IFT 6085 - Lecture 4
Black-box Models and Lower bounds

Please report any bugs to the scribes or instructor.

Scribes:
Winter 2018: Sai Krishna, Krishna Murthy
Winter 2019 Ian Porada, Théo Moins, William Starnaud

Instructor: Ioannis Mitliagkas

1 Summary

In the previous lectures, we derived upper bounds on the convergence rate of *gradient descent* for minimizing different types of objective functions. Specifically, we found that depending on the properties we have for the objective function (Lipschitz-continuity, convexity, strong convexity, and smoothness), we can obtain different upper bounds on the convergence rate of gradient descent minimizing this objective function. We also saw that these different bounds are obtained with different algorithms: e.g. considering the last term of the sequence we construct, or the mean of all the points.

In this lecture, we will summarize these previously derived upper bounds. We will then derive a new lower bound for the convergence rate of a black box model minimizing a smooth, strongly convex objective function. Our definition of black box model encompasses a class of algorithms which includes *gradient descent*.

An upper bound on convergence rate is useful to see the efficiency of an algorithm, which is how many steps are needed to obtain a result with a given precision. In contrast, a lower bound allows us to consider the best possible upper bound we can possibly obtain for a class of algorithms. Thus, we are able to restrict ourselves in future research for a better rate of convergence: ideally models with be close this lower bound, but it would be useless to search for better.

2 Convergence rate

In the table below, $D_1 = \|x_1 - x^*\|_2$. It denotes the initial suboptimality, i.e., the L^2 distance of the initial guess x_1 from the optimal point x^*.

Properties of the Objective Function	**Upper bound on Convergence Rate of Gradient Descent**
convex and L-Lipschitz	$\frac{D_1 L}{\sqrt{T}}$
convex and β-smooth	$\frac{\beta D_1^2}{T}$
α-strongly convex and L-Lipschitz	$\frac{L^2}{\alpha T}$
α-strongly convex and β-smooth	$\beta D_1^2 \exp(-\frac{4T}{\kappa})$

We can see that assuming smoothness leads to a faster convergence rate as compared to the case when the objective is assumed to only be L-Lipschitz. When we assume additional properties, for instance strong convexity and smoothness, we get exponential convergence, which is our best upper bound so far. Some authors also refer to this as *linear*

convergence, due to the fact that a semi-logarithmic plot of the convergence rate across gradient descent iterations is reminiscent of a straight line.

Another *seemingly weird* observation from the above table is that, when considering a convex objective that is strongly convex and L-Lipschitz, but not necessarily smooth, the upper bound on the rate of convergence is independent of the initial guess. To understand this, we could first look at what the two assumptions mean. An α-strongly convex function is always lower bounded by a quadratic of curvature α. In addition to this, when we assume that the function is L-Lipschitz, the gradients cannot exceed L. Hence, we're interested only in functions whose curvature is upper bounded by $\frac{L}{\alpha}$. This usually happens when we restrict ourselves to a particular subset of the domain of the convex objective where these properties hold. And when they do hold, we find that the initialization does not play a role in determining the convergence rate. For more details, see theorem 3.9 from [1].

3 Black box model and taxonomy of different methods

A black box model assumes that the algorithm does not know the objective function f being minimized. Information about the objective function can only be accessed by querying an *oracle*. The oracle serves as a bridge between the unknown objective function and the optimizer. At any given step, the optimizer queries the oracle with a *guess* x, and the oracle responds with information about the function around x (eg. value, gradient, Hessian, etc.). This model is suitable when we want to lower bound the asymptotic complexity of minimizing f *regardless of what algorithm we use*. The complexity can be estimated as the number of queries that can be made to the oracle until we find an ϵ-approximate minimum for the convex function f.

Here we present a brief taxonomy on optimization methods, based on the nature of information about the function that the methods require.

3.1 Zeroth order methods

These methods only require the value of function f at the current guess x. They do not access any other *higher-level* information like gradients.

For example: the bisection method, genetic algorithms, simulated annealing and Metropolis method are a few techniques that *can* fall under this category.

3.2 First order methods

These methods can inquire the value of the function f and its first derivative (gradient or Jacobian) of the function ∇f at the current guess x. These methods are widely used for optimization in machine learning problems.

Some of the methods include gradient descent (with or without averaging), Nesterov's accelerated gradient methods [4], and Polyak's momentum[5].

