STAT 9912: Acceleration, Greed and Hedging in Optimization

Taught by Dr. Jason Altschuler
Notes written by Faraz Radhman

Contents

1 Setting the Stage

1.1 Optimization and Gradient Descent

Mathematical optimization tackles the set of problems that can be formulated as

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & x \in \mathcal{X} \end{aligned}$$

This course aims to study progress in first order optimizers: algorithms that aim to solve the above problem given an oracle that can compute the following given $\boldsymbol{x}$

1. $f(\boldsymbol{x})$
2. $\nabla f(\boldsymbol{x})$

The canonical algorithm to solve this problem is Gradient Descent (GD) and is described by the following update

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \alpha_t \nabla f(\boldsymbol{x}_t)$$

where $\{\alpha_t\}$ parameterizes different "schedules" for GD.

Here are some questions to start discussion:

Q1) Why move in the direction $-\nabla f$

Q2) Can GD converge? (with any step sizes)

Q3) How fast? (with optimal step sizes)

1.2 Why $-\nabla f$?

The gradient descent algorithm, as many other tools in science and engineering, can be derived from solving a linear approximation to the optimization objective.

Writing the first-order Taylor-expansion at $\boldsymbol{x}$ we get

$$f(\boldsymbol{x} + \boldsymbol{v}) \approx f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{v} \rangle$$

If we want to use this crude optimization to (try to) move $\boldsymbol{x}$ in a direction that reduces f, we can setup the following problem.

$$\begin{aligned} \underset{\boldsymbol{v}}{\text{minimize}} \quad & f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{v} \rangle \\ \text{subject to} \quad & \|\boldsymbol{v}\|_2^2 \leq 1 \end{aligned}$$

In English, this asks: "If we were only allowed to move one unit away from $\boldsymbol{x}$ what direction should we move to decrease $f(\boldsymbol{x})$ the most?".

We will find that

$$\boldsymbol{v}^* \propto -\nabla f(\boldsymbol{x})$$

A comment should be made that the choice of the ℓ_2 norm here is not obvious. Different norms will induce different update rules creating a general class of algorithms called methods of steepest descent.

2 Quadratics

The simplest place to start our study is in the optimization of convex quadratic functions.

We can consider any function that can be expressed as

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}\boldsymbol{x} - \boldsymbol{b}^\top \boldsymbol{x} + c$$

where $\boldsymbol{H} \succeq 0$[1] (necessary for f to be convex) and $\boldsymbol{H}$ is symmetric.[2]

One assumption we will make is that $m\boldsymbol{I} \preceq \boldsymbol{H} \preceq M\boldsymbol{I}$ where $m, M > 0$ making the f M-smooth and m-strongly convex.

This problem has a closed-form optimal solution given by

$$\boldsymbol{x}^* = \boldsymbol{H}^{-1}\boldsymbol{b}$$

which can be derived via the FOC. This equality will be useful for deriving convergence rates for GD on quadratics.

2.1 Do we converge?

Much can be said about GD by studying the difference to between the current solution and the optimal solution

$$\boldsymbol{x}_t - \boldsymbol{x}^*$$

over time. Intuitively we want the distance to go down... and fast!

Plugging in the update rule we can see how

$$(\boldsymbol{x}_{t+1} - \boldsymbol{x}^*) = (\boldsymbol{x}_t - \boldsymbol{x}^*) - \alpha_t \nabla f(\boldsymbol{x})$$

Note that

$$\begin{aligned}
\nabla f(\boldsymbol{x}) &= \boldsymbol{H}\boldsymbol{x}_t - \boldsymbol{b} \\
&= \boldsymbol{H}\boldsymbol{x}_t - \boldsymbol{H}\boldsymbol{H}^{-1}\boldsymbol{b} \\
&= \boldsymbol{H}\boldsymbol{x}_t - \boldsymbol{H}\boldsymbol{x}^* \\
&= \boldsymbol{H}\left(\boldsymbol{x}_t - \boldsymbol{x}^*\right)
\end{aligned}$$

where the third lines comes from the closed form solution $\boldsymbol{x}^* = \boldsymbol{H}^{-1}b$.

[1] $\boldsymbol{H} \succeq 0 \Leftrightarrow \boldsymbol{x}^\top \boldsymbol{H}\boldsymbol{x} \geq 0, \quad \forall \boldsymbol{x} \in \mathbb{R}^m \Leftrightarrow \boldsymbol{H}$ is positive semidefinite (PSD) by defn. $\Leftrightarrow \boldsymbol{x}^\top \boldsymbol{H}\boldsymbol{x}$ is convex.

[2] We can assume $\boldsymbol{H}$ is symmetric without loss of generality because if $\boldsymbol{H}$ is not symmetric, we can replace it with $\hat{\boldsymbol{H}} = \frac{1}{2}\left(\boldsymbol{H} + \boldsymbol{H}^\top\right)$, which yields an equivalent function $\hat{f}$

If we plug in this term for $\nabla f(x)$ we get a recurrence

$$\begin{aligned}(\boldsymbol{x}_{t+1} - \boldsymbol{x}^*) &= (\boldsymbol{x}_t - \boldsymbol{x}^*) - \alpha_t \boldsymbol{H}(bx_t - \boldsymbol{x}^*) \\ &= (\boldsymbol{I} - \alpha_t \boldsymbol{H})(\boldsymbol{x}_t - \boldsymbol{x}^*)\end{aligned}$$

This is a Linear Dynamical System (LDS). Recall that an LDS

$$\boldsymbol{z}_{t+1} = \boldsymbol{A}\boldsymbol{z}_t$$

will converge so long as $|\lambda_i| < 1$ for all eigenvalues λ_i of $\boldsymbol{A}$.

