Course Notes for EE227C (Spring 2018): Convex Optimization and Approximation

Instructor: Moritz Hardt
Email: hardt+ee227c@berkeley.edu

Graduate Instructor: Max Simchowitz
Email: msimchow+ee227c@berkeley.edu

October 15, 2018

6 Discovering acceleration

In this lecture, we seek to find methods that converge faster than those discussed in previous lectures. To derive this accelerated method, we start by considering the special case of optimizing quadratic functions. Our exposition loosely follows Chapter 17 in Lax's excellent text [Lax07].

6.1 Quadratics

Definition 6.1 (Quadratic function)**.** A quadratic function $f\colon \mathbb{R}^n \to \mathbb{R}$ takes the form:

$$f(x) = \frac{1}{2}x^T Ax - b^T x + c,$$

where $A \in S^n$, $b \in \mathbb{R}^n$ and $c \in \mathbb{R}$.

Note that substituting $n = 1$ into the above definition recovers the familiar univariate quadratic function $f(x) = ax^2 + bx + c$ where $a, b, c \in \mathbb{R}$, as expected. There is one subtlety in this definition: we restrict A to be symmetric. In fact, we could allow $A \in \mathbb{R}^{n\times n}$ and this would define the same class of functions, since for any $A \in \mathbb{R}^{n\times n}$ there is a symmetric matrix $\tilde{A} = \frac{1}{2}\left(A + A^T\right)$ for which:

$$x^T Ax = x^T \tilde{A}x \;\forall x \in \mathbb{R}^n.$$

Restricting $A \in S^n$ ensures each quadratic function has a *unique* representation.

The gradient and Hessian of a general quadratic function take the form:

$$\nabla f(x) = Ax - b$$
$$\nabla^2 f(x) = A.$$

Note provided A is non-singular, the quadratic has a unique critical point at:

$$x^* = A^{-1}b.$$

When $A \succ 0$, the quadratic is *strictly convex* and this point is the unique global minima.

6.2 Gradient descent on a quadratic

In this section we will consider a quadratic $f(x)$ where A is positive definite, and in particular that:

$$\alpha I \preceq A \preceq \beta I,$$

for some $0 < \alpha < \beta$. This implies that f is α-strongly convex and β-smooth.

From Theorem **??** we know that under these conditions, gradient descent with the appropriate step size converges linearly at the rate $\exp\left(-t\frac{\alpha}{\beta}\right)$. Clearly the size of $\frac{\alpha}{\beta}$ can dramatically affect the convergence guarantee. In fact, in the case of a quadratic, this is related to the *condition number* of the matrix A.

Definition 6.2 (Condition number)**.** Let A be a real matrix. Its *condition number* (with respect to the Euclidean norm) is:

$$\kappa(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)},$$

the ratio of its largest and smallest eigenvalues.

So in particular, we have that $\kappa(A) \leqslant \frac{\beta}{\alpha}$; henceforth, we will assume that α, β correspond to the minimal and maximal eigenvalues of A so that $\kappa(A) = \frac{\beta}{\alpha}$. It follows from Theorem **??** that gradient descent with a constant step size $\frac{1}{\beta}$ converges as

$$\|x_{t+1} - x^*\|^2 \leqslant \exp\left(-t\frac{1}{\kappa}\right)\|x_1 - x^*\|^2.$$

In many cases, the function f is ill-conditioned and κ can easily take large values. In this case, case convergence could be very slow as we will need $t > \kappa$ before the error gets small. Can we do better than this?

To answer this question, it will be instructive to analyze gradient descent specifically for quadratic functions, and derive the convergence bound that we previously proved for any strongly convex smooth functions. This exercise will show us where we are losing performance, and suggest a method that can attain better guarantees.

Theorem 6.3. *Assume $f : \mathbb{R}^n \to \mathbb{R}$ is a quadratic where the quadratic coefficient matrix has a condition number κ. Let x^* be an optimizer of f, and let x_t be the updated point at step t using gradient descent with a constant step size $\frac{1}{\beta}$, i.e. using the update rule $x_{t+1} = x_t - \frac{1}{\beta}\nabla f(x_t)$. Then:*

$$\|x_{t+1} - x^*\|^2 \leqslant \exp\left(-\frac{t}{\kappa}\right)\|x_1 - x^*\|^2.$$