3.3 Second order methods

These methods require the value of the function f, its first derivative (gradient or Jacobian) ∇f, and its second derivative (Hessian) $\nabla^2 f$ at the current guess x. Since these methods use information about the local curvature (which is encoded in the Hessian), they converge in a smaller number of iterations. However, each iteration is computationally intensive, as it typically involves an inversion of the Hessian. Another characteristic of these methods is the *self-tuning* property. The step size (learning rate) is determined implicitly from the curvature information and need not be tuned as a hyperparameter.

Newton's method is a very popular example of a second order method [1]. Improving the efficiency of these algorithms is an active area of research.

[1] Another class of techniques, sometimes referred to as *Quasi-Newton methods* is frequently used by the machine learning community. These techniques attempt to infer (approximate) second order information by using only first order information. BFGS and L-BFGS are popular Quasi-

3.4 Adaptive methods and conjugate gradients

The methods we mentioned until this point assume that all dimensions of a vector-valued variable (or sometimes all variables in the objective function) have a common set of hyperparameters. *Adaptive methods* relax this assumption and allow for every variable (or sometimes every dimension of a vector) to have its own set of hyperparameters (learning rate, momentum, etc). Some popular methods under this paradigm that are used in training deep neural networks are AdaGrad[2] and ADAM[3].

Conjugate gradient descent is a technique that we will not deal with in this course. But, to summarize what it does, it attains the minimum for a quadratic (exact minimum, no suboptimality) in exactly d iterations, where d is the dimensionality of the quadratic function. Exploiting conjugate gradients in deep network training is still an active area of research.

4 Lower bounds

Up until now, we have looked only at upper bounds on convergence rates for convex objectives that are optimized using gradient descent solvers. We will now derive a lower bound for the special case when our objective function is smooth and strongly-convex.

4.1 Why are lower bounds useful?

Lower bounds are useful because they tell us what's the best possible rate of convergence we can have given a category of optimizer. Without lower bounds, an unnecessary amount of research energy would be spent in designing better optimizers, even if convergence rate improvement is impossible within this category of algorithm. Caveats here are that even if we prove that each procedure has a lower bounded rate of convergence, we don't know if a specific method reaches this bound.

Figure 1 is a visualization of the bounds on convergence rates described thus far. As implied by comparing the slopes of these lines, our upper bounds on gradient descent describe a slower rate of convergence than the lower bounds we will derive. Upper bounds on accelerated methods' convergence rates match the derived lower bounds in all but a constant. Any convergence below the lower bound is impossible for the black box models and objective function we will describe.

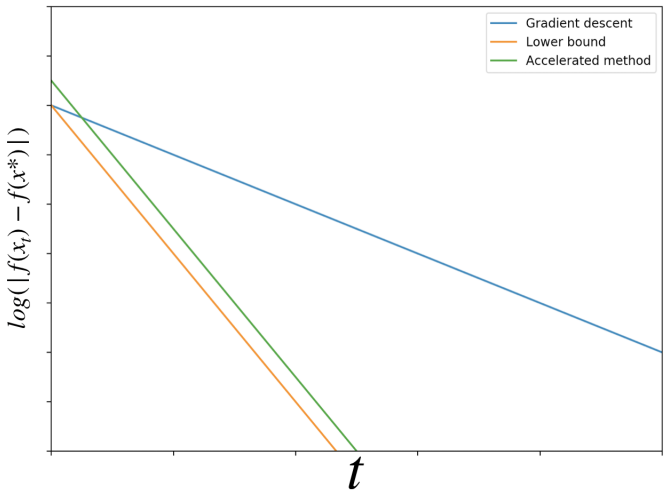

Figure 1: Converence rate for gradient descent and accelerated methods in comparison with lower bound

Newton methods

4.2 Assumptions

Under the black box model for first-order methods, we consider a sequence of iterates $x_1, x_2, x_3, ...$ and a sequence of gradients $g_1, g_2, g_3, ...$ The optimizer updates the variable x (interchangeably referred to as the parameter vector) using an update rule that satisfies the following criterion:

$$x_{t+1} \in span\left(g_1, g_2, ..., g_t\right)$$

This means that x_{t+1} must be a linear combination of the sequence of gradients computed thus far in the algorithm. Here, t is an index into the number of steps the optimizer is run for. Further, in our analysis, we assume that the initial guess x_1 is always the zero vector ($x_1 = \mathbf{0}$), without loss of generality. The above model can easily be tweaked to hold for arbitrary initializations, but in that case, expressions for bounds become tedious. We only make this assumption for algebraic simplicity, because we are interested in bounds on asymptotic rates [2].