Recall that we assumed that $m\boldsymbol{I} \preceq \boldsymbol{H} \preceq M\boldsymbol{I}$, so our dynamics matrix $\boldsymbol{A}_t = (\boldsymbol{I} - \alpha_t \boldsymbol{H})$ we know that the eigenvalues for this dynamical system sit in

$$\lambda_i \in [1 - \alpha M, 1 - \alpha m]$$

Hence, if set all steps to an adequately chosen $\alpha_t = \alpha$ the above LDS will converge to 0.

2.2 How fast do we converge?

Before we can answer this question we need to define what it means to converge "quickly". Consider the metric

$$R_T = \frac{\|\boldsymbol{x}_T - \boldsymbol{x}^*\|}{\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|}$$

which captures the fraction of the original distance remaining in our optimization. Naturally we want this as low as possibile. Taking the geometric mean $(R_T)^{1/T}$ can tell us about the average contraction rate of our error per optimization step.

In our analysis we will focus on worst-case analysis over the functions f, and random inits x_0, so if $\alpha_t = \alpha$ is constant, then it suffices to consider R_1. We will start here

Consider the following minimax problem where we look for the algorithm (parameetrized by constant step-size α) that minimizes the the worst case 1-step contraction rate R_1

$$\operatorname*{argmin}_{\alpha} \max_{f, \boldsymbol{x}_0} R_1 := \frac{\|\boldsymbol{x}_1 - \boldsymbol{x}^*\|}{\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|}$$

Using the LDS update to expand the numerator,

$$\operatorname*{argmin}_{\alpha} \max_{\boldsymbol{H}, \boldsymbol{x}_0} \frac{\|(\boldsymbol{I} - \alpha \boldsymbol{H})(\boldsymbol{x}_0 - \boldsymbol{x}^*)\|}{\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|}$$

Some may recognize the inner maximization as the matrix norm of $(\boldsymbol{I} - \alpha \boldsymbol{H})$. Since $\boldsymbol{H}$ is symmetric this is equivalent to the maximum eigenvalue.[3] So we can express the minimax as

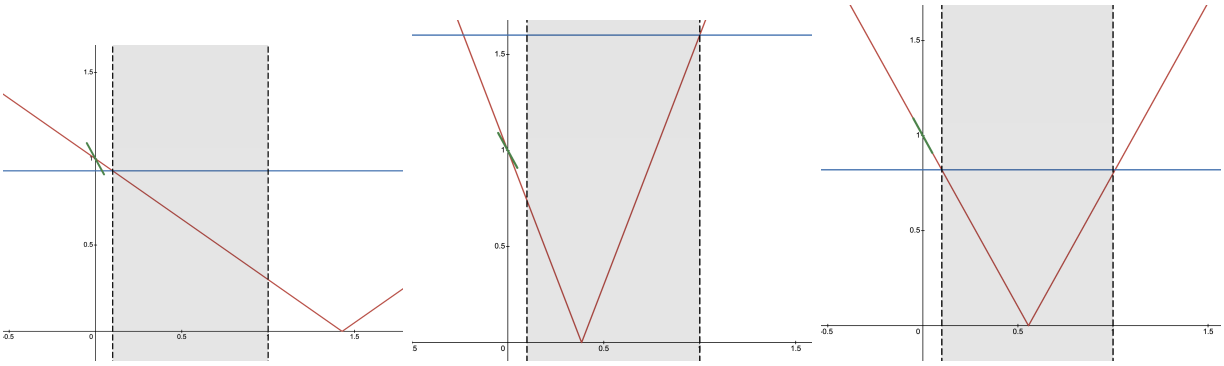
$$\operatorname*{argmin}_{\alpha} \max_{\boldsymbol{H}} \lambda_{\max}(\boldsymbol{I} - \alpha \boldsymbol{H})$$

[3] More information on the matrix norm can be found here.

Since we assumed $m\boldsymbol{I} \preceq \boldsymbol{H} \preceq M\boldsymbol{I}$ what we can further simplify to

$$\underset{\alpha}{\operatorname{argmin}} \max_{m \leq \lambda \leq M} |1 - \alpha\lambda|$$

There is a nice geometric interpretation to this problem. If we think $|1 - \alpha\lambda|$ as a linear function, we can think of sweeping the slope of an absolute function pinned at $(0, 1)$.



You can play around with this exact plot on Desmos here.

Analytically, since the function is a (peice-wise) linear, we know that the maximum is at either boundary allowing us to write

$$\underset{\alpha}{\operatorname{argmin}} \max \{|1 - \alpha m|, |1 - \alpha M|\}$$

With arithmetic we can arrive at

$$\alpha^* = \frac{2}{M + m}$$

If we plug this into R_1 we get a **Convergence Rate:**

$$R_1 = \frac{M - m}{M + m} = 1 - \mathcal{O}\left(\frac{1}{k}\right)$$

If we want **Iteration Complexity** (a.k.a. running time): Recall:

$$error_n \leq R_1^n \cdot error_0$$

Suppose we want to ask how many iterations need to get $error_n = \epsilon$

i.e.

$$R_1^n \leq \epsilon \Rightarrow n = k \log\left(\frac{1}{\epsilon}\right)$$

Note the fact: $1 - \delta =$

2.3 Optimal Schedules

In the above section, we derived optimal rates for GD when we can pick a single fixed step-size. The natural follow-up is to ask if we can do better with non-constant step-sizes.

