CS794/CO673: Optimization for Data Science

Lec 01: Linear Systems

Yaoliang Yu

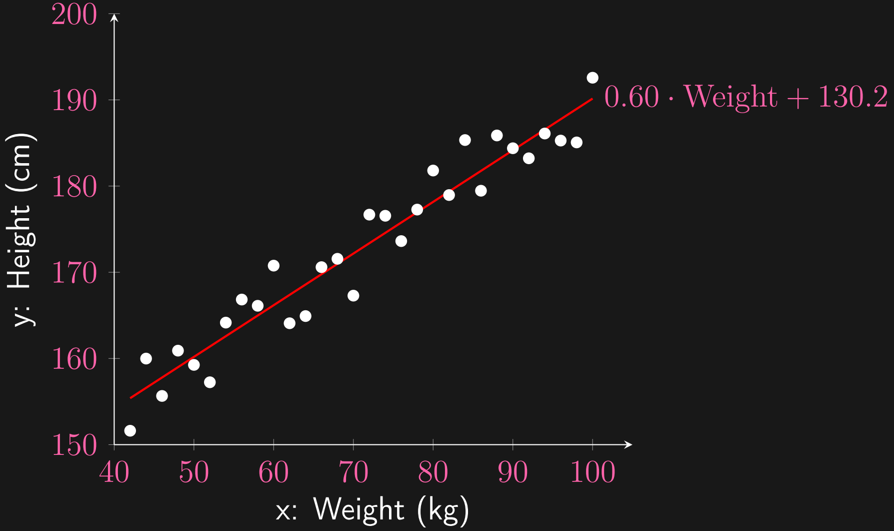
UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS
DAVID R. CHERITON SCHOOL
OF COMPUTER SCIENCE

September 9, 2022

Problem

- $A \in \mathbb{S}^d_{++}$: symmetric and positive definite
  - all eigenvalues of A are real and positive
  - unique solution $\mathbf{w}_\star = A^{-1} \cdot \mathbf{b}$
- "One-line" code: A\b
- Twist: only matrix-vector product allowed, e.g. $A\mathbf{w}$ (and $A^\top \mathbf{w}$)
- Progress measure:
  - $\|\mathbf{w} - \mathbf{w}_\star\|_2$, not computable hence only of theoretical value
  - $\|A\mathbf{w} - \mathbf{b}\|_2 = \|A\mathbf{w} - A\mathbf{w}_\star\|_2$, computable

Linear Regression



Formalizing Linear Regression

- Affine function: $f(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle + b$, where $\langle \mathbf{x}, \mathbf{w} \rangle := \sum_j x_j w_j$
- Want: $f(\mathbf{x}_i) \approx y_i$, by tuning $\mathbf{w}$ and b
- Least squares (dates back to Gauss):

$$\min_{\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}} \sum_i (f(\mathbf{x}_i) - y_i)^2$$

- In matrix form:

$$\min_{\mathsf{w} \in \mathbb{R}^p} \|\mathsf{X}\mathsf{w} - \mathbf{y}\|_2^2, \quad \text{where} \quad \mathsf{w} = \begin{pmatrix} \mathbf{w} \\ b \end{pmatrix}, \mathsf{X} = \begin{bmatrix} \mathbf{x}_1 & \cdots & \mathbf{x}_n \\ 1 & \cdots & 1 \end{bmatrix}^\top$$

- Normal equation: $\underbrace{\mathsf{X}^\top \mathsf{X}}_{A} \cdot \mathsf{w} = \underbrace{\mathsf{X}^\top \mathbf{y}}_{\mathbf{b}}$

Ridge Regression

- Is $|\langle \mathbf{x}_i, \mathbf{w}\rangle + b - y_i|$ the distance from $(\mathbf{x}_i, y_i)$ to the line $y = \langle \mathbf{x}, \mathbf{w}\rangle + b$?

- Orthogonal regression:

$$\lambda_\star := \min_{\mathbf{w}\in\mathbb{R}^p} \frac{\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2}{\|\mathbf{w}\|_2^2} \quad \equiv \quad \min_{\mathbf{w}\in\mathbb{R}^p} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 - \lambda_\star\|\mathbf{w}\|_2^2$$

- Ridge regression:

$$\min_{\mathbf{w}\in\mathbb{R}^p} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$$

- $A = \mathbf{X}^\top\mathbf{X} + \lambda\mathbb{I}$: symmetric and positive definite for $\lambda > 0$

A. E. Hoerl and R. W. Kennard (1970). "Ridge regression: biased estimation for nonorthogonal problems". *Technometrics*, vol. 12, no. 1, pp. 55–67.

