ECE 901: Large-scale Machine Learning and Optimization **Spring 2018**

Lecture 4 — 02/01/2018

Lecturer: Dimitris Papailiopoulos *Scribe: Pradyot Prakash, Muni Sreenivas Pydi*

Note: These lecture notes are still rough, and have only have been mildly proofread.

4.1 Outline

- Importance of convexity
- Gradient descent: understand its mechanics and convergence rates (i.e., how fast does the algorithm reach the optimal solution)
- Why and how the structure of the objective function helps in solving an optimization problem? In general, without any structure, it is difficult to say anything useful.

Main question for today: Why is convexity useful and how to exploit it algorithmically?

4.2 Convex functions

We will look at convex functions of the form $f : \mathbb{X}^d \to \mathbb{R}$, where $\mathbb{X} \subset \mathbb{R}$. Recall that the definition of a convex function is,

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y) \quad \forall x,\ y \in \mathbb{X}^d \tag{4.1}$$

Let's assume that f is differentiable. In general, this need not be true and in that case subgradients are useful. f being differentiable implies that,

$$f(x) \geq f(x_0) + \langle \nabla f(x_0), x - x_0 \rangle \quad \forall x,\ x_0 \in \mathbb{X}^d \tag{4.2}$$

This is equivalent to saying that $f(x)$ is $\mathcal{O}(a^T x)$, i.e., lower bounded by a linear function.

Remark: 1^{st} order Taylor approximation of f operates as its "global underestimator". These hyperplanes always exist $\forall\ d$.

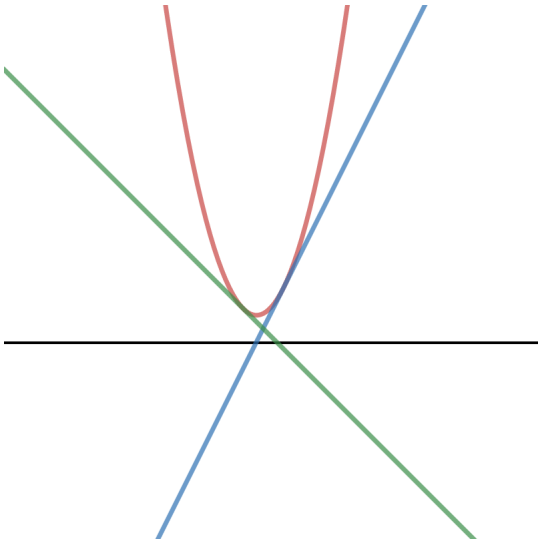

Figure 4.1. A convex function and it's two underapproximations

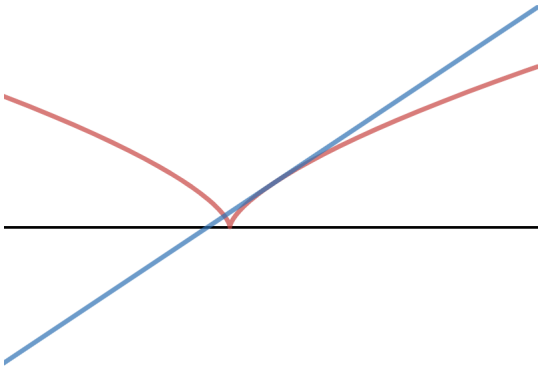

Figure 4.2. The tangent at a point for a non-convex function does not always lie below the function

This approximation does not hold for a non-convex function. This motivates Theorem 4.1.

Theorem 4.1. *A differentiable function is convex iff* $f(x) \geq f(x_0)+\langle \nabla f(x_0), x-x_0 \rangle \ \forall x,\ x_0 \in \mathbb{X}^d$.

Remark: For a convex function, "local minimizers" or "critical points" are globally optimal. This can be seen by observing that at a critical point x_0, $\nabla f(x_0) = 0$. From Theorem 4.1, $f(x) \geq f(x_0) + \langle 0, x - x_0 \rangle \Rightarrow f(x) \geq f(x_0) \ \ \forall x \in \mathbb{X}^d$. This property is an important

consequence of f being convex and not a defining trait of convex functions.

Question: Can we use the underestimator property to (approximately) "solve" $\min_x f(x)$?