2.3.1 Warmup: Spectral Annihilation[4]

We spent some time deriving the optimal constant step-size for worst-case qudaratic problems with a given mininum and maximum eigenvalue. Before continuing, let's build understanding about what we can do if we have access to the entire eigen-spectrum of our problem.

In particular, let's start with the 1d-case with

$$f(x) = \frac{1}{2}Ax^2$$

The gradient $\nabla f(x) = Ax$, so after one step of gradient descent with step-size α, the next iterate is

$$x^+ = x - \alpha Ax$$

You'll notice that simply setting $\alpha = \frac{1}{A}$ yields

$$x^+ = x - \frac{A}{A}x = 0 = x^*$$

Allowing us to solve in one iteration. Now consider that we have a 2d problem.

Consider diagonal $A \in \mathbb{R}^{2\times 2}$ and

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} = \frac{1}{2}\left(A_{11}x_1^2 + A_{22}x_2^2\right)$$

The gradient $\nabla f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$, so after one step of gradient descent with step-size α, the next iterate is

$$x_1^+ = x - \alpha A_{11}x$$
$$x_2^+ = x - \alpha A_{22}x$$

Like the 1d case, we can simply select the diagonal elements (eigen-values for general matrices) to be our reciprocal step-sizes to arrive knock the corresponding entry right to the optimum $\boldsymbol{x}^* = \boldsymbol{0}$. Let $\alpha_1 = \frac{1}{A_{11}}$, $\alpha_2 = \frac{1}{A_{22}}$, then after one step we knock out x_1

$$x_1^{++} = x_1^+ - \frac{A_{11}}{A_{11}}x_1 = 0$$
$$x_2^{++} = x_2^+ - \frac{A_{22}}{A_{11}}x_2 = \left(1 - \frac{A_{22}}{A_{11}}\right)x_2$$

And after the second step we get knock x_2.

$$x_1^+ = x_1 - \frac{1}{A_{22}}\cdot 0 = 0$$
$$x_2^+ = x_2^+ - \frac{A_{22}}{A_{22}}x_2^+ = 0$$

[4]Not included in original lectures.

This logic extends to d-dimensional problems and non-diagonal problems where we arrive at the optimum in d-steps. An important takeaway here is that **the order of step-sizes do not matter**. Because, the eigen-spectrum of a quadratic function is indentical at all points, the step with step-size reciprocal to the ith eigenvalue is will annihilate the ith entry *no matter where it currently is in the domain* i.e. no matter where the previous descent steps went. This is a nice property of quadratic functions but it does not generalize to quadratics, we will see this a common theme that shows up in later algorithms.

2.3.2 Optimal 2-Step Schedules

With the above insight, we can return to the case where we only assume bounds $0 < m \leq \lambda_i \leq M$ on the eigenvalues. For simplicity, we should start by computing an optimal schedule for $n = 2$ steps. The problem will take a familiar form,

$$\min_{\alpha,\beta} \max_{f, \boldsymbol{x}_0 \neq \boldsymbol{x}^*} \frac{\|\boldsymbol{x}_2 - \boldsymbol{x}^*\|}{\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|}$$

where α, β are our step-sizes.

Suppose R_1^* was the optimal one-step contraction. Our hope would be that the extra flexibility of step-size would allow $\sqrt{R_2^*} < R_1^*$.

To analyze this, we can folow very similar steps to above. The two-step error recurrence can be expressed as

$$\boldsymbol{x}_2 - \boldsymbol{x}^* = (\boldsymbol{I} - \beta \boldsymbol{H})(\boldsymbol{I} - \alpha \boldsymbol{H})(\boldsymbol{x}_0 - \boldsymbol{x}^*)$$

Note that the contraction term is a matrix polynomial of degree ≤ 2. .

$$\begin{aligned} p(\boldsymbol{H}) &= (\boldsymbol{I} - \beta \boldsymbol{H})(\boldsymbol{I} - \alpha \boldsymbol{H}) \\ &= \boldsymbol{I} - (\alpha + \beta)\boldsymbol{H} + \alpha\beta \boldsymbol{H}^2 \end{aligned}$$

In this expression we see that $p(\boldsymbol{0}) = \boldsymbol{I}$. This is analogous to our plot of $y = |1 - \alpha\lambda|$ above where the function was pinned $(0, 1)$ at. Since p pinned at $(\boldsymbol{0}, \boldsymbol{I})$ has two degrees of freedom and is parameterized by two step-size choices α, β, we have a one-to-one mapping with the set $\hat{\mathcal{P}}_2 = \{p \in \mathcal{P}_2 : p(\boldsymbol{0}) = \boldsymbol{I}\}$. Squinting at $p(\boldsymbol{H})$ in factorized form will tell us that α, β are the inverse roots of p. Easiest way to see this is we pass a scalar x and get $p(x) = (1 - \beta x)(1 - \alpha x)$. Setting $p(x) = 0$ we get solutions $x \in \{\alpha^{-1}, \beta^{-1}\}$.