Proof. Consider the quadratic function

$$f(x) = \frac{1}{2}x^T Ax - b^T x + c\,,$$

where A is a symmetric $n \times n$ matrix, $b \in \mathbb{R}^n$ and $c \in \mathbb{R}$. A gradient descent update with step size η_t takes the form:

$$x_{t+1} = x_t - \eta_t \nabla f(x_t) = x_t - \eta_t\,(Ax_t - b)$$

Subtracting x^* from both sides of this equation and using the property that $Ax^* - b = \nabla f(x^*) = 0$:

$$\begin{aligned}
x_{t+1} - x^* &= (x_t - \eta_t\,(Ax_t - b)) - (x^* - \eta_t\,(Ax^* - b)) \\
&= (I - \eta_t A)(x_t - x^*) \\
&= \prod_{k=1}^{t}(I - \eta_k A)(x_1 - x^*)\,.
\end{aligned}$$

Thus,

$$\|x_{t+1} - x^*\|_2 \leqslant \left\|\prod_{k=1}^{t}(I - \eta_t A)\right\|_2 \|x_1 - x^*\|_2 \leqslant \left(\prod_{k=1}^{t}\|I - \eta_k A\|_2\right)\|x_1 - x^*\|_2\,.$$

Set $\eta_k = \frac{1}{\beta}$ for all k. Note that $\frac{\alpha}{\beta}I \preceq \frac{1}{\beta}A \preceq I$, so:

$$\left\|I - \frac{1}{\beta}A\right\|_2 = 1 - \frac{\alpha}{\beta} = 1 - \frac{1}{\kappa}.$$

It follows that

$$\|x_{t+1} - x^*\|_2 \leqslant \left(1 - \frac{1}{\kappa}\right)^t \|x_1 - x^*\|_2 \leqslant \exp\left(-\frac{t}{\kappa}\right)\|x_1 - x^*\|_2\,. \qquad \blacksquare$$

6.3 Connection to polynomial approximation

In the previous section, we proved an upper bound on the convergence rate. In this section, we would like to improve on this. To see how, think about whether there was any point in the argument above where we were careless? One obvious candidate is that our choice of step size, $\eta_k = \frac{1}{\beta}$, was chosen rather arbitrarily. In fact, by choosing the sequence η_k we can select *any* degree-t polynomial of the form:

$$p(A) = \prod_{k=1}^{t} (I - \eta_k A)\,.$$

Note that:

$$\|p(A)\| = \max_{x \in \lambda(A)} |p(x)|$$

where $p(A)$ is a matrix polynomial, and $p(t)$ is the corresponding scalar polynomial. In general, we may not know the set of eigenvalues $\lambda(A)$, but we do know that all eigenvalues are in the range $[\alpha, \beta]$. Relaxing the upper bound, we get

$$\|p(A)\| \leqslant \max_{x \in [\alpha,\beta]} |p(x)|\,.$$

We can see now that we want a polynomial $p(a)$ that takes on small values in $[\alpha, \beta]$, while satisfying the additional normalization constraint $p(0) = 1$.

6.3.1 A simple polynomial solution

A simple solution has a uniform step size $\eta_t = \frac{2}{\alpha+\beta}$. Note that

$$\max_{x \in [\alpha,\beta]} \left|1 - \frac{2}{\alpha+\beta}x\right| = \frac{\beta-\alpha}{\alpha+\beta} \leqslant \frac{\beta-\alpha}{\beta} = 1 - \frac{1}{\kappa},$$

recovering the same convergence rate we proved previously. The resulting polynomial $p_t(x)$ is plotted in Figure 1 for degrees $t = 3$ and $t = 6$, with $\alpha = 1$ and $\beta = 10$. Note that doubling the degree from three to six only halves the maximum absolute value the polynomial attains in $[\alpha, \beta]$, explaining why convergence is so slow.

6.4 Chebyshev polynomials

Fortunately, we can do better than this by speeding up gradient descent using Chebyshev polynomials. We will use Chebyshev polynomials of the first kind, defined by the recurrence relation:

$$\begin{aligned} &T_0(a) = 1, \quad T_1(a) = a \\ &T_k(a) = 2aT_{k-1}(a) - T_{k-2}(a), \text{ for } k \geqslant 2\,. \end{aligned}$$

Figure 2 plots the first few Chebyshev polynomials.

Why Chebyshev polynomials? Suitably rescaled, they minimize the absolute value in a desired interval $[\alpha, \beta]$ while satisfying the normalization constraint of having value 1 at the origin.

