Eigenvalues

A. Donev, Spring 2021

$$\underbrace{|A - \lambda I|}_{\text{poly}_n(\lambda)} = 0$$

Abel's theorem says no closed-form solution for $n \geq 5$

All eigenvalue methods must be iterative/approximate

Sidenote: In fact, solving polynomial eqs is done using eigenvalues (Matlab's roots)

Two cases:

(1) We only want a few eigenvectors – with smallest or largest $|\lambda|$

Google's PageRank algorithm finds the eigenvector with the largest eigenvalue (Power Method)

(2) All eigenvectors (next Wed., pre-recorded)

QR algorithm

Power – Method

$$A = X \Lambda X^{-1}$$

$$A^2 = X \Lambda \underbrace{X^{-1} X}_{= I} \Lambda X^{-1}$$

$$= X \Lambda^2 X^{-1}$$

$$A^n = X \Lambda^n X^{-1}, \quad n \geq 1 \text{ integer}$$

Eigenvalues of A^n are $(\lambda_i)^n$, and eigenvectors are the same.

As $n \to \infty$

$$\Lambda^n = \begin{bmatrix} \lambda_1^n & & \\ & \ddots & \\ & & \lambda_n^n \end{bmatrix}$$

Largest eigenvalue in modulus will dominate Λ^n

Aside: A^T ?

$\overline{T}^{\overline{T}}$

$$\pi^{3/4} = \left(\pi^3\right)^{1/4} = x$$

$$x^4 = \pi^3$$

$$\ln\left(\pi^{\pi}\right) = \pi \ln(\pi)$$

$$\pi^{\pi} = e^{\pi \ln \pi}$$

$$A \cdot A = O(n^3) \text{ FLOPS}$$

$$A \cdot x = O(n^2) \text{ FLOPS}$$

$[n \times n]$ $[n \times 1]$

$$\underbrace{\left(A \ldots \ldots \left(A \cdot \left(A \cdot \left(A \cdot \left(A \cdot x\right)\right)\right)\right)\right)}_{n \text{ times}} = A^n x$$

this can be computed without forming A

A^n

Choose a random vector q_0

compute $x_n = A^n q_0$

$$x_n = \bar{X} \Lambda^n \bar{X}^{-1} q_0$$

$$a$$

$$\bar{X}^{-1} q_0 = a$$

$$\bar{X}\, a = q_0$$

$\vec{a}$ is $\vec{q_0}$ expressed in the eigenbasis of A

$$x_n = X(\Lambda^n a)$$

$$\lim_{n\to\infty} x_n = ?$$

$$\Lambda^n \xrightarrow[n\to\infty]{} \begin{bmatrix} \lambda_1^n & & \\ & \ddots & \\ & & 0 \end{bmatrix}$$

Assume $\lambda_1 > \lambda_2 \geq \lambda_3 \geq \lambda_4 \cdots$

strict inequality

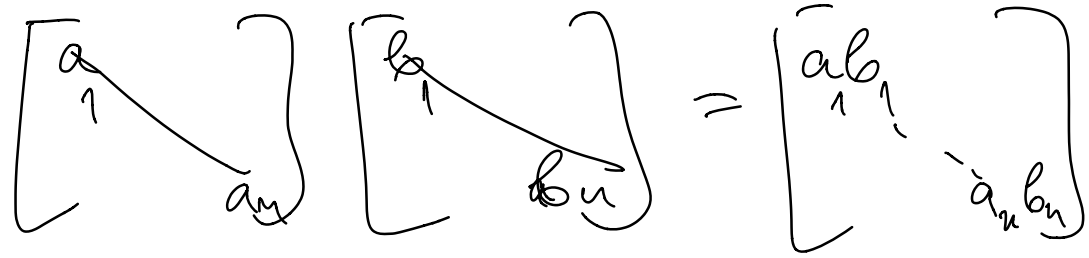
$$\Lambda^n a \longrightarrow \begin{bmatrix} \lambda_1^n a_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$X \begin{bmatrix} \lambda_1^n a_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = x_1 \cdot (\lambda_1^n a_1)$$