This lets us write the min of the minimax as

$$\min_{\text{poly } p \in \hat{\mathcal{P}}_2} \max_{m\boldsymbol{I} \preceq \boldsymbol{H} \preceq M\boldsymbol{I}} R_2 = \|p(\boldsymbol{H})\|$$

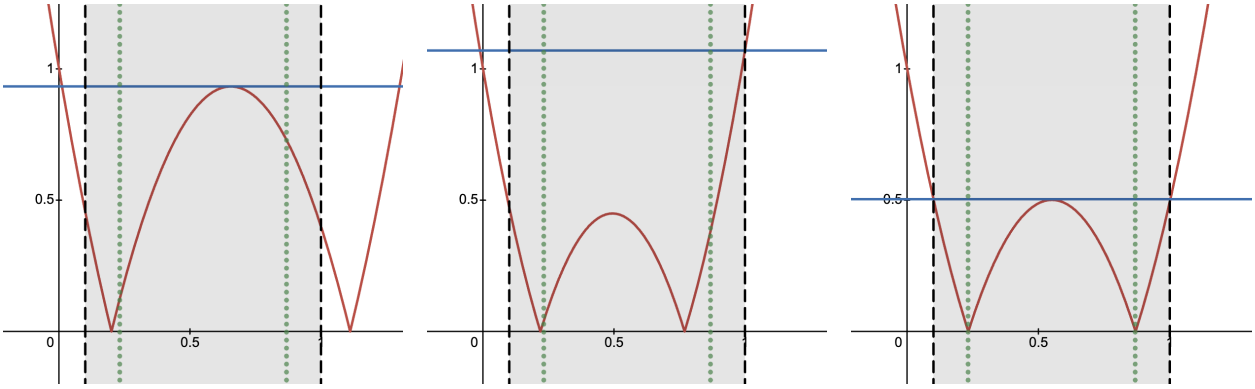
Because $\boldsymbol{H}$ is symmetric, we may digonalize as $\boldsymbol{H} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top$ and extract the orthogonal matrices out of the polynomial to get $p(\boldsymbol{H}) = \boldsymbol{Q}p(\boldsymbol{\Lambda})\boldsymbol{Q}^\top$. Additionally, matrix-norms are invariant under orthgonal transforms so $||\boldsymbol{Q}p(\boldsymbol{\Lambda})\boldsymbol{Q}^\top|| = ||p(\boldsymbol{\Lambda})||$ meaning that our inner maximization reduces to the worst-case eigenvalue once again.

$$\min_{p \in \hat{\mathcal{P}}_2} \max_{m \leq \lambda \leq M} |p(\lambda)|$$

In fact, if we expand p, this simplified form is an exact generalization of the 1-step picture that we drew above!

$$\min_{\alpha,\beta} \max_{m \le \lambda \le M} \left|1 - (\alpha+\beta)\lambda + \alpha\beta\lambda^2\right|$$

We can plot the polynomial and the maximum value over $[m, M]$ and analyze the worst-case rate. An interactive plot is available here.



Center too big | Right too big | Just right

The optimal roots (inverse step-sizes) are shown as the dotted green line at $\left\{\alpha^{-1}, \beta^{-1}\right\} = \left\{\frac{M+m}{2} \pm \frac{M-m}{2\sqrt{2}}\right\}$

2.3.3 The General Case

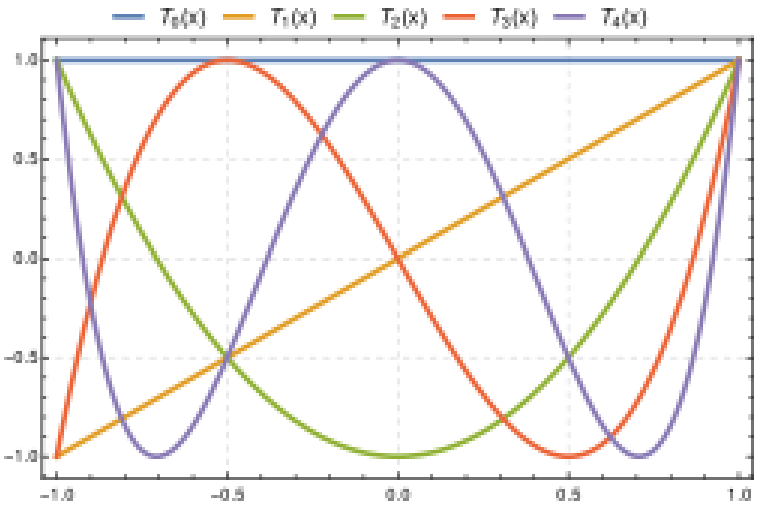
The general problem $n \ge 2$ steps with step sizes given by $\{\alpha_i\}_{i\in[n]}$ can be expressed as

$$R_n^* = \min_{\alpha,\beta} \max_{f, \boldsymbol{x}_0 \neq \boldsymbol{x}^*} \frac{\left\| \left(\prod_{i\in[n]} (\boldsymbol{I} - \alpha_i \boldsymbol{H}) \right) (\boldsymbol{x}_0 - \boldsymbol{x}^*) \right\|}{\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|}$$

All of the above steps may be repeated to arrive at the form

$$\min_{p \in \hat{\mathcal{P}}_n} \max_{m \le \lambda \le M} |p(\lambda)|$$

This problem is closely related the the Chebyshev Polynomials and the solution the n-step problem will turn out to be exactly the nth Chebyshev Polynomials T_n (modulo some affine transforms on the input/output). The next section will dive deeper into what the Chebyshev Polynomials, but for now we will just provide a graph and some rough intuition for why they are the solution.