4.3 Lower bounds for smooth and strongly convex objectives

Now, we proceed to derive a lower bound for an objective function which is α-strongly convex and β-smooth. Before we proceed, we outline some notation that will be used in the proof.

Theorem 1. *(Theorem 3.15 from [1]) ($\kappa > 1$) There exists a β-smooth and α-strongly convex function $f : l_2 \mapsto \mathbb{R}$ with condition number $\kappa = \frac{\beta}{\alpha}$ such that for any $t \geq 1$ and any* black box *procedure (see Section 3), the following lower bound holds.*

$$f(x_t) - f(x^*) \geq \frac{\alpha}{2}\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{2(t-1)} ||x_1 - x^*||^2 \tag{1}$$

for large values of κ,

$$\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{2(t-1)} \approx \exp\left(-\frac{4(t-1)}{\sqrt{\kappa}}\right) \tag{2}$$

Proof. As is typical of lower bound proofs, we prove this theorem by constructing an example. The example function we construct is an ℓ_2 function. Informally speaking, ℓ_2 functions are vectors with infinitely many coordinates that are also square summable. Formally,

$$\ell_2 = \{x = (x(n)), n \in \mathbb{N}, \sum_{i=1}^{\infty} x(i)^2 < +\infty\}$$

We define an operator that assumes the form of a tridiagonal matrix. Let

$$A = \begin{bmatrix} 2 & -1 & 0 & \dots & \dots & \dots & \dots & \dots & \dots \\ -1 & 2 & -1 & 0 & \dots & \dots & \dots & \dots & \dots \\ 0 & -1 & 2 & -1 & 0 & \dots & \dots & \dots & \dots \\ \vdots & 0 & -1 & 2 & -1 & 0 & \dots & \dots & \dots \\ \vdots & \vdots & 0 & -1 & 2 & -1 & 0 & \dots & \dots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \end{bmatrix}$$

This operator has all its eigenvalues in the range [0,4].
This can be found by the use of Gershgorin disks. First, we let $R_i = \sum_{j \neq i} |a_{ij}|$. Then we define the Gershgorin discs $D(a_{ii}, R_i)$ of A as $D(a_{ii}, R_i) = \{z \in \mathbb{C}, |a_{ii} - z| \leq R_i\}$. The Gershgorin circle theorem [6] then states that every eigenvalue of A must lie within at least one the discs $D(a_{ii}, R_i)$.

[2] If the initial guess was not the zero vector, but x_{init}, the update rule criterion must be modified to $x_{t+1} \in span\left(x_{init}, g_1, g_2, ..., g_t\right)$

So with our expression of A, we only have two different discs, namely $D(2,1)$ and $D(2,2)$. The firs one is included in the latter one (i.e. $D(2,1) \subset D(2,2)$). Moreover, A is symmetrical, so the eigenvalues of A are real. We can infer that the eigenvalues of A must be in $[0,4]$.

Thus, we can define the following quadratic function:

$$f(x) = \frac{\alpha(\kappa-1)}{8}\left(\langle Ax, x\rangle - 2\langle e_1, x\rangle\right) + \frac{\alpha}{2}\|x\|^2$$

Here, $\langle ., . \rangle$ denotes the vector inner-product (also called the dot product) and e_1 denotes the first vector of the canonical basis, i.e.,

$$e_1 = [1, 0, 0, \dots]^T.$$

Since A is symmetric by definition,

$$\langle Ax, x\rangle = x^T A^T x = x^T A x$$

Also note that f is α-strongly convex (the $\frac{\alpha}{2}\|x\|^2$ term ensures that) and β-smooth. β-smoothness arises from the property that the eigenvalues of A are bounded to be in the range $[0,4]$. We now compute the gradient of f.

$$\nabla f(x) = \frac{\alpha(\kappa-1)}{4}\left(Ax - e_1\right) + \alpha x$$

Recall that, under the black box model we assumed that the starting point for our gradient descent routine will be $x_1 = 0$. Plugging that into the expression above, we get

$$\nabla f(x)_{x=x_1} = -\frac{\alpha(\kappa-1)}{4}e_1 = \begin{bmatrix} \frac{-\alpha(\kappa-1)}{4} \\ 0 \\ 0 \\ \vdots \end{bmatrix}$$

Using this expression, it is easy to show—by mathematical induction—that if x_{t-1} has non-zero entries up to element at index $t-1$, then x_t will have non-zero entries up to index t. The way the Hessian A is designed, the non-zero values propagate linearly across the dimensions, one dimension per each step of the gradient descent routine.
That is, $x_t(i) = 0 \; \forall i \geq t$. Let's now consider the norm

$$\begin{aligned}
\|x_t - x^*\|^2 &= \sum_{i=1}^{\infty}\left(x_t(i) - x^*(i)\right)^2 \\
&= \sum_{i=1}^{t-1}\left(x_t(i) - x^*(i)\right)^2 + \sum_{i=t}^{\infty}\left(x_t(i) - x^*(i)\right)^2 && \text{(separating the non-zero entries)} \\
&= C + \sum_{i=t}^{\infty}\left(x_t(i) - x^*(i)\right)^2 && \text{(with } C > 0\text{)} \\
&\geq \sum_{i=t}^{\infty}\left(x_t(i) - x^*(i)\right)^2 && \text{(the first term above was positive)} \\
&= \sum_{i=t}^{\infty}\left(x^*(i)\right)^2 && \text{(all terms are zeros, beginning from term } t\text{)}
\end{aligned} \tag{3}$$

Of course, as we keep running gradient descent, $\|x_t\|$ keeps getting smaller and smaller (if the learning rate is appropriately specified). Strong convexity gives us

$$\begin{aligned}
f(x_t) - f(x^*) &\geq \frac{\alpha}{2}\|x_t - x^*\|^2 \\
&\geq \frac{\alpha}{2}\sum_{i=t}^{\infty}\left(x^*(i)\right)^2
\end{aligned} \tag{4}$$

If we differentiate f and set ∇f to 0, we obtain an infinite linear system, of the following form (using the definition of A):

$$\begin{cases} 1 - 2\frac{\kappa+1}{\kappa-1}x^*(1) + x^*(2) = 0, \\ x^*(k-1) - 2\frac{\kappa+1}{\kappa-1}x^*(k) + x^*(k+1) = 0 \; \forall k \geq 2. \end{cases} \tag{5}$$

The solution of the above system is given by

$$x^*(i) = \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^i$$

Hint for derivation: We sum all rows in the linear system to obtain $1 - x^*(1) + \sum_{k=1}^{\infty}[2 - 2(\frac{\kappa+1}{\kappa-1})]x^*(k) = 0$. We put $x^*(k) = r^k \; \forall k \geq 1$ and we use the value of geometric series to solve for r.

Now, we plug this into the above expression, which gives us

$$\begin{aligned} f(x_t) - f(x^*) &\geq \frac{\alpha}{2}\|x_t - x^*\|^2 \\ &\geq \frac{\alpha}{2}\sum_{i=t}^{\infty}(x_t^*(i))^2 \\ &= \frac{\alpha}{2}\sum_{i=t}^{\infty}\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{2i} && \text{(By solution to the linear system (5) above)} \\ &= \frac{\alpha}{2}\sum_{i=1}^{\infty}\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{2(i+t-1)} && \text{(By change of variable)} \\ &= \frac{\alpha}{2}\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{2(t-1)}\sum_{i=1}^{\infty}\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{2i} \\ &= \frac{\alpha}{2}\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{2(t-1)}\sum_{i=1}^{\infty}(x^*(i))^2 && \text{(By solution to the linear system (5) above)} \\ &= \frac{\alpha}{2}\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{2(t-1)}||x_1 - x^*||^2 && \text{(By derivation (3) above for } x_1\text{)} \end{aligned} \tag{6}$$

This proves the theorem.

□

References

[1] S. Bubeck. *Convex Optimization: Algorithms and Complexity*. Foundations and Trends in Machine Learning, 2014.

[2] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[3] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

[4] Y. Nesterov. A method of solving a convex programming problem with convergence rate o(1/k2). *Soviet Mathematics*, 27 no 2:372–376, 1983.

[5] B. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4 issue 5:1–17, 1964.

[6] S.Gerschgorin. Uber die abgrenzung der eigenwerte einer matrix. *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na*, pages 749–754, 1931.