Richardson Extrapolation

```
Input: $\mathbf{w}_0 \in \mathbb{R}^d$, $A \in \mathbb{R}^{d\times d}$, $\mathbf{b} \in \mathbb{R}^d$
for $t = 0, 1, \ldots$ do
    $\mathbf{g}_t \leftarrow A\mathbf{w}_t - \mathbf{b}$                    // “gradient”
    $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \mathbf{g}_t$              // $\eta_t$ is the step size
```

- Repeatedly subtract a multiple of “gradient” $\mathbf{g}_t := A\mathbf{w}_t - \mathbf{b}$

$$A\mathbf{w}_{t+1} - \mathbf{b} = A[\mathbf{w}_t - \eta_t(A\mathbf{w}_t - \mathbf{b})] - \mathbf{b} = (I - \eta_t A)(A\mathbf{w}_t - \mathbf{b})$$

$$= \underbrace{\prod_{\tau=0}^{t}(I - \eta_\tau A)\cdot}_{\mathscr{P}_{t+1}(A)} \quad \underbrace{(A\mathbf{w}_0 - \mathbf{b})}_{\text{initial gradient}}$$

- In other words, $\mathbf{g}_t = \mathscr{P}_t(A)\cdot \mathbf{g}_0$ with $\mathscr{P}_0 \equiv \mathbb{I}$.

Polynomials

- Polynomial of degree k defined for a real scalar λ:

$$\mathscr{P}_k(\lambda) = p_0 + p_1\lambda + p_2\lambda^2 + \cdots + p_k\lambda^k = \sum_{l=0}^{k} p_l\lambda^l$$

- Extend to a symmetric matrix A:

$$A = \sum\nolimits_j \lambda_j \mathbf{u}_j\mathbf{u}_j^\top \implies \mathscr{P}_t(A) = \sum\nolimits_j \mathscr{P}_t(\lambda_j)\mathbf{u}_j\mathbf{u}_j^\top$$

  - i.e., apply the polynomial to eigenvalues while fix eigenvectors

- Can extend to smooth functions and asymmetric matrices

Constant Step Size

$$A\mathbf{w}_t - \mathbf{b} = (I - \eta A)^t \cdot (A\mathbf{w}_0 - \mathbf{b})$$

$$\begin{aligned} \|A\mathbf{w}_t - \mathbf{b}\|_2 &\le \|(I - \eta A)^t\|_{\mathrm{sp}} \cdot \|A\mathbf{w}_0 - \mathbf{b}\|_2 \\ &= \|I - \eta A\|_{\mathrm{sp}}^t \cdot \|A\mathbf{w}_0 - \mathbf{b}\|_2 \end{aligned}$$

- $\|A\mathbf{w}_0 - \mathbf{b}\|_2$: initial error, controlled by $\mathbf{w}_0$
- Assume $\mathrm{spectrum}(A) \in [\sigma, \mathsf{L}]$:

$$\|I - \eta A\|_{\mathrm{sp}} = \max_{\lambda \in \mathrm{spectrum}(A)} |1 - \eta\lambda| \le |1 - \eta\sigma| \vee |1 - \eta\mathsf{L}|$$

- Minimizing RHS $\implies \eta_* = \frac{2}{\mathsf{L}+\sigma}$
- Plugging back obtain ($\kappa := \mathsf{L}/\sigma$ is the condition number of A):

$$\|A\mathbf{w}_t - \mathbf{b}\|_2 \le \left(\tfrac{\mathsf{L}-\sigma}{\mathsf{L}+\sigma}\right)^t \cdot \|A\mathbf{w}_0 - \mathbf{b}\|_2 = \left(\tfrac{\kappa-1}{\kappa+1}\right)^t \cdot \|A\mathbf{w}_0 - \mathbf{b}\|_2$$

- Linear convergence; slower for larger κ

Dynamic Step Size

$$A\mathbf{w}_t - \mathbf{b} = \underbrace{\prod_{\tau=0}^{t}(I - \eta_\tau A)}_{\mathscr{P}_t(A)} \cdot (A\mathbf{w}_0 - \mathbf{b})$$

$$\|A\mathbf{w}_t - \mathbf{b}\|_2 \leq \|\mathscr{P}_t(A)\|_{\mathrm{sp}} \cdot \|A\mathbf{w}_0 - \mathbf{b}\|_2$$

- Can no longer find optimal η_t in closed-form
- Possible to find near-optimal step size η_t
- May have to fix `maxiter` beforehand

D. Young (1954). "Iterative Methods for Solving Partial Difference Equations of Elliptic Type". *Transactions of the American Mathematical Society*, vol. 76, no. 1, pp. 92–111.