Chebyshev Polynomials $T_n(x)$ for degrees $n = 1$ through $n = 5$ on $[-1, 1]$.

You can see in the above figure that the extrema of the Chebyshev polynomials lie in $\{-1, 1\}$. It should be intuitive how this corresponds to the absolute value of the polynomial all sharing a common maxima.

Take for granted that the solution to the minimax is

$$p_n^* = \frac{T_n(L(\lambda))}{T_n(L(0))} \tag{1}$$

where L affinely maps the eigenvalue range $[m, M] \to [-1, 1]$ which is the natural domain of the Chebyshev polynomial.[5]

The maximum value that T_n takes on is 1 so maximizing over λ we get

$$R_n^* = \frac{1}{T_n(L(0))}$$

We can understand the limiting optimal rate by taking the large n limit:

We get that

$$\begin{aligned}
\lim_{n\to\infty} R_n^* &= \lim_{n\to\infty} \frac{1}{T_n(L(0))} \\
&= \lim_{n\to\infty} \frac{2}{\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n + \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{-n}} \\
&= 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n \\
&= 2\left(1 - \frac{2}{\sqrt{\kappa}+1}\right)^n
\end{aligned}$$

[5]This works out to be $L(\lambda) = \frac{\lambda - m}{M - m} \cdot 2 - 1$. For $\lambda = 0$ we get $L(0) = \frac{-2m}{M-m} - 1 = \frac{-M-m}{M-m} = \frac{m+M}{m-M} = \frac{1+\kappa}{1-\kappa}$

The second line follows from the first definition of the Chebyshev polynomials which shows up in the following section. For now it suffices to take this as true.

Taking the geometric mean over n steps and dropping constants we see that the final optimal rate is

$$\mathcal{O}\left(\frac{1}{\sqrt{\kappa}}\right)$$

which is the famous "accelerated" rate for gradient descent.

Similarly we can get an interation complexity $R_n \leq \epsilon$ is $n = \Theta\left(\sqrt{k}\log(1/\epsilon)\right)$

Philosophical Aside:

Why was the optimal steps for $n = 2$ suboptimal for $n = 1$.

It means that we have fewer choices to pick from we have to pick something "in-between" to trade off different edge cases ("hedging").

Young 1953. Acceleration Methods FnTML.

2.4 The Chebyshev Polynomials and Accelerated Methods

Some more resources (Mason & Handscomb, Riulin, Trefethen)

Various Definitions of Chebyshev Polys:

- **Explicit:**
$$T_n(z) = \frac{\left(z - \sqrt{z^2-1}\right)^n + \left(z + \sqrt{z^2-1}\right)^n}{2}$$
- **Trigonometric:**
$$T_n(z) = \cos(n \cdot \arccos(z)), \quad z \in [-1, 1]$$
- **Roots:**
$$\left\{\cos\left(\frac{2t+1}{2n}\pi\right)\right\}_{t=0\ldots n-1}$$
  - $\left\{\frac{2t+1}{2n}\pi\right\}_{t=0\ldots n-1}$ are angles spread uniformly around the unit circle. So applying cos can be thought of as projecting the points from the unit circle onto the x-axis.
  - Figure 1 is a nice picture for the distribution of roots (uniformly) on a circle then projected on to the x-axis. it is called the arcsine distribution and will come up in random step-size schedules
- **3-term recurrence**
$$T_{n+1}(z) = 2zT_n(z) - T_{n-1}(z), \quad T_0(z) = 1, T_1(z) = z$$
- **Extremal:**
$$T_n(z)/2^{n-1} = \underset{p \text{ def } n, p \text{ monic}}{\operatorname{argmin}} \max_{z \in [-1,1]} |p(z)|$$

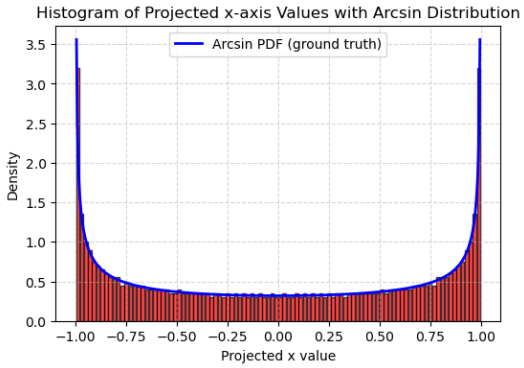


Figure 1: Distribution of roots of Chebyshev polynomials projected onto the x-axis (arcsine distribution).

– This is why they showed up in our step-size derivation above.

A core message to takeaway here is that there are many equivalent ways to derive step-size schedules yielding accelerated descent *and* they all correspond to some way to define the Chebyshev polynomials. This duality follow directly from the relationship between inverse roots and step-sizes.

2.4.1 Polyak's Heavy Ball

To begin we can look at the Polyak Heavy Ball Method which introduced the notion of momentum to first-order optimization. Polyak's Heavy Ball can be dervied via the 3-term recurrence.

