Optimization Analysis with Polynomials

General Background

- Polynomials have long been applied to solving linear equations (or minimizing quadratic objectives) and have many well-known properties that are useful for analysis
    - Chebyshev Polynomials (CPs) are especially common and well-understood
    - Chebyshev iterative method, conjugate gradient method, etc.

Our Goal: formulate our optimization problem as a polynomial, and leverage known properties of polynomials to characterize performance

- Chebyshev polynomials have seen a recent resurgence in theoretical machine learning research
    - PCA, decentralized algorithms, analysis of quadratic objectives, etc.

Chebyshev Polynomials

- Chebyshev polynomials (CPs) are groups/sequences of polynomial functions that are related to the sine and cosine functions
  - Can be defined recursively or with trigonometric functions
  - Widely used within approximation theory
- CPs of the first and second kind are most commonly used
  - There are also third, fourth kind, etc.
- **First Kind:**
  - *Trigonometric Definition:* $T_n(cos\theta) = \cos(n\theta)$
  - *Recursive Definition:* $T_0(x) = 1;\ \ T_1(x) = x;\ \ T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$
- **Second Kind:**
  - *Trigonometric Definition:* $U_n(\cos\theta)\sin\theta = \sin((n+1)\theta)$
  - *Recursive Definition:* $U_0(x) = 1;\ \ U_1(x) = 2x;\ \ U_{n+1}(x) = 2xU_n(x) - U_{n-1}(x)$

Chebyshev Polynomials

- Both T and U are defined within the domain [-1, 1]
  - Extrema of ± 1 or $\pm(n+1)$ are achieved on the endpoints of the domain [-1, 1]
  - Same number of extreme points occur in the domain [-1, 1]
  - Same number of zeros occur (exactly n) in the domain [-1, 1]
- Chebyshev polynomials are "orthogonal" polynomials
  - *Weighting function:*

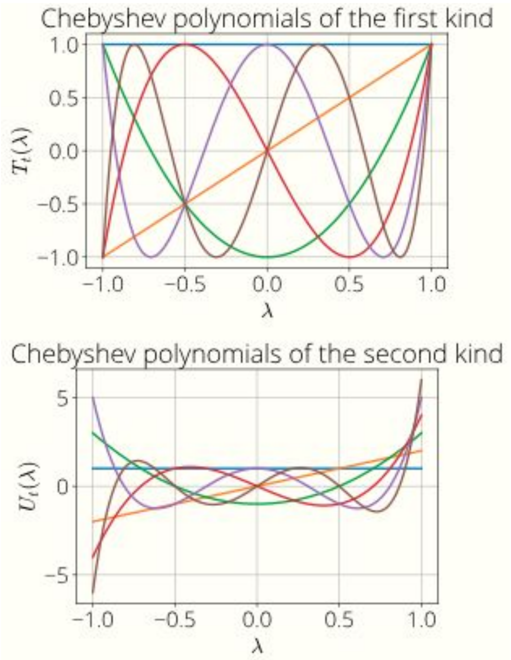
$$\mathrm{d}\mu(\lambda) = \begin{cases} \frac{1}{\pi\sqrt{1-\lambda^2}} & \text{if } \lambda \in [-1, 1] \\ 0 & \text{otherwise}. \end{cases}$$

  - *Orthogonality condition:*

$$\int T_i(\lambda) T_j(\lambda)\,\mathrm{d}\mu(\lambda) \begin{cases} > 0 \text{ if } i = j \\ = 0 \text{ otherwise}. \end{cases}$$

- CPs of the second kind can be defined w.r.t. CPs of the first kind

$$U_t(\lambda) = \int \frac{T_{t+1}(\xi) - T_{t+1}(\lambda)}{\xi - \lambda}\,\mathrm{d}\mu(\xi)\,.$$



source

Problem Setup

- How can we connect optimization algorithms to polynomials?
- We consider a strongly convex, quadratic objective:

$$\boxed{f(x) = \frac{1}{2}x^\top Hx + b^\top x}$$

  - H is a positive definite, square matrix
  - We aim to minimize this objective w.r.t. x
  - We define the largest/smallest eigenvalues of H as L and ℓ
  - $x^\star$ denotes the optimal solution to f(x)
  - *Extending analysis from quadratic objectives to general, strongly convex objectives is an open research problem*
- Gradient-based methods are usually used to provide a solution
- **Main Idea:** *Any gradient-based method for this objective can be matched with a polynomial function that determines its convergence speed*