Can We Do Better?

$$\min_{\mathscr{P}_t} \max_{A} \|\mathscr{P}_t(A)\|_{\mathrm{sp}}$$

- $\mathscr{P}_t$ any polynomial of degree t and $\mathscr{P}_t(0) = 1$
- A any matrix with spectrum in $[\sigma, \mathsf{L}]$
- Minimax analysis
- Be careful about the ordering:

$$\neq \max_{A} \min_{\mathscr{P}_t} \|\mathscr{P}_t(A)\|_{\mathrm{sp}}$$

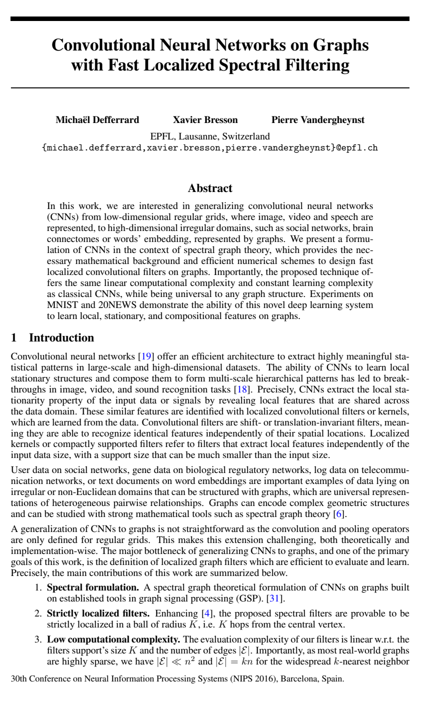
Chebyshev Polynomial

$$\mathscr{T}_0(\lambda) = 1, \quad \mathscr{T}_1(\lambda) = \lambda, \quad \mathscr{T}_{k+1}(\lambda) = 2\lambda \cdot \mathscr{T}_k(\lambda) - \mathscr{T}_{k-1}(\lambda),$$

or directly as:

$$\mathscr{T}_k(\lambda) = \begin{cases} \cos(k \cdot \arccos \lambda), & \text{if } |\lambda| \leq 1 \\ \cosh(k \cdot \operatorname{arccosh} \lambda), & \text{if } \lambda > 1 \\ (-1)^k \cosh\big(k \cdot \operatorname{arccosh}(-\lambda)\big), & \text{if } \lambda < -1 \end{cases}.$$

$|\mathscr{T}_k(\lambda)| \leq 1$, with equality attained iff $\lambda = \cos \frac{l}{k}\pi, \; l = 0, 1, \ldots, k$



Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering

Michaël Defferrard **Xavier Bresson** **Pierre Vandergheynst**

EPFL, Lausanne, Switzerland
{michael.defferrard,xavier.bresson,pierre.vandergheynst}@epfl.ch

Abstract

In this work, we are interested in generalizing convolutional neural networks (CNNs) from low-dimensional regular grids, where image, video and speech are represented, to high-dimensional irregular domains, such as social networks, brain connectomes or words' embedding, represented by graphs. We present a formulation of CNNs in the context of spectral graph theory, which provides the necessary mathematical background and efficient numerical schemes to design fast localized convolutional filters on graphs. Importantly, the proposed technique offers the same linear computational complexity and constant learning complexity as classical CNNs, while being universal to any graph structure. Experiments on MNIST and 20NEWS demonstrate the ability of this novel deep learning system to learn local, stationary, and compositional features on graphs.

1 Introduction

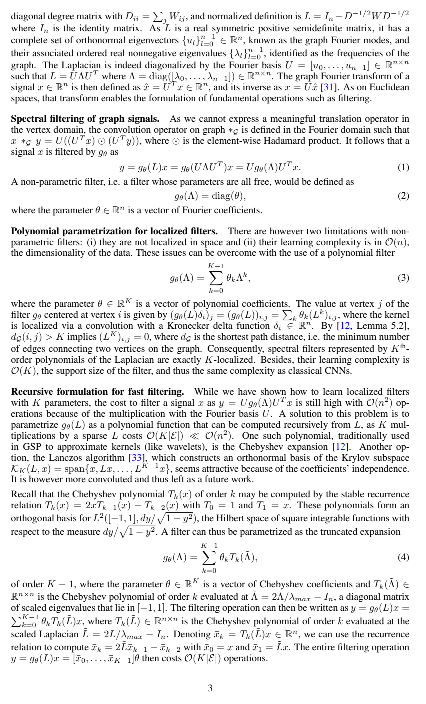
Convolutional neural networks [19] offer an efficient architecture to extract highly meaningful statistical patterns in large-scale and high-dimensional datasets. The ability of CNNs to learn local stationary structures and compose them to form multi-scale hierarchical patterns has led to breakthroughs in image, video, and sound recognition tasks [18]. Precisely, CNNs extract the local stationarity property of the input data or signals by revealing local features that are shared across the data domain. These similar features are identified with localized convolutional filters or kernels, which are learned from the data. Convolutional filters are shift- or translation-invariant filters, meaning they are able to recognize identical features independently of their spatial locations. Localized kernels or compactly supported filters refer to filters that extract local features independently of the input data size, with a support size that can be much smaller than the input size.

User data on social networks, gene data on biological regulatory networks, log data on telecommunication networks, or text documents on word embeddings are important examples of data lying on irregular or non-Euclidean domains that can be structured with graphs, which are universal representations of heterogeneous pairwise relationships. Graphs can encode complex geometric structures and can be studied with strong mathematical tools such as spectral graph theory [6].

A generalization of CNNs to graphs is not straightforward as the convolution and pooling operators are only defined for regular grids. This makes this extension challenging, both theoretically and implementation-wise. The major bottleneck of generalizing CNNs to graphs, and one of the primary goals of this work, is the definition of localized graph filters which are efficient to evaluate and learn. Precisely, the main contributions of this work are summarized below.

1. **Spectral formulation.** A spectral graph theoretical formulation of CNNs on graphs built on established tools in graph signal processing (GSP). [31].
2. **Strictly localized filters.** Enhancing [4], the proposed spectral filters are provable to be strictly localized in a ball of radius K, i.e. K hops from the central vertex.
3. **Low computational complexity.** The evaluation complexity of our filters is linear w.r.t. the filters support's size K and the number of edges $|\mathcal{E}|$. Importantly, as most real-world graphs are highly sparse, we have $|\mathcal{E}| \ll n^2$ and $|\mathcal{E}| = kn$ for the widespread k-nearest neighbor

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.



diagonal degree matrix with $D_{ii} = \sum_j W_{ij}$, and normalized definition is $L = I_n - D^{-1/2} W D^{-1/2}$ where I_n is the identity matrix. As L is a real symmetric positive semidefinite matrix, it has a complete set of orthonormal eigenvectors $\{u_l\}_{l=0}^{n-1} \in \mathbb{R}^n$, known as the graph Fourier modes, and their associated ordered real nonnegative eigenvalues $\{\lambda_l\}_{l=0}^{n-1}$, identified as the frequencies of the graph. The Laplacian is indeed diagonalized by the Fourier basis $U = [u_0, \dots, u_{n-1}] \in \mathbb{R}^{n \times n}$ such that $L = U \Lambda U^T$ where $\Lambda = \mathrm{diag}([\lambda_0, \dots, \lambda_{n-1}]) \in \mathbb{R}^{n \times n}$. The graph Fourier transform of a signal $x \in \mathbb{R}^n$ is then defined as $\hat{x} = U^T x \in \mathbb{R}^n$, and its inverse as $x = U \hat{x}$ [31]. As on Euclidean spaces, that transform enables the formulation of fundamental operations such as filtering.

Spectral filtering of graph signals. As we cannot express a meaningful translation operator in the vertex domain, the convolution operator on graph $*_\mathcal{G}$ is defined in the Fourier domain such that $x *_\mathcal{G} y = U((U^T x) \odot (U^T y))$, where $\odot$ is the element-wise Hadamard product. It follows that a signal x is filtered by g_θ as

$$y = g_\theta(L) x = g_\theta(U \Lambda U^T) x = U g_\theta(\Lambda) U^T x. \tag{1}$$

A non-parametric filter, i.e. a filter whose parameters are all free, would be defined as

$$g_\theta(\Lambda) = \mathrm{diag}(\theta), \tag{2}$$

where the parameter $\theta \in \mathbb{R}^n$ is a vector of Fourier coefficients.