Recall that the 3-term recurrence was

$$T_{n+1}(z) = 2zT_n(z) - T_{n-1}(z)$$

We need to shift / rescale in order to apply this recurrence to $p_{n+1}(\boldsymbol{H})$, our optimal contraction polynomials. We defined $L(\lambda) = a\lambda + b$ to handle scaling the input [6]. The RHS of the recurrence can then be expressed in terms of lambda $2(a\lambda + b)T_n(L(\lambda)) - T_{n-1}(L(\lambda))$, but we also need to rescale the out put as in (1) so taking this into accourn the optimal polynomial recurrence is

$$p_{n+1}(\lambda) = ((c_1 + c_2\lambda)\, p_n(\lambda) - c_3 p_{n-1}(\lambda))$$

for some values of c_1, c_2, c_3

Next, we want to go from this recurrence into something that we can implement. Let's start with some wishful thinking: for $n+1$ steps assume we contract according to $p_{n+1}(\boldsymbol{H})$ which is optimal, then write the contraction via the recurrence.

$$\begin{aligned}
\boldsymbol{x}_{n+1} - \boldsymbol{x}^* &= p_{n+1}(\boldsymbol{H})(\boldsymbol{x}_0 - \boldsymbol{x}^*) \\
&= ((c_1 + c_2\boldsymbol{H})\, p_n(\boldsymbol{H}) - c_3 p_{n-1}(\boldsymbol{H}))\, (\boldsymbol{x}_0 - \boldsymbol{x}^*) \\
&= (c_1 + c_2\boldsymbol{H})\, \underbrace{p_n(\boldsymbol{H})(\boldsymbol{x}_0 - \boldsymbol{x}^*)}_{=(\boldsymbol{x}_n - \boldsymbol{x}^*)} - c_3\, \underbrace{p_{n-1}(\boldsymbol{H})(\boldsymbol{x}_0 - \boldsymbol{x}^*)}_{=(\boldsymbol{x}_{n-1} - \boldsymbol{x}^*)}
\end{aligned}$$

[6] for some values a, b (we will only be concerned with the functional form for now)

Where the third line points out that the recurrence allows us to express the optimal $n+1$ step iterate in terms of our n and $n-1$ step iterates. Further rearranging gives.

$$\begin{aligned}\boldsymbol{x}_{n+1} - \boldsymbol{x}^* &= c_1\left(\boldsymbol{x}_n - \boldsymbol{x}^*\right) + c_2 \underbrace{\boldsymbol{H}\left(\boldsymbol{x}_n - \boldsymbol{x}^*\right)}_{=\nabla f(\boldsymbol{x}_n)} - c_3\left(\boldsymbol{x}_{n-1} - \boldsymbol{x}^*\right) \\ &= c_1\boldsymbol{x}_n + c_3'\boldsymbol{x}_{n-1} + c_2\nabla f(\boldsymbol{x}_n) - \left(c_3' + c_1\right)\boldsymbol{x}^*\end{aligned}$$

The first line identifies the second term as the gradient for the quadratic function. In rearranging we substitute $c_3' = -c_3$ for clarity. If we constraint $c_1 + c_3' = 1$ we can add $\boldsymbol{x}^*$ to both sides to get

$$\begin{aligned}\boldsymbol{x}_{n+1} &= c_1\boldsymbol{x}_n + c_3'\boldsymbol{x}_{n-1} + c_2\nabla f(\boldsymbol{x}_n) \\ &= \boldsymbol{x}_n - c_3' \underbrace{\left(\boldsymbol{x}_n - \boldsymbol{x}_{n-1}\right)}_{\text{momentum}} + c_2\nabla f(\boldsymbol{x}_n)\end{aligned}$$

The final step rearranges the update rule in terms of a "momentum" term. Inductively, if we assumed that $\boldsymbol{x}_n, \boldsymbol{x}_{n-1}$ were the optimal iterates, then by construction it follows that $\boldsymbol{x}_{n+1}$ is optimal up to $n+1$ step step-size schedule on vanilla gradient descent, thus as we take many updates $n \to \infty$ the algorithm achieves the accelerated rate.

If we were to have carried valyes for the constants throughout, we would have arrived at

$$c_3' = \left(\frac{2}{\sqrt{M} + \sqrt{m}}\right)^2, \quad c_2 = \left(\frac{\sqrt{M} - \sqrt{m}}{\sqrt{M} + \sqrt{m}}\right)^2$$

The relationship between momentum and step-size parameters c_3' and c_2 is worth its own section. Fabian Pedregosa's 2023 ICML Blog *A Hitchhiker's Guide to Momentum* may be of interest for further reading and includes derivation of the regions for which (c_3', c_2) achieve the accelerated rate.

2.4.2 Random Stepsizes via Potential Theory

Note that the derivation for optimal step-size schedules in 2.3.3 **never** made an assmption on the ordering of step-sizes. Informally, as number n steps we are optimizing grows large, taking a random ordering of these step-sizes can be approximated by sampling iid from some distribution - that is something like $\mu_n = \frac{1}{n}\sum_t \delta(\alpha_t)$ for large n. The figure 1 would suggest that the limiting distribution $\mu_n \to \mu$ is the arcsine distribution.

