimates have bounded variance). If $x \notin S$ it can return garbage. Let $\mathcal{O}$ be the set of all such oracles.

Let A be a *first-order algorithm* (generates sequences in $x_0 + \sum_i \text{span}(\hat{g}(x_i))$ that (knowing S) queries the oracle T times, and returns a guess for optimum as x_T. Let $\mathcal{A}_T(S)$ be the set of all such algorithms, possibly randomized.

Then, for some universal constant c,

$$\sup_{O \in \mathcal{O}} \sup_{S \in \mathcal{S}_r^\infty} \inf_{A \in \mathcal{A}_T(S)} \sup_{f \in \mathcal{F}_L^\infty(S)} \mathbb{E}[f(x_T)] - f(x_f^*) \geq c L_\infty r_\infty \sqrt{\frac{d}{T}}$$

where the expectation is over the oracle noise or any internal randomness of the algorithm.

5 Exercising Caution

Intuitively, it should get worse with a larger sized set, a larger Lipschitz constant, larger dimension (or at least not improve with dimension, it could be independent of dimension), and it should get better with T. An alternative way to read it in English is

There exists a stochastic first order oracle, **there exists** a bounded convex set, such that **for all** algorithms that query the oracle at points in this set, **there exists** some Lipschitz convex function on which they will fail to get closer than "insert rate here" to the optimum.

This means that you will never be able to prove an upper bound of the form

There exists an algorithm (your favourite new one that you invented), such that **for all** Lipschitz convex functions, I will get closer than "insert faster rate here" to the optimum (with only access to stochastic gradients on any bounded convex set).

HOWEVER, in many situations, you may have much more control over the problem. The convex function may have structure that you can exploit (like decomposability), the set may have some particular form (probability simplex), or the stochasticity is very specific (subsampling points). These could help you get faster rates!

6 Eg: Smooth vs Nonsmooth f, FO Deterministic

The following bounds are from Nesterov's Introductory Lectures on Convex Optimization. In short - smoothness buys you a lot, non-stochasticity buys you a lot.

Nonsmooth Let A be any first order method starting from $x_0 \in \mathbb{R}^d$ that has access to a first order non-stochastic oracle. Assume that the solution x^* to the minimization problem $\min_x f(x)$ exists and $\|x_0 - x^*\|_2 \leq D_2$ and that f is L_2-Lipschitz on $\{x : \|x_0 - x\|_2 \leq D_2\}$ - let the class of such functions be $P(x_0, D_2, L_2)$. Then for any class $P(x_0, D_2, L_2)$, and any $0 \leq t \leq d-1$, there exists a function $f \in P(x_0, D_2, L_2)$ such that

$$f(x_t) - f(x^*) \geq \frac{L_2 D_2}{1 + \sqrt{t+1}}$$

(for any A that generates a sequence $\{x_t\}$ satisfying $x_{t+1} = x_0 + \text{span}(g(x_0), ..., g(x_t))$).

Smooth Similar setup. For any $0 \leq t \leq (d-1)/2$ and any x_0, there exists a function f in the class of functions which is infinitely differentiable with an L-Lipschitz gradient ($\|\nabla f(y) - \nabla f(x)\|_2 \leq L\|y - x\|$) such that any first order method satisfies

$$f(x_t) - f(x^*) \geq \frac{3L\|x_0 - x^*\|^2}{32(t+1)^2} \tag{1}$$

$$\|x_t - x^*\|^2 \geq \frac{1}{8}\|x_0 - x^*\|^2 \tag{2}$$

7 Example Upper Bounds

Stochastic subgradient descent (SGD) has the upper bound

$$\mathbb{E}[f(x_T)] - f(x^*) \leq \frac{L_2 D_2}{\sqrt{T}}$$

(the subscript represents the norm in which we measure the diameter or Lipschitz constant of the set - note that r_∞ was not the diameter). This may or may not be optimal in different settings. For example, exponentiated gradient descent achieves

$$\mathbb{E}[f(x_T)] - f(x^*) \leq \frac{L_1 D_1 \sqrt{\log d}}{\sqrt{T}}$$

If we were minimizing over the simplex, with constant diameter in $\|.\|_1$ or $\|.\|_2$, and if the function was such that each coordinate of its gradient was bounded by a constant say 1 (equivalently it is 1-Lipschitz w.r.t. $\|.\|_1$). Since $L_2 \leq \sqrt{d} L_1$ and $D_2 \leq D_1$, this could lead to an exponentially better rate in dimension ($\sqrt{\log d}$ vs $\sqrt{d}$) over SGD.

8 Enter Conditioning, Acceleration

An interesting dependence that arises for strongly convex functions is the dependence on *condition number*, which is the ratio of largest and smallest eigenvalues of the positive definite hessian of f.