Polynomial parametrization for localized filters. There are however two limitations with non-parametric filters: (i) they are not localized in space and (ii) their learning complexity is in $\mathcal{O}(n)$, the dimensionality of the data. These issues can be overcome with the use of a polynomial filter

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k, \tag{3}$$

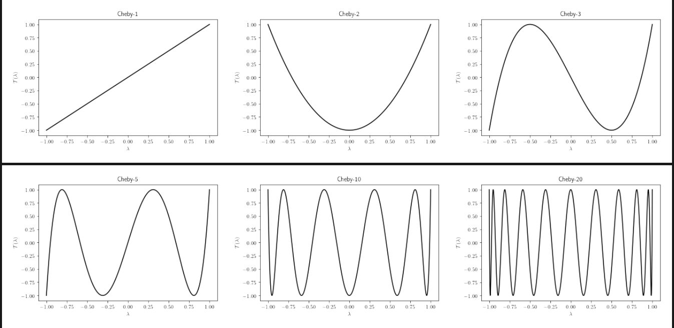
where the parameter $\theta \in \mathbb{R}^K$ is a vector of polynomial coefficients. The value at vertex j of the filter g_θ centered at vertex i is given by $(g_\theta(L)\delta_i)_j = (g_\theta(L))_{i,j} = \sum_k \theta_k (L^k)_{i,j}$, where the kernel is localized via a convolution with a Kronecker delta function $\delta_i \in \mathbb{R}^n$. By [12, Lemma 5.2], $d_\mathcal{G}(i,j) > K$ implies $(L^K)_{i,j} = 0$, where $d_\mathcal{G}$ is the shortest path distance, i.e. the minimum number of edges connecting two vertices on the graph. Consequently, spectral filters represented by K^{th}-order polynomials of the Laplacian are exactly K-localized. Besides, their learning complexity is $\mathcal{O}(K)$, the support size of the filter, and thus the same complexity as classical CNNs.

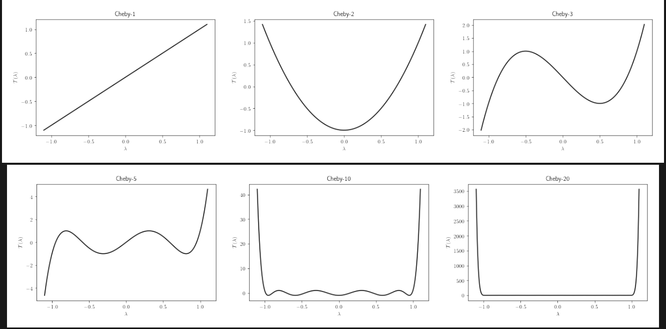
Recursive formulation for fast filtering. While we have shown how to learn localized filters with K parameters, the cost to filter a signal x as $y = U g_\theta(\Lambda) U^T x$ is still high with $\mathcal{O}(n^2)$ operations because of the multiplication with the Fourier basis U. A solution to this problem is to parametrize $g_\theta(L)$ as a polynomial function that can be computed recursively from L, as K multiplications by a sparse L costs $\mathcal{O}(K|\mathcal{E}|) \ll \mathcal{O}(n^2)$. One such polynomial, traditionally used in GSP to approximate kernels (like wavelets), is the Chebyshev expansion [12]. Another option, the Lanczos algorithm [33], which constructs an orthonormal basis of the Krylov subspace $\mathcal{K}_K(L, x) = \mathrm{span}\{x, Lx, \dots, L^{K-1}x\}$, seems attractive because of the coefficients' independence. It is however more convoluted and thus left as a future work.

Recall that the Chebyshev polynomial $T_k(x)$ of order k may be computed by the stable recurrence relation $T_k(x) = 2x T_{k-1}(x) - T_{k-2}(x)$ with $T_0 = 1$ and $T_1 = x$. These polynomials form an orthogonal basis for $L^2([-1,1], dy/\sqrt{1-y^2})$, the Hilbert space of square integrable functions with respect to the measure $dy/\sqrt{1-y^2}$. A filter can thus be parametrized as the truncated expansion

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda}), \tag{4}$$

of order $K-1$, where the parameter $\theta \in \mathbb{R}^K$ is a vector of Chebyshev coefficients and $T_k(\tilde{\Lambda}) \in \mathbb{R}^{n \times n}$ is the Chebyshev polynomial of order k evaluated at $\tilde{\Lambda} = 2\Lambda/\lambda_{max} - I_n$, a diagonal matrix of scaled eigenvalues that lie in $[-1,1]$. The filtering operation can then be written as $y = g_\theta(L)x = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L}) x$, where $T_k(\tilde{L}) \in \mathbb{R}^{n \times n}$ is the Chebyshev polynomial of order k evaluated at the scaled Laplacian $\tilde{L} = 2L/\lambda_{max} - I_n$. Denoting $\bar{x}_k = T_k(\tilde{L})x \in \mathbb{R}^n$, we can use the recurrence relation to compute $\bar{x}_k = 2\tilde{L}\bar{x}_{k-1} - \bar{x}_{k-2}$ with $\bar{x}_0 = x$ and $\bar{x}_1 = \tilde{L}x$. The entire filtering operation $y = g_\theta(L)x = [\bar{x}_0, \dots, \bar{x}_{K-1}]\theta$ then costs $\mathcal{O}(K|\mathcal{E}|)$ operations.


Cheby-1
Cheby-2
Cheby-3
Cheby-5
Cheby-10
Cheby-20
$T(\lambda)$
λ
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


Cheby-1
Cheby-2
Cheby-3
Cheby-5
Cheby-10
Cheby-20
$T(\lambda)$
λ
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
−1.5
−2.0
4
2
0
−2
−4
40
30
20
10
3500
3000
2500
2000
1500
1000
500

Translation and Scaling

$$\mathscr{C}_{t+1}(\lambda) = \frac{\mathscr{T}_{t+1}(\mathscr{S}(\lambda))}{\mathscr{T}_{t+1}(\mathscr{S}(0))}, \quad \text{where} \quad \mathscr{S}(\lambda) := \frac{2\lambda}{\mathsf{L}-\sigma} - \frac{\mathsf{L}+\sigma}{\mathsf{L}-\sigma}$$

$$\mathscr{C}_{t+1}(\lambda) = \tfrac{\mathscr{S}(\lambda)}{\mathscr{S}(0)} \cdot \gamma_t \cdot \mathscr{C}_t(\lambda) - (\gamma_t - 1) \cdot \mathscr{C}_{t-1}(\lambda), \quad \text{where}$$

$$\gamma_t := 2\mathscr{S}(0)\frac{\mathscr{T}_t(\mathscr{S}(0))}{\mathscr{T}_{t+1}(\mathscr{S}(0))} = \frac{4\mathscr{S}^2(0)}{4\mathscr{S}^2(0) - \gamma_{t-1}}$$

- $\mathscr{C}_0(\lambda) = 1, \mathscr{C}_1(\lambda) = \frac{\mathscr{S}(\lambda)}{\mathscr{S}(0)}, \gamma_0 = 2$
- $\gamma_t \downarrow \underline{\gamma} := \frac{2(\kappa+1)}{(\sqrt{\kappa}+1)^2}$, recall $\kappa = \sigma/\mathsf{L}$

$$\mathscr{C}_{t+1}(\lambda) = \frac{\mathscr{S}(\lambda)}{\mathscr{S}(0)} \cdot \gamma_t \cdot \mathscr{C}_t(\lambda) - (\gamma_t - 1) \cdot \mathscr{C}_{t-1}(\lambda)$$

$$\begin{aligned}
A\mathbf{w}_{t+1} - \mathbf{b} &= \mathscr{C}_{t+1}(A) \cdot (A\mathbf{w}_0 - \mathbf{b}) \\
&= \left[\frac{\mathscr{S}(A)}{\mathscr{S}(0)} \cdot \gamma_t \cdot \mathscr{C}_t(A) - (\gamma_t - 1) \cdot \mathscr{C}_{t-1}(A)\right] \cdot (A\mathbf{w}_0 - \mathbf{b}) \\
&= [I - \tfrac{2A}{\mathsf{L}+\sigma}]\gamma_t \cdot \mathscr{C}_t(A)(A\mathbf{w}_0 - \mathbf{b}) - (\gamma_t - 1) \cdot \mathscr{C}_{t-1}(A)(A\mathbf{w}_0 - \mathbf{b}) \\
&= [I - \eta_* A]\gamma_t \cdot (A\mathbf{w}_t - \mathbf{b}) - (\gamma_t - 1) \cdot (A\mathbf{w}_{t-1} - \mathbf{b}) \\
&= (A\mathbf{w}_t - \mathbf{b}) - \eta_* \gamma_t \cdot A(A\mathbf{w}_t - \mathbf{b}) + (\gamma_t - 1) \cdot (A\mathbf{w}_t - A\mathbf{w}_{t-1})
\end{aligned}$$

