HW assignments: I will accept late assignments:

- penalty of 7% / day.
- you need to inform me of intent to submit late. (before deadline!)
- I discourage you from submitting late.

HW for today: <u>21.6</u> (Book)
(can start on C2)

Eigenvalue algorithms: Power iteration

MATH 6610 Lecture 18

October 19, 2020

Trefethen & Bau: Lectures 12, 25

Eigenvalues

We've seen that eigenvalues, i.e., numbers λ such that

$$Av = \lambda v, \qquad (v \neq 0)$$

play fundamental roles in linear algebra. How are eigenvalues computed?

Eigenvalues

We've seen that eigenvalues, i.e., numbers λ such that

$$Av = \lambda v, