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

This version of the notes has not yet been thoroughly checked. Please report any bugs to the scribes or instructor.

**Scribe(s):** Sai Krishna, Krishna Murthy **Instructor:** Ioannis Mitliagkas

## 1 Summary

In the previous lecture(s) we looked at some properties of convex functions, namely strong convexity, $\beta$-smoothness. We also derived upper bounds for rates of convergence when *gradient descent* is used in the *minimization* of $\lambda$-strongly convex and $\beta$-smooth functions. Note that these bounds do not — in general — hold for maximization of convex functions.
In this lecture we will conclude our discussion on upper bounds and summarize these bounds for various assumptions on the nature of the convex objective function. We shall then derive a lower bound for the convergence rate of gradient descent on convex objectives that are $\beta$-smooth $\alpha$-strongly convex.

## 2 Convergence rate

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

In the above table, $D_1$ denotes the initial suboptimality, i.e., the $L2$ distance of the initial guess $x_1$ from the optimal point $x^*$. So, $D_1 = \|x_1 - x^*\|_2^2$. As one would expect, assuming smoothness leads to a faster convergence rate compared to the case when the objective is assumed to only be L-Lipschitz. When we assume furthermore, for instance strong convexity and smoothness, we get exponential convergence (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).

One *seemingly weird* observation from the above table is that, when you consider 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. A $\lambda$-strongly convex function is always lower bounded by a quadratic of curvature $\lambda$. In addition to this, when we assume that it is L-Lipschitz, the gradients cannot exceed $L$. Hence, we're interested only in functions whose curvature is upper bounded by $\frac{L}{\lambda}$. This usually happens when we restrict ourselves to a particular subset of the domain of the convex objective where these properties hold. And when these 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

A black box model assumes that we do 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, ...). 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 $\epsilon$-approximate minimum for the convex function $f$.

Specifically, 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 $g_1, g_2, g_3, ....$ Further, in our analysis, we assume that the initial guess $x_1$ is always the zero vector ($x_1 = \mathbf{0}$). The optimizer updates the variable $x$ (interchangeably referred to as the parameter vector) using an update rule that satisfies the following criterion.

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

Here $t$ is an index into the number of steps the optimizer is run for. The assumption that $x_1 = \mathbf{0}$ is 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 [1].

## 4 A Taxonomy of Optimization Methods

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

### 4.1 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 (gradients, for example, are examples of such higher-level information). Bisection method, genetic algorithms, simulated annealing and Metropolis method are a few techniques that *can* fall under this category.

### 4.2 First order methods

These methods can inquire the value of the function $f$ and its first derivative (gradient or Jacobian) of the function $\nabla f$ at the current guess $x$. These methods are widely used for optimization in machine learning problems. Some of the methods include gradient descent, Nesterov's accelerated gradient methods, Polyak's momentum (some of which will be covered in subsequent lectures).

### 4.3 Second order methods

These methods require the value of the function $f$, its first derivative (gradient or Jacobian) $\nabla f$, 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 [2].

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

[2]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-Newton methods

## 5 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 and ADAM.

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.

## 6 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.

### 6.1 Why are lower bounds useful?

Lower bounds are very useful because they tell you what's the best you can do using a given optimizer. If not for lower bounds, a lot of research energy would be spent in designing better optimizers when you can't actually do any better. Caveats here are that, existence of a lower bound does not imply that an optimizer exists that attains the lower bound on convergence rate. Also, lower bounds do not also tell you about how tight they are, given that they are mostly proved by contrived examples.

### 6.2 Lower bounds for smooth and strongly convex objectives

Now, we proceed to derive a lower bound for an objective function, which, in addition to being convex, is also $\alpha$-strongly convex and $\beta$-smooth. Before we proceed, we outline some notation that will be used in the proof.

The *condition number* for a matrix will be denoted by $\kappa$. It is the ratio of the largest singluar value of the matrix to its smallest singluar value. If the condition number of a matrix is infinity, then the corresponding linear system is termed *singular*. If the condition number is too large (but not infinity), then the corresponding linear system is termed *ill-conditioned* (or *poorly conditioned*).

And from the black box model we defined in Section 3, we also have the initial guess $x_1 = \mathbf{0}$. Also recall that $x_{t+1} \in span\,(g_1, g_2, ....g_t)$, where $g_i$ is the gradient at the $i^{th}$ time step.

**Theorem 1.** *(Theorem 3.15 from [1]) ($\kappa > 1$) There exists a $\beta$-smooth and $\alpha$-strongly convex function $f\colon l_2 \mapsto \mathbb{R}$ 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 $\kappa$,*

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

*Proof.* As is typical of lower bound proofs, we prove this theorem by constructing an example. The example function we construct is an $\ell_2$ function. Informally speaking, $\ell_2$ 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

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

Using the operator defined above, we can define the following quadratic function.

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

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 (as $A$ is the Hessian of a quadratic function $f$, and Hessians, by definition, are symmetric),

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

Also note that $f$ is $\alpha$-strongly convex (the $\frac{\alpha}{2}\|x\|^2$ term ensures that) and $\beta$-smooth. $\beta$-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 $f$.

$$\nabla f(x) = \frac{\alpha(k-1)}{4}\left(Ax - 2e_1\right) + \alpha x$$

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

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

Using this expression, it is easy to show — by mathematical induction — that if $x_{t-1}$ has non-zero entries upto element at index $t-1$, then $x_t$ will have non-zero entries upto 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

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

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

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

If we differentiate $f$ and set $\nabla f$ to $0$, we obtain an infinite linear system, of the following form.

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

The solution of the above system is given by

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

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

$$\begin{aligned} f(x_t) - f(x^*) &\geq \frac{\alpha}{2}\|x_t - x^*\|^2 \\ &\geq \frac{\alpha}{2}\sum_{i=t}^{\infty}(x_t^*(i))^2 \\ &= \frac{\alpha}{2}\sum_{i=t}^{\infty}\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{2i} \\ &= \frac{\alpha}{2}\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{2(t-1)}||x_1 - x^*||^2 \end{aligned} \tag{6}$$

This proves the theorem.

$\square$

## References

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