$$\mathbf{w}_{t+1} = \underbrace{\mathbf{w}_t - \gamma_t \eta_t (A\mathbf{w}_t - \mathbf{b})}_{\text{Richardson}} + \overbrace{(\gamma_t - 1)}^{\text{positive}} \underbrace{(\mathbf{w}_t - \mathbf{w}_{t-1})}_{\text{momentum}}$$

Chebyshev method

```
Input: $\mathbf{w}_0, \mathbf{b} \in \mathbb{R}^d$, $A \in \mathbb{S}^d_{++} \in [\sigma, \mathsf{L}]$, $\gamma_0 = 2$, $\kappa = \frac{\mathsf{L}}{\sigma}$
$\mathbf{g}_0 \leftarrow A\mathbf{w}_0 - \mathbf{b}$
$\mathbf{w}_1 \leftarrow \mathbf{w}_0 - \eta_0 \mathbf{g}_0$                                  // $\eta_t \equiv \frac{2}{\mathsf{L}+\sigma}$
for $t = 1, 2, \ldots$ do
    $\mathbf{g}_t \leftarrow A\mathbf{w}_t - \mathbf{b}$                                    // gradient
    $\gamma_t \leftarrow \frac{4(\kappa+1)^2}{4(\kappa+1)^2 - (\kappa-1)^2 \gamma_{t-1}}$          // $\gamma_t$ is the momentum size
    $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \gamma_t \cdot \eta_t \mathbf{g}_t + (\gamma_t - 1)(\mathbf{w}_t - \mathbf{w}_{t-1})$   // $\eta_t \equiv \frac{2}{\mathsf{L}+\sigma}$
```

- Recall $\gamma_t \downarrow \underline{\gamma} := \frac{2(\kappa+1)}{(\sqrt{\kappa}+1)^2}$; $\gamma_t \equiv \underline{\gamma} \implies$ Polyak's heavy ball:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \frac{4}{(\sqrt{\mathsf{L}}+\sqrt{\sigma})^2} \mathbf{g}_t + \frac{\sqrt{\mathsf{L}}-\sqrt{\sigma}}{\sqrt{\mathsf{L}}+\sqrt{\sigma}} (\mathbf{w}_t - \mathbf{w}_{t-1}).$$

- Both require knowing σ and L

Comparison

$$\|A\mathbf{w}_t - \mathbf{b}\|_2 \le \|\mathscr{C}_t(A)\|_{\mathrm{sp}} \cdot \|A\mathbf{w}_0 - \mathbf{b}\|_2$$

$$\textcircled{1} \le \left[\cosh \ln \left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^t\right]^{-1} \cdot \|A\mathbf{w}_0 - \mathbf{b}\|_2$$

$$\textcircled{2} \le 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^t \cdot \|A\mathbf{w}_0 - \mathbf{b}\|_2$$

$\textcircled{1}$: $|\mathscr{C}_t(\lambda)| = \frac{|\mathscr{T}_t(\mathscr{S}(\lambda))|}{|\mathscr{T}_t(\mathscr{S}(0))|} \le \frac{1}{\left|\mathscr{T}_t\left(\frac{1+\kappa}{1-\kappa}\right)\right|}$, $|\mathscr{T}_t(\lambda)| = \cosh(t \cdot \operatorname{arccosh}|\lambda|)$

$\textcircled{2}$: $\cosh(x) := \frac{\exp(x)+\exp(-x)}{2} \ge \frac{\exp(x)}{2}$, $\operatorname{arccosh} y := \ln(y \pm \sqrt{y^2-1})$