Intuition: $f(x_0)+\langle\nabla f(x_0), x-x_0\rangle$ is a lower bound. So moving along the negative gradient direction will give us a smaller function value!

Modelling this in an iterative setup, let's assume that we have $x_{k+1} = x_k + u_k$ at the k^{th} step of the iteration. We need to compute u_k such that $\nabla f(x_k) \leq \epsilon$.

4.2.1 Gradient Descent

$$x_{k+1} = \arg\min_x \underbrace{f(x_k) + \langle\nabla f(x_k), x - x_k\rangle}_{\text{Minimize the linear approximation}} + \underbrace{\frac{1}{2\gamma}||x - x_k||_2^2}_{\text{but not by too much!}} \tag{4.3}$$

Just by optimizing the term in red, will give us the objective value of $-\infty$. This term essentially gives us the best direction to move in. The term in blue is a regularizer which controls the rate by which we move towards the value minimizing the function value. This function is quadratic in x, so taking the gradient and setting it to 0, will give us the minima,

$$\begin{aligned}
&\nabla_x\{f(x_k) + \langle\nabla f(x_k), x - x_k\rangle + \frac{1}{2\gamma}||x - x_k||_2^2\} = 0 \\
\implies &\nabla f(x_k) + \frac{1}{\gamma}(x - x_k) = 0 \\
\implies &\boxed{x_{k+1} = x_k - \gamma\nabla f(x_k)}.
\end{aligned}$$

This gives us the gradient descent algorithm. The choice of γ determines the amount by which we move in the negative gradient direction. If it's too small, we will take very small steps and the algorithm will take longer to converge. Make it too big, and we might overshoot the minima and keep on wobbling around the minima. Usually, γ is chosen through cross-validation. Approaches based on grid search, active search are active areas of research in Machine Learning. It is also interesting to note that γ may not be the same across iterations. Line search is one of the approaches used to find the best suited γ in such a setting.

4.2.2 Convergence Rates

Convergence rates tell us how fast we approach a (locally) optimal solution, *in the worst case* with respect to all functions in the class we are analyzing.

The general form of convergence rates will be something like

$$|f(x_T) - f^\star| \leq p(T)\ S(f)$$

where T is the number of iterations that we run the algorithm for, and $f^\star$ is the desired optimum. $|f(x_T) - f^\star|$ measures the distance to the optimum. $P(T)$ is the 'rate' function and $S(f)$ captures the dependence on f. Alternately, people also look at the distance of the current x_k value from the optimal $x^\star$, $||x_k - x^\star||_2$.

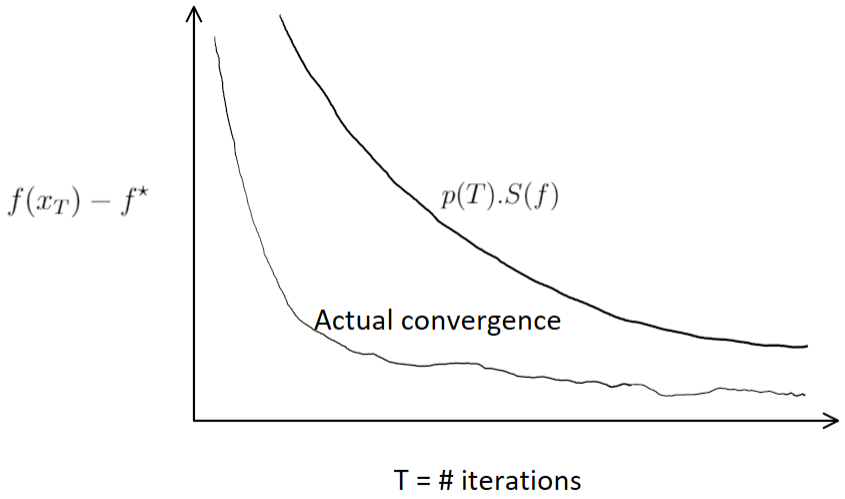

Figure 4.3. The actual convergence of the algorithm can be much faster than the theoretical convergence bound.

Warning 1: Worst case bounds may be too pessimistic and not too close to reality.

Warning 2: If some Algorithm A has a faster convergence rate than Algorithm B, then it does not necessarily imply that Algorithm A is actually faster in *practice*.