Alternatively, we can give theoretical justification for μ by showing that it is the optimal distribution. To begin, will express our problem using the matrix norm.

$$\begin{aligned}R_n(\omega) &= \max_f \left\| \prod \left(\boldsymbol{I} - \alpha_t(\omega)\boldsymbol{\Lambda}\right) \right\| \\ &= \max_{\lambda \in [m,M]} \left\| \prod \left|1 - \alpha_t(\omega)\lambda\right| \right\|\end{aligned}$$

Where $\alpha_t(\omega)$ are function of the random draw from μ, and the second line diagonalizes. The optimization problem we will be interested in for the random case is

$$\min_{\mu} \lim_{n\to\infty} \mathbf{E}_{\{\alpha_t\}\sim\mu^n} \left[\sqrt[n]{R_n} \right]$$

We want to study how the geometric mean converges. To do this, we can analyze the log one-step contraction rate. In general, if X_i are distributed i.i.d and $G_n := (\prod X_i)^{1/n}$ is the geometric mean, then $\mathbf{E} \log X_i = m$ implies that $G_n \to e^m$ almost surely.[7] Thus it suffices to study the problem minimizing

$$\log R(u) = \max_{\lambda\in[m,M]} \mathbf{E} \left(\log \left| 1 - \frac{\lambda}{\alpha} \right| \right)$$

since it will also minimize the geometric average of the rates. To solve the problem cleanly we can make a connection to electrostatic potential theory. To begin, we write the two problems as follow:

- Step-size Problem: copy paste from aboves but we use $\beta = \alpha^{-1}$ to minimize over the inverse-step-size distribution γ since it allows us to make a cleaner connection to potentual theory.

$$\min_{\gamma} \max_{\lambda\in[m,M]} \mathbf{E}_{\beta\sim\gamma} \log \left| 1 - \frac{\lambda}{\beta} \right|$$

- Electrostatics Problem: Meinimization of potentials on a line where the $\log 1/|\beta - \alpha|$ kernel defines the energy between particles

$$\min_{\gamma\in\mathcal{P}(m,M)} \mathbf{E}_{\alpha,\beta\sim\gamma} \log \frac{1}{|\beta - \alpha|}$$

Fundamental Theorem of Potential Theory: Characterized uniquely by an equalizing property

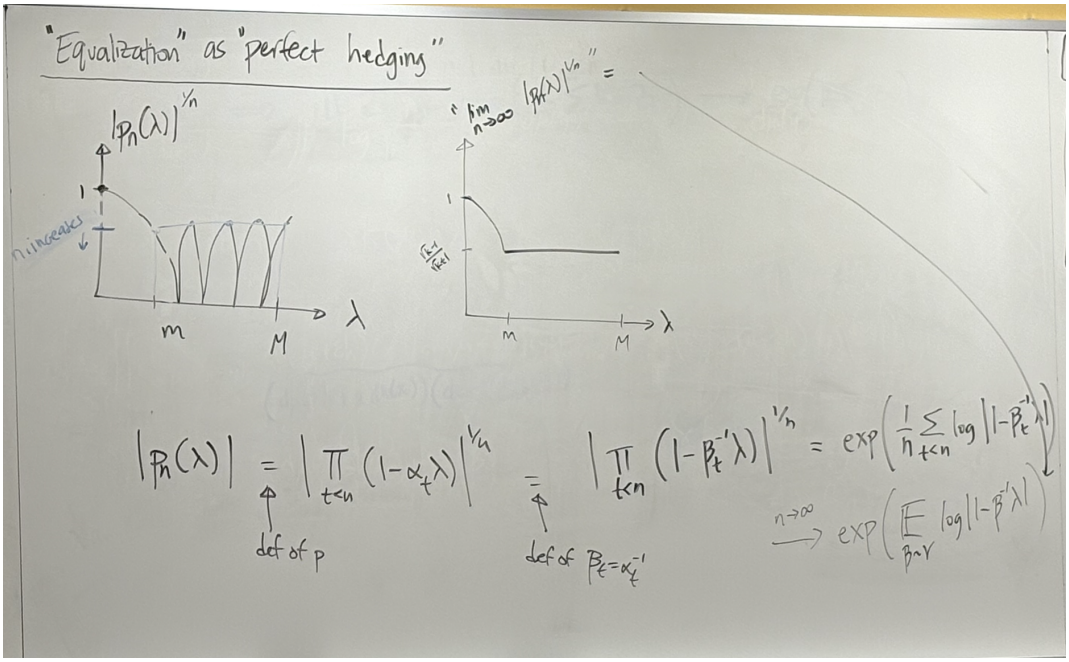
$$\lambda \mapsto \mathbf{E} \log \left(\frac{1}{|\beta - \alpha|} \right)$$

In other words, suppose we sample from the continuum of points leading to the same log potential no matter where you are on the graph. One way to think about this intuitively is that this is a "stationary" potential since (if we allow particles to move around) there is no place where the potential is higher / lower than where it is in the distribution.

Jason provided some intuitive argument for the actual conclusion that arcsine is the optimal γ but it was more complex

Connection to perfect hedging:

[7] See further discussion at this Math Stack Exchange post.



In the limit of Chebyshev step sizes what ends up happening is that we are "equally" hedged against everything. This means that **CHEBYSHEV BASED METHODS NEVER DO BETTER** than

$$\frac{\sqrt{k}-1}{\sqrt{k}+1}$$

There is some connection to Green's function that I should learn about: fundamentally, the step size optimum is given by the negative green's function evaluated on $[m, M]^c$ or something like this on the complex plane.