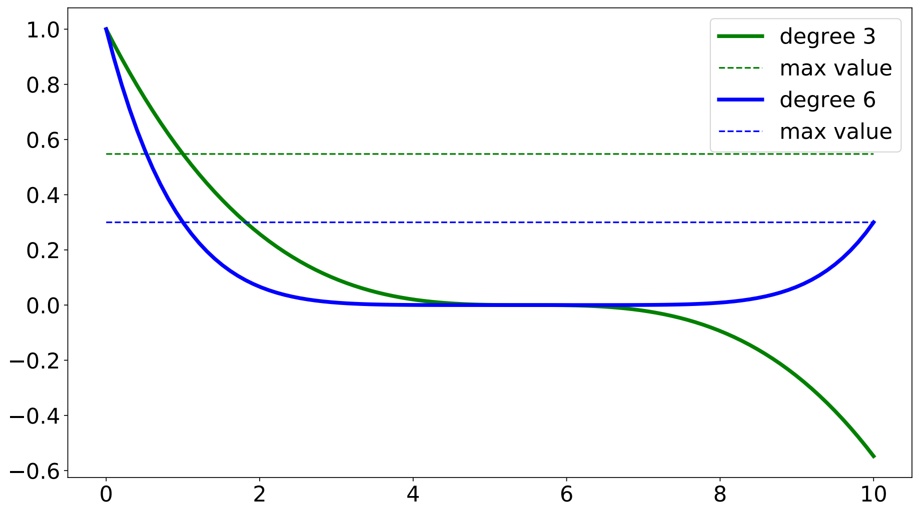

Figure 1: Naive Polynomial

Recall that the eigenvalues of the matrix we consider are in the interval $[\alpha, \beta]$. We need to rescale the Chebyshev polynomials so that they're supported on this interval and still attain value 1 at the origin. This is accomplished by the polynomial

$$P_k(a) = \frac{T_k\left(\frac{\alpha+\beta-2a}{\beta-\alpha}\right)}{T_k\left(\frac{\alpha+\beta}{\beta-\alpha}\right)}.$$

We see on figure 3 that doubling the degree has a much more dramatic effect on the magnitude of the polynomial in the interval $[\alpha, \beta]$.

Let's compare on figure 4 this beautiful Chebyshev polynomial side by side with the naive polynomial we saw earlier. The Chebyshev polynomial does much better: at around 0.3 for degree 3 (needed degree 6 with naive polynomial), and below 0.1 for degree 6.

6.4.1 Accelerated gradient descent

The Chebyshev polynomial leads to an accelerated version of gradient descent. Before we describe the iterative process, let's first see what error bound comes out of the Chebyshev polynomial.

So, just how large is the polynomial in the interval $[\alpha, \beta]$? First, note that the maximum value is attained at α. Plugging this into the definition of the rescaled Chebyshev polynomial, we get the upper bound for any $a \in [\alpha, \beta]$,

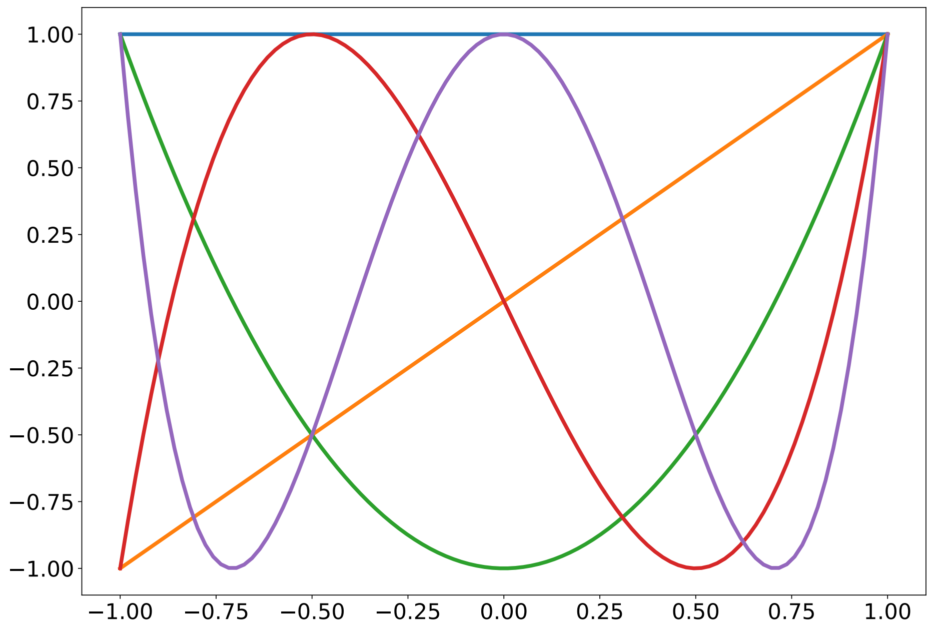

Figure 2: Chebychev polynomials of varying degrees.