Analysis of Gradient Descent

- We begin with the simple case of gradient descent, defined below

$$x_{t+1} = x_t - \frac{2}{L+\ell}\nabla f(x_t)$$

- Noticing that $\nabla f(x^\star) = 0$, this expression can be manipulated to yield the following

$$x_{t+1} - x^\star = \underbrace{\boxed{\left(I - \frac{2}{L+\ell}H\right)^{t+1}}}_{\text{t+1-th degree polynomial}}(x_0 - x^\star)$$

- Taking norms and invoking Cauchy-Schwarz, we arrive at the following

$$\|x_{t+1} - x^\star\| \leq \max_{\lambda\in[\ell,L]}\left|(1 - \frac{2}{L+\ell}\lambda)^{t+1}\right| \|x_0 - x^\star\|$$

- Worst-case convergence must be determined by the factor $\max_{\lambda\in[\ell,L]}\left|(1 - \frac{2}{L+\ell}\lambda)^{t+1}\right|$, so solving for this value yields our final convergence rate...

$$\boxed{\|x_t - x^\star\| \leq \left(\frac{L-\ell}{L+\ell}\right)^t \|x_0 - x^\star\|}$$

Analysis of General Gradient-Based Methods

- We consider methods that combine the current iterate, gradient, and different of previous iterates

$$x_{t+1} = x_t + c_t^{(t)} \nabla f(x_t) + \sum_{i=0}^{t-1} c_i^{(t)} (x_{i+1} - x_i)$$

  - All $c_j^{(t)}$ values are scalars (notice that not specifying these values makes the polynomial very generic)
  - Recovers many gradient-based methods (e.g., Polyak method), excluding those with matrix preconditioners
- We define the following polynomial in a residual fashion
  - 0-th degree polynomial is 1 by construction
  - This must be true to satisfy $\|x_0 - x^\star\| \leq \|x_0 - x^\star\|$
  - All residual polynomials satisfy $P_t(0) = 1$

$$\boxed{\begin{cases} P_{t+1}(\lambda) = (1 + c_t^{(t)}\lambda) P_t(\lambda) + \sum_{i=0}^{t-1} c_i^{(t)} (P_{i+1}(\lambda) - P_i(\lambda)) \\ P_0(\lambda) = 1 \end{cases}}$$

- Using this residual polynomial, we can show the following for iteration t of any gradient-based method

$$x_t - x^\star = P_t(H)(x_0 - x^\star)$$

Current error depends on two terms: polynomial and initialization

- We can replace the above matrix polynomial with a scalar bound by replacing H with its eigendecomposition

$$\|\boldsymbol{x}_t - \boldsymbol{x}^\star\| \leq \underbrace{\max_{\lambda \in [\ell, L]}}_{\text{conditioning}} \underbrace{|P_t(\lambda)|}_{\text{algorithm}} \underbrace{\|\boldsymbol{x}_0 - \boldsymbol{x}^\star\|}_{\text{initialization}}$$

source

Going in reverse...

$$\|\boldsymbol{x}_t - \boldsymbol{x}^\star\| \le \underbrace{\max_{\lambda \in [\ell, L]}}_{\text{conditioning}} \underbrace{|P_t(\lambda)|}_{\text{algorithm}} \underbrace{\|\boldsymbol{x}_0 - \boldsymbol{x}^\star\|}_{\text{initialization}}$$

source

- What residual polynomial will give us the best convergence rate? Can we find the following?
  $$\arg\min_{P} \max_{\lambda \in [\ell, L]} |P(\lambda)|$$
  - If we find this “optimal” residual polynomial, we can reverse-engineer it to derive a better optimization method!
  - This minimization is over all t-degree polynomials such that $P_t(0) = 1$ (makes the problem non-trivial)
- Interestingly, numerical analysis tells us that the above optimization problem is solved by a modified CP
- Specifically, the following shifted CP achieves the smallest value within the domain $[\ell, L]$

$$P_t(\lambda) = \frac{T_t(\sigma(\lambda))}{T_t(\sigma(0))}, \text{ where } \sigma(\lambda) = \frac{L+\ell}{L-\ell} - \frac{2}{L-\ell}\lambda$$