2.5 Understanding the best case scenarios via a two player game

- Min plays a step-size distribution
- Max plays a quadratic function

Game

$$Rate = \inf_{\gamma} \sup_{\lambda\in[m,M]} \mathbf{E}_{\beta\sim\gamma} \log\left|1-\beta^{-1}\lambda\right|$$

To make the game more symmetric we can lift the function player into distribution over λs giving

$$\inf_{\gamma\in\mathcal{P}(E)} \sup_{\rho\in\mathcal{P}(E)} \mathbf{E}_{\beta\sim\gamma,\lambda\sim\rho} \log\left|1-\beta^{-1}\lambda\right|$$

3 Nesterov's Accelertion and Curvature

This section is based on work from Su, Boyd and Candes (2015) and Shi et al. (2018).

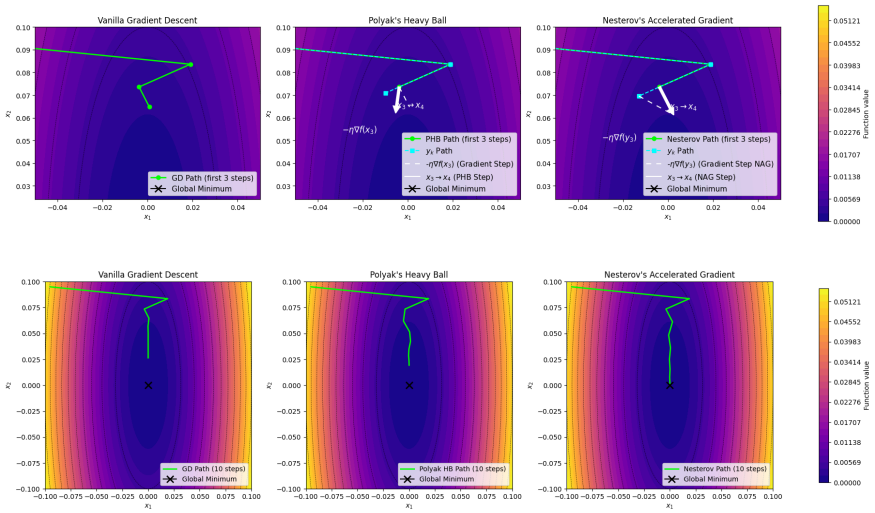


Figure 2: Top row: Illustration of Nesterov's accelerated gradient descent steps on the quadratic function $f(\boldsymbol{x}) = 10x_1^2 + x_2^2$. Bottom row: Comparison between 10 steps of vanilla gradient descent, Nesterov acceleration, and Polyak heavy ball methods on the same function

Acceleration over the class of L-smooth convex functions has proven to be a more challenging problem. In 1983, Yurii Nesterov proposed a simple modification to the gradient descent rule yielding the $O(1\sqrt{T})$ rate to match the quadratic setting. Termed, Nesterov acceleration, the update can be written as:

$$\begin{aligned} \boldsymbol{y}_k &= \boldsymbol{x}_k + \beta_k(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}) \\ \boldsymbol{x}_{k+1} &= \boldsymbol{y}_k - \eta\nabla f(\boldsymbol{y}_k) \end{aligned}$$

where β_k is a momentum parameter (which can be tuned or scheduled) and L is the smoothness parameter of f.

Visually, this update first extrapolates $\boldsymbol{x}_k$ in the direction of its last iteration $(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})$ to get $\boldsymbol{y}_k$, and then performs a gradient step from the extrapolated point $\boldsymbol{y}_k$. We can visualize the updates on quadratic $f(\boldsymbol{x}) = 10x_1^2 + x_2^2$ in Figure 2

The proof for NAG is complicated, so we are going to leave it out for now, but there is useful intuition in comparing the algorithm to Polyak's heavy ball. The updates are almost identical except in Nesterov's the gradient is computed at the point $\boldsymbol{y}_k$ as opposed to $\boldsymbol{x}_k$.

$$\begin{aligned} \text{Polyak's Update} \quad & \boldsymbol{x}_{k+1} = \boldsymbol{y}_k - \eta\nabla f(\boldsymbol{x}_k) \\ \text{Nesterov's Update} \quad & \boldsymbol{x}_{k+1} = \boldsymbol{y}_k - \eta\nabla f(\boldsymbol{y}_k) \end{aligned}$$

As in previous algorithms the challenge of adapting quadratic methods to general convex functions is that curvature may vary from one point to another. Intuitively, we might hope that $\nabla f(\boldsymbol{y}_k)$ provides more of curvature information than $\nabla f(\boldsymbol{x}_k)$.

We can start to see this by writing $\nabla f(\boldsymbol{y}_k)$ as a taylor approximation around $\nabla f(\boldsymbol{y}_k)$.

$$\begin{aligned}\nabla f(\boldsymbol{y}_k) &\approx \nabla f(\boldsymbol{x}_k) + \nabla^2 f(\boldsymbol{x}_k)(\boldsymbol{y}_k - \boldsymbol{x}_k) \\ &= \nabla f(\boldsymbol{x}_k) + \nabla^2 f(\boldsymbol{x}_k)\, \beta_k(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})\end{aligned}$$

For the quadratic case, this approximation is exact.