- For Richardson's algorithm:

$$\left(\frac{\kappa-1}{\kappa+1}\right)^t \|A\mathbf{w}_0 - \mathbf{b}\|_2 \le \epsilon \implies t \le \ln\frac{\|A\mathbf{w}_0-\mathbf{b}\|_2}{\epsilon} \Big/ \ln\frac{\kappa+1}{\kappa-1} \le \boxed{\frac{\kappa+1}{2}\ln\frac{\|A\mathbf{w}_0-\mathbf{b}\|_2}{\epsilon}}$$

- For Chebyshev's algorithm:

$$2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^t \|A\mathbf{w}_0 - \mathbf{b}\|_2 \le \epsilon \implies \boxed{t \le \frac{\sqrt{\kappa}+1}{2}\ln\frac{\|A\mathbf{w}_0-\mathbf{b}\|_2}{\epsilon/2}}$$

Possible and the Impossible

- Chebyshev method is minimax optimal
- Richardson method is not optimal
- Memory of 2 suffices!

Can we still do better?!

Conjugate gradient

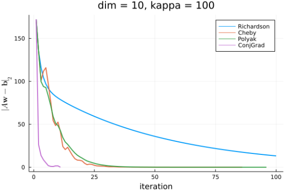
```
Input: $\mathbf{w}_0 \in \mathbb{R}^d$, $A \in \mathbb{S}^d_{++}$, $\mathbf{b} \in \mathbb{R}^d$, $\gamma_0 = 1$
$\mathbf{g}_0 \leftarrow A\mathbf{w}_0 - \mathbf{b}$
$\eta_0 \leftarrow \|\mathbf{g}_0\|_2^2 / \|\mathbf{g}_0\|_A^2$                      // $\|\mathbf{g}\|_A^2 := \langle A\mathbf{g}, \mathbf{g}\rangle$
$\mathbf{w}_1 \leftarrow \mathbf{w}_0 - \eta_0 \mathbf{g}_0$
for $t = 1, 2, \ldots$ do
    $\mathbf{g}_t \leftarrow A\mathbf{w}_t - \mathbf{b}$                                  // gradient
    $\eta_t \leftarrow \|\mathbf{g}_t\|_2^2 / \|\mathbf{g}_t\|_A^2$                             // step size
    $\gamma_t \leftarrow \frac{\eta_{t-1}\|\mathbf{g}_{t-1}\|_2^2\gamma_{t-1}}{\eta_{t-1}\|\mathbf{g}_{t-1}\|_2^2\gamma_{t-1} - \eta_t\|\mathbf{g}_t\|_2^2}$     // $\gamma_t$ is the momentum size
    $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \gamma_t \cdot \eta_t \mathbf{g}_t + (\gamma_t - 1)\,(\mathbf{w}_t - \mathbf{w}_{t-1})$
```

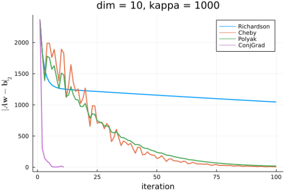
$$\eta_t = \operatorname*{argmin}_{\eta>0} \tfrac{1}{2}\left\langle A(\mathbf{w}_t - \eta\mathbf{g}_t), \mathbf{w}_t - \eta\mathbf{g}_t\right\rangle - \left\langle \mathbf{w}_t - \eta\mathbf{g}_t, \mathbf{b}\right\rangle .$$

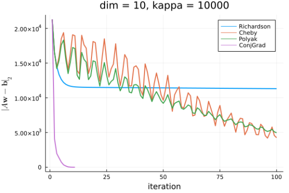
- strikingly similar to Chebyshev's method
- automatically tunes η and γ

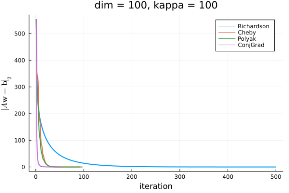
Does it work?

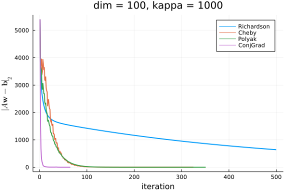
☞ code

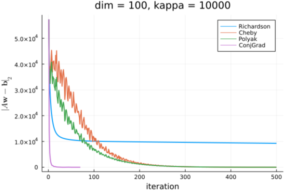












- Cheby and Polyak oscillate! (later we'll see how to iron them)
- Richardson can even be faster (initially or for certain instances)
- Oh boy, that conjugate gradient is fast!