$$|P_k(a)| \leqslant |P_k(\alpha)| = \frac{|T_k(1)|}{|T_K\left(\frac{\beta+\alpha}{\beta-\alpha}\right)|} \leqslant \left|T_K\left(\frac{\beta+\alpha}{\beta-\alpha}\right)^{-1}\right|.$$

Recalling the condition number $\kappa = \beta/\alpha$, we have

$$\frac{\beta+\alpha}{\beta-\alpha} = \frac{\kappa+1}{\kappa-1}.$$

Typically κ is large, so this is $1+\epsilon$, $\epsilon \approx \frac{2}{\kappa}$. Therefore, we have

$$|P_k(a)| \quad \leqslant |T_k(1+\epsilon)^{-1}|.$$

To upper bound $|P_k|$, we need to lower bound $|T_k(1+\epsilon)|$.
Fact: for $a > 1$, $T_k(a) = \cosh\left(k \cdot \operatorname{arccosh}(a)\right)$ where:

$$\cosh(a) = \frac{e^a + e^{-a}}{2}, \quad \operatorname{arccosh}(a) = \ln\left(x + \sqrt{x^2-1}\right).$$

Now, letting $\phi = \operatorname{arccosh}(1+\epsilon)$:

$$e^\phi = 1 + \epsilon + \sqrt{2\epsilon+\epsilon^2} \geqslant 1 + \sqrt{\epsilon}.$$

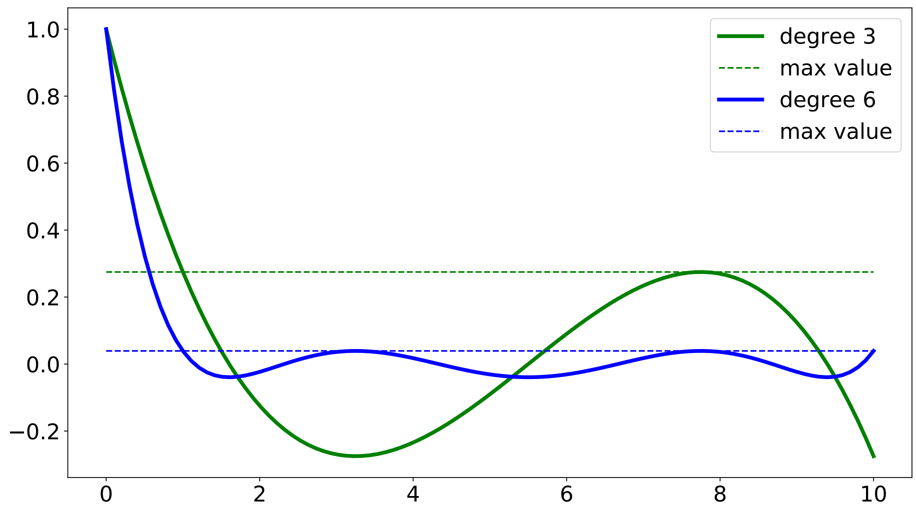

Figure 3: Rescaled Chebyshev

So, we can lower bound $|T_k(1+\epsilon)|$:

$$\begin{aligned}|T_k(1+\epsilon)| &= \cosh\left(k \operatorname{arccosh}(1+\epsilon)\right)\\ &= \cosh(k\phi)\\ &= \frac{(e^{\phi})^k + (e^{-\phi})^k}{2}\\ &\geqslant \frac{(1+\sqrt{\epsilon})^k}{2}.\end{aligned}$$

Then, the reciprocal is what we needed to upper bound the error of our algorithm, so we have:

$$|P_k(a)| \leqslant |T_k(1+\epsilon)^{-1}| \leqslant 2(1+\sqrt{\epsilon})^{-k}.$$

Thus, this establishes that the Chebyshev polynomial achieves the error bound:

$$\begin{aligned}\|x_{t+1} - x^*\| &\leqslant 2(1+\sqrt{\epsilon})^{-t}\,\|x_0 - x^*\|\\ &\approx 2\left(1+\sqrt{\frac{2}{\kappa}}\right)^{-t}\|x_0 - x^*\|\\ &\leqslant 2\exp\left(-t\sqrt{\frac{2}{\kappa}}\right)\|x_0 - x^*\|.\end{aligned}$$

This means that for large κ, we get quadratic savings in the degree we need before the error drops off exponentially. Figure 5 shows the different rates of convergence, we clearly see that the