$$x_n \longrightarrow (\lambda_1^n a_1)\, x_1$$

$$\frac{x_n}{\|x_n\|} \longrightarrow x_1$$

aside



$$x_n \to x_1$$

How about λ_1

$$A x_n \approx \lambda_1 x_n$$

$$\lambda_1 = \frac{(A x_n)_{1,2,3,\ldots,n}}{(x_n)_{1,2,3\ldots,n}}$$

Rayleigh quotient

$$X_n \cdot A X_n = X_n^T A X_n \Rightarrow$$

$$X_1^T (\lambda_1 X_1)$$

$$= \lambda_1 (X_1^T X_1)$$

$$= \lambda_1$$

$$\lambda_1 \simeq X_n \cdot (A X_n)$$

If X_n is normalized

$$\lambda_1 \simeq \frac{X_n \cdot (A X_n)}{X_n \cdot X_n}$$

$$\min_{X \neq 0} \frac{X^T A X}{X^T X} = \lambda_{min}$$

$$\max_{X \neq 0} \frac{X^T A X}{X^T X} = \lambda_{max}$$

$$\Rightarrow |\lambda_1| \leq |\lambda_{max}|$$

Algorithm (Power method)

1) Choose random q_0

$$\tilde{q}_k = A q_{k-1}$$ (One matrix vector multiply)

$$q_k = \frac{\tilde{q}_k}{\|\tilde{q}_k\|}$$

$$\lambda_k = q_k^T A q_k$$

$$r_k = \|A q_k - \lambda_k q_k\|_2$$

residual

Stop if $r_k < \varepsilon \cdot \lambda_k$ (ε e.g. 10^{-9})

This guarantees 9 digits on λ if $\varepsilon = 10^{-9}$

$$\vec{q}_0 = \overline{\overline{X}}\,\vec{a}$$

$$q_0 = \sum_{i=1}^{n} a_i x_i$$

$$a_i \neq 0 \quad \forall i$$

(why we choose q_0 random)

$$\vec{q}_n = A^n q_0 = A^n \sum a_i x_i$$

$$= \sum a_i \left(A^n x_i\right) =$$

$$= a_i \left(\lambda_i^n x_i\right)$$

$$= \sum \left(a_i \lambda_i^n\right) x_i$$

$$\tilde{q}_n = \sum a_i \lambda_1^n \cdot \left(\frac{\lambda_i}{\lambda_1}\right)^n x_i$$

$$\delta = \left|\frac{\lambda_2}{\lambda_1}\right| \qquad \delta < 1$$

$$\tilde{q}_n \longrightarrow a_1 \lambda_1^n x_1 + O(\delta^n)$$

↖ normalize it

$$\| q_n - (\pm x_1) \| = O(\delta^n)$$

linearly convergent

$$q_n \rightarrow x_1 + O(\delta) x_2 + O(\delta) x_3 + \cdots$$

$$\lambda_n = \vec{q}_n \cdot (A \vec{q}_n)$$

$$= \underbrace{x_1 \cdot (A x_1)}_{\lambda_1} + O(\delta^2) \sum_{i,j \neq 1} x_i \cdot (A x_j)$$

$$\lambda_n = \lambda_1 + O(\delta^2)$$

$$\| \lambda_n - \lambda_1 \| = O(\delta^{2n})$$

Eigenvalue is more accurate

If $\delta << 1$, iteration (power method) converges very fast, especially for eigenvalue

Eigenvalue algorithms converge well (rapidly) if eigenvalues are (well) separated

In Matlab

(QR method)

$[X, \Lambda] = \text{eig}(A)$

all eigs & vectors

$O(n^3)$ but expensive

(much more than LU)

$[X, \Lambda] = \text{eigs}(A, n, \text{eig})$

a few of eigs / vectors

sparse

(Power method)