  - $\sigma(\cdot)$ simply transforms the domain $[\ell, L]$ to $[-1, 1]$, which is more appropriate for CPs
- We can then plug the above CP into the residual polynomial recurrence and solve for the appropriate coefficients within the gradient-based optimization method -- this yields the **Chebyshev Iterative Method**

Chebyshev Iterative Method

Input: starting guess $\boldsymbol{x}_0$, $\rho = \frac{L-\ell}{L+\ell}$

$\boldsymbol{x}_1 = \boldsymbol{x}_0 - \frac{2}{L+\ell}\nabla f(\boldsymbol{x}_0)$

$\omega_1 = 2$

For $t = 1, 2, \ldots$ do

$$\omega_{t+1} = (1 - \tfrac{\rho^2}{4}\omega_t)^{-1}$$

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + (1 - \omega_{t+1})(\boldsymbol{x}_{t-1} - \boldsymbol{x}_t) - \omega_{t+1}\tfrac{2}{L+\ell}\nabla f(\boldsymbol{x}_t)$$

Convergence Rate →

$$\underbrace{\|x_t - x^\star\| \le 2\left(\frac{\sqrt{L} - \sqrt{\ell}}{\sqrt{L} + \sqrt{\ell}}\right)^t \|x_0 - x^\star\|}_{\text{Better than gradient descent!}}$$

Analysis of Gradient Descent with Momentum

- Gradient Descent with momentum is an optimization method with the following form

$$x_1 = x_0 - \frac{\eta}{1+m}\nabla f(x_0)$$

$$\boxed{x_{t+1} = x_t + m(x_t - x_{t-1}) - \eta\nabla f(x_t)}$$

  - Here, m and η are the momentum and learning rate hyperparameters, respectively
- The momentum method shown above is just a specialized case of the general, gradient-based method. So, we can again express the error at a given iteration with residual polynomials

$$x_t - x^\star = P_t(H)(x_0 - x^\star)$$

- The definition of the residual polynomial is expressed as follows for the momentum method

$$P_{t+1}(\lambda) = (1 + m - \eta\lambda)P_t(\lambda) - mP_{t-1}(\lambda)$$

$$P_0(\lambda) = 1; \quad P_1(\lambda) = 1 - \frac{\eta}{1+m}\lambda$$

This recurrence is simple in comparison to the general case,
which includes a sum over all previous residual polynomials

Analysis of Gradient Descent with Momentum

- **Residual Polynomial of the Momentum Method:** a combination of first/second kind CPs
  - Can be shown using induction and by leveraging the recursive definition of CPs

$$\boxed{P_t(\lambda) = m^{\frac{t}{2}}\left(\frac{2m}{1+m}T_t(\sigma(\lambda)) - \frac{m-1}{1+m}U_t(\sigma(\lambda))\right)}$$

$$\text{where} \quad \sigma(\lambda) = \frac{1}{2\sqrt{m}}(1+m-\eta\lambda)$$

- **We can now leverage this residual polynomial to derive a convergence rate!** Plugging in the CP polynomial definition above expresses the derivation of a convergence rate as the analysis of CPs