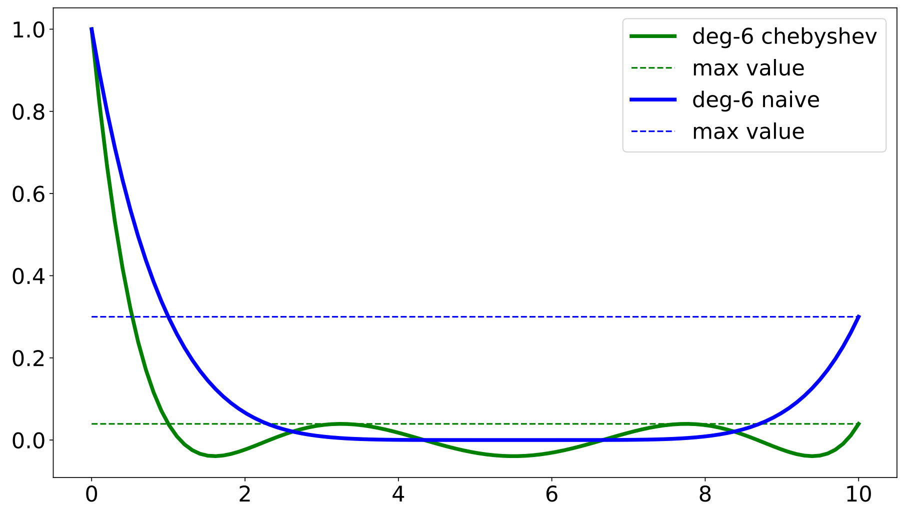

Figure 4: Rescaled Chebyshev VS Naive Polynomial

6.4.2 The Chebyshev recurrence relation

Due to the recursive definition of the Chebyshev polynomial, we directly get an iterative algorithm out of it. Transferring the recursive definition to our rescaled Chebyshev polynomial, we have:

$$P_{K+1}(a) = (\eta_k a + \gamma_k) P_k(a) + \mu_k P_{k-1}(a).$$

where we can work out the coefficients η_k, γ_k, μ_k from the recurrence definition. Since $P_k(0) = 1$, we must have $\gamma_k + \mu_k = 1$. This leads to a simple update rule for our iterates:

$$\begin{aligned} x_{k+1} &= (\eta_k A + \gamma_k) x_k + (1 - \gamma_K) x_{k-1} - \eta_k b \\ &= (\eta_k A + (1 - \mu_k)) x_k + \mu_k x_{k-1} - \eta_k b \\ &= x_k - \eta_k (A x_k - b) + \mu_k (x_k - x_{k-1}). \end{aligned}$$

We see that the update rule above is actually very similar to plain gradient descent except for the additional term $\mu_k(x_k - x_{k-1})$. This term can be interpreted as a *momentum* term, pushing the algorithm in the direction of where it was headed before. In the next lecture, we'll dig deeper into momentum and see how to generalize the result for quadratics to general convex functions.

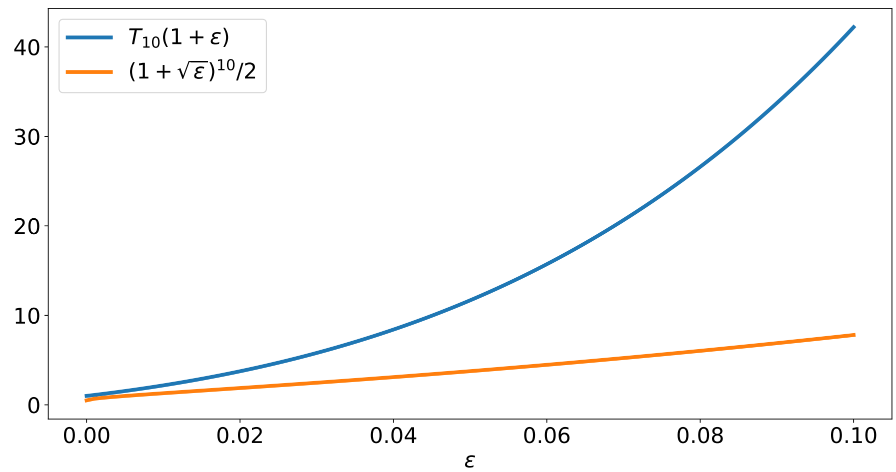

Figure 5: Convergence for naive polynomial and Chebyshev

References

[Lax07] Peter D. Lax. *Linear Algebra and Its Applications*. Wiley, 2007.