Interestingly, (I've heard that) in the 1980s, it was known that if the function is smooth and strongly convex, vanilla gradient descent can achieve a function error of ϵ in $\frac{L}{\lambda}\log(1/\epsilon)$ steps. However, the best lower bound that could be proven was only $\sqrt{\frac{L}{\lambda}}\log(1/\epsilon)$. Similarly, for smooth functions vanilla gradient descent achieves $1/T$ rate but the best lower bound was $1/T^2$ (both those lower bounds were possibly his, or in a textbook by Nemirovski-Yudin).

This motivated Nesterov to find a better method, and he then came up with the amazing accelerated gradient descent algorithm which matches the lower bound in both settings. Bottom line: for smooth functions, I see no reason to ever use GD when one can use accelerated GD - it is a pity that this is neither taught enough nor emphasized enough. Amazingly, acceleration has carried forward to a hoard of other settings that we have seen in this course and last sem's optimization course, including coordinate descent, proximal gradient methods, etc (for vectors or matrices).

9 Eg: SC f, smooth-deterministic and general-stochastic

(Nesterov's book) Let f be μ strongly convex and have L-Lipschitz gradient, and be infinitely differentiable (given μ, L, let F be class of all such functions), then for any first order method A, we have

$$\|x_t - x^*\|^2 \geq \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{2t} \|x_0 - x^*\|^2 \tag{3}$$

$$f(x_t) - f(x^*) \geq \frac{\mu}{2}\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{2t} \|x_0 - x^*\|^2 \tag{4}$$

where $\kappa = L/\mu$ is the condition number.

(ABRW'11) However, for stochastic oracles, if f is L-Lipschitz and strongly convex, the minimax lower bound is

$$\mathbb{E}[f(x_T) - f(x^*)] \geq c\frac{\kappa^2}{T}$$

10 Important "Exceptions" to Lower Bounds

As mentioned earlier, the lower bounds are worst case and *black-box* meaning you know nothing about the function and can only access it through oracle queries. This study has led to huge advances in theory and practice, and in our understanding of the fundamental difficulty of convex optimization. However, in many circumstances we have more structure that we can take advantage of - here are two examples.

Accelerated Proximal Gradient Descent As a first example, consider the lasso loss function $\min_\beta \|y - X\beta\|_2^2 + \lambda\|\beta\|_1$. In the black box setting this is just a non-smooth function and subgradient methods will get you $1/\sqrt{T}$ convergence ($1/T$ if the first term is strongly convex). However, accelerated proximal gradient descent can get you $1/T^2$ (exponentially small if the first term is strongly convex). This argument applies to almost any smooth + nonsmooth function where we can calculate a prox for the nonsmooth term. You cannot do better than this of course, since if $\lambda = 0$, we cannot beat the lower bounds for smooth convex or strongly convex optimization (of $1/T$ or e^{-T} without conditioning terms).

Smooth Empirical Risk Minimization $\min_w \frac{1}{n}\sum_{i=1}^n l_i(w)$ where $l_i(w)$ is a smooth loss function that represents the regression or classification loss of the weight vector w on the i-th example (x_i, y_i) - logistic/squared loss for example (not the nonsmooth hinge). If there is regularization, this can also be included in the sum: $\frac{1}{n}\sum_{i=1}^n l_i(w) + \lambda\|w\|^2 = \frac{1}{n}\sum_{i=1}^n (l_i(w) + \lambda\|w\|^2) = \frac{1}{n}\sum_{i=1}^n l_i'(w)$. A common way to minimize such losses is SGD, where instead of calculating the entire gradient $\frac{1}{n}\sum_{i=1}^n \nabla l_i(w_t)$ at each step, we pick a random data point and calculate just $\nabla l_i(w_t)$. Note that this subsampled gradient is unbiased, i.e. its expected value is the full gradient. Hence, SGD will get you a $1/\sqrt{T}$ rate for such functions. If they're strongly convex, say due to regularization, then SGD can get you a $1/T$ rate. However, we can actually exploit the fact that this is not just *any* unbiased gradient - the function has a particular form as an average of losses, and the unbiased gradient specifically picks one out of these n possible losses and returns you the exact gradient on that point. A method called SAG (Minimizing Finite Sums with the Stochastic Average Gradient - Schmidt-LeRoux-Bach 2012) indeed achieves a $1/T$ rate for convex objectives and exponentially fast (also called *linear* rate) for strongly convex objectives.

Future Directions Lower bounds and upper bounds for structured objectives. Recent algorithms include Pegasos (for hinge loss), Stochastic Dual Coordinate Ascent (SDCA), Stochastic Variance Reduced Gradient (SVRG), etc. For lower bounds, we don't know what's the best we can do in the above setting $\min_w \frac{1}{n}\sum_{i=1}^n l_i(w)$ where $l_i(w)$ is smooth. For example, can we accelerate these methods to get $1/T^2$ or make the exponentially fast rate better conditioned? Will these methods work with non-smooth functions if the nonsmoothness is in a term that has a prox function? (a conversation with Mark Schmidt suggests Proximal-SAG works well in practice)