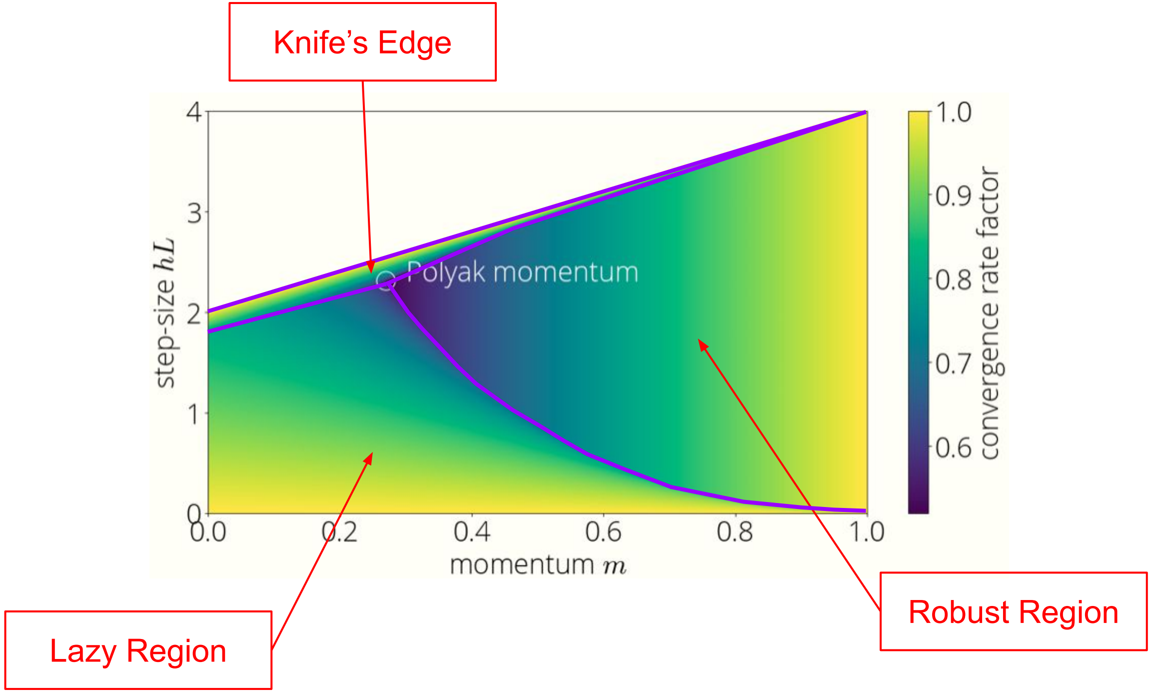
$$\begin{aligned} x_t - x^\star &= P_t(H)(x_0 - x^\star) \\ \Rightarrow \|x_t - x^\star\| &\leq \max_{\lambda\in[\ell,L]} |P_t(\lambda)| \; \|x_0 - x^\star\| \end{aligned} \xrightarrow{\text{CP formulation of } P_t(\lambda)} \max_{\lambda\in[\ell,L]} |P_t(\lambda)| \leq m^{\frac{t}{2}}\left(\frac{2m}{1+m}\max_{\lambda\in[\ell,L]} |T_t(\sigma(\lambda))| + \frac{1-m}{1+m}\max_{\lambda\in[\ell,L]} |U_t(\sigma(\lambda))|\right)$$

Analysis of Gradient Descent with Momentum

$$\max_{\lambda\in[\ell,L]} |P_t(\lambda)| \leq m^{\frac{t}{2}} \left(\frac{2m}{1+m} \underbrace{\max_{\lambda\in[\ell,L]} |T_t(\sigma(\lambda))|}_{\text{unknowns}} + \frac{1-m}{1+m} \underbrace{\max_{\lambda\in[\ell,L]} |U_t(\sigma(\lambda))|}_{\text{unknowns}} \right)$$

- Everything is known except for the bound on CPs, and we know how to derive bounds for CPs!
- **Simplest Case:** $\sigma(\lambda) \in [-1,1] \Rightarrow \max_{\lambda\in[\ell,L]} |\sigma(\lambda)| \leq 1 \Rightarrow \boxed{\frac{(1-\sqrt{m})^2}{\eta} \leq \ell \leq L \leq \frac{(1+\sqrt{m})^2}{\eta}}$ (use distinction of cases and definition of sigma)
  - CPs are bounded by ± 1 and $\pm t+1$
  - m and η that satisfy the above inequality form a "robust region" of convergence for the momentum method
- Within the robust region, we achieve the following convergence rate: $\boxed{\max_{\lambda\in[\ell,L]} |P_t(\lambda)| \leq m^{\frac{t}{2}} \left(1 + \frac{1-m}{1+m} t\right)}$
  - Convergence only depends on the momentum hyperparameter (not learning rate) in the robust region
  - This insensitivity to step size within the robust region is leveraged by the yellowfin momentum tuner
- Convergence can also be analyzed outside of the robust region (i.e., there are other convergent hyperparameter settings)

Convergent Regions of Gradient Descent with Momentum



Analysis of Polyak Momentum

- Polyak Momentum follows the momentum method recurrence with the following momentum and learning rate

$$m = \left(\frac{\sqrt{L}-\sqrt{\ell}}{\sqrt{L}+\sqrt{\ell}}\right)^2 \qquad \eta = \left(\frac{2}{\sqrt{L}+\sqrt{\ell}}\right)^2$$

- **Minimizing the convergence rate of the momentum method in the robust region yields Polyak Momentum!**
  - Recall the convergence rate for momentum method in the robust region: $\max_{\lambda \in [\ell, L]} |P_t(\lambda)| \le m^{\frac{t}{2}} \left(1 + \frac{1-m}{1+m} t\right)$
  - Also, recall the hyperparameter bounds that define the robust region: $\frac{(1-\sqrt{m})^2}{\eta} \le \ell \le L \le \frac{(1+\sqrt{m})^2}{\eta}$
  - The best convergence rate occurs with the smallest possible momentum in the robust region
  - By turning the inequalities above into equalities and solving for the hyperparameters, we recover the settings of Polyak momentum:

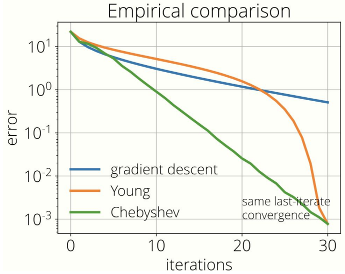
$$m = \left(\frac{\sqrt{L}-\sqrt{\ell}}{\sqrt{L}+\sqrt{\ell}}\right)^2 \qquad \eta = \left(\frac{2}{\sqrt{L}+\sqrt{\ell}}\right)^2$$

- Because the above hyperparameter lie in the robust region, we can easily leverage previous results to obtain a convergence rate for Polyak Momentum:

$$\max_{\lambda \in [\ell, L]} |P_t(\lambda)| \le \boxed{\left(\frac{\sqrt{L}-\sqrt{\ell}}{\sqrt{L}+\sqrt{\ell}}\right)^t} \left(1 + \frac{2\sqrt{L\ell}}{L+\ell} t\right)$$

Square root factor improvement in comparison to Gradient Descent

Acceleration without Momentum



- We again consider a strongly convex, quadratic objective
- Before, we achieved an accelerated rate by solving for the optimal residual polynomial, leading to the Chebyshev iterative method
- **Young's Method:** use a curated learning rate schedule to achieve accelerated rate without any momentum term
  - To match the optimal residual polynomial (and therefore achieve the accelerated rate), our optimization method's residual polynomial must have the same roots (this is sufficient due to the $P_t(0) = 1$ constraint on residual polynomials)
  - Consider a learning rate schedule with a specific learning rate at every iteration

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

  - Using this formulation, we can derive the following:

$$x_{t+1} - x^\star = (I - \eta_t H) \dots (I - \eta_0 H)(x_0 - x^\star) \xrightarrow{\text{Residual Polynomial}} \boxed{P_t(\lambda) = (1 - \eta_t \lambda) \dots (1 - \eta_0 \lambda)}$$

  - Roots of $P_t(\lambda)$ are in factored form and occur at $1/\eta_t, \ \dots \ , 1/\eta_0$
  - Learning rates can be manipulated to match roots of the optimal residual polynomial (roots of a CP have closed form)
  - **Result:** a learning rate schedule that yields comparable acceleration to the Chebyshev iterative method
- The results for Young's method hold no matter the order of step sizes taken during the optimization process. Additionally, the only guarantee is on the performance of the t-th iterate, and the method might perform arbitrarily bad until this iteration.
  - Note that the resulting learning rate schedule is completely dependent on the value of t!

Applications in Current Research

- Fractal learning rates for improved stability
- Guarantees for accelerated power iteration
- Average-case convergence guarantees for gradient descent
- Yellowfin momentum tuner
- Theoretical analysis of cyclical learning rate schedules