Remark: However, convergence rates are informative and can help us understand what structures allow for faster algorithms and sometimes can be good guides towards algorithm design.

Without imposing some structure on the function f, one can only prove so much. Let's focus our attention on Lipschitz functions. This class of function is more well behaved and does not change too rapidly. Now we see a theorem on the convergence rate of gradient descent on Lipschitz functions.

Theorem 4.2. *Let f be convex. Assume that $||x_1 - x^\star||_2 \leq R$ and $||\nabla f(x)||_2 \leq L$ (which implies that f is L-Lipschitz). Then, if we set $\gamma = \frac{R}{L\sqrt{T}}$,*

$$f\Big(\frac{1}{T}\sum_{i=1}^{T} x_k\Big) - f(x^\star) \leq \frac{RL}{\sqrt{T}}. \tag{4.4}$$

Proof: We start with the under estimator:

$$f(x_k) - f(x^\star) \leq \langle \nabla f(x_k), x_k - x^\star \rangle = \Big\langle \frac{x_k - x_{k+1}}{\gamma}, x_k - x^\star \Big\rangle.$$

We will use the fact that $a^T b = \frac{1}{2}\big\{||a||^2 + ||b||^2 - ||a-b||^2\big\}$, which implies

$$f(x_k) - f(x^\star) \leq \frac{1}{2\gamma}\big\{||x_k - x^\star||^2 + \underbrace{||x_k - x_{k+1}||^2}_{= \gamma^2||\nabla f(x_k)||^2 \leq \gamma^2 L^2} - ||x_{k+1} - x^\star||^2\big\}.$$

Hence,

$$\begin{aligned}
f(x_k) - f(x^\star) &\leq \frac{1}{2\gamma}\big\{||x_k - x^\star||^2 - ||x_{k+1} - x^\star||^2\big\} + \frac{\gamma L^2}{2} \\
f(x_{k-1}) - f(x^\star) &\leq \frac{1}{2\gamma}\big\{||x_{k-1} - x^\star||^2 - ||x_k - x^\star||^2\big\} + \frac{\gamma L^2}{2} \\
&\vdots \\
f(x_0) - f(x^\star) &\leq \frac{1}{2\gamma}\big\{||x_0 - x^\star||^2 - ||x_1 - x^\star||^2\big\} + \frac{\gamma L^2}{2}
\end{aligned}$$

Summing the above series of equations:

$$\begin{aligned}
\sum_{t=1}^{T} \big(f(x_t) - f(x^\star)\big) &\leq \frac{-||x_{T+1} - x^\star||^2 + ||x_0 - x^\star||^2}{2\gamma} + T\frac{\gamma L^2}{2} \\
\implies \frac{1}{T}\sum_{t=1}^{T} f(x_t) - f(x^\star) &\leq \frac{||x_0 - x^\star||^2}{2\gamma T} + \frac{\gamma L^2}{2}
\end{aligned}$$

Since the function f is convex, we have

$$f\Big(\frac{1}{T}\sum_{i=1}^{T} x_k\Big) - f(x^\star) \leq \frac{1}{T}\sum_{t=1}^{T} f(x_t) - f^\star \leq \frac{R^2}{2\gamma T} + \frac{\gamma L^2}{2}$$

Now, we find the tightest upper bound by minimizing the right hand side of the above equation with respect to γ.

$$\min_\gamma \frac{R^2}{2\gamma T} + \frac{\gamma L^2}{2} \implies \gamma = \frac{R}{L\sqrt{T}}.$$

$$\therefore \quad f\Big(\frac{1}{T}\sum_{i=1}^{T} x_k\Big) - f(x^\star) \leq \frac{RL}{\sqrt{T}}.$$

□

Corollary 4.3. *For ϵ approximation of the convergence rate in Theorem 4.2 (i.e. for $\epsilon = \frac{RL}{\sqrt{t}}$), we need $T = \frac{R^2L^2}{\epsilon^2}$ number of steps.*

One may question the feasibility of the assumptions made in the statement of the above Theorem. In practice, the function f may not always be convex and depends on the model we are optimizing. For example, SVMs have a convex loss-function while the famous Neural networks (in most cases) don't. The Lipschitzness of f is quite often true as most of the commonly used loss functions (logistic, hinge) have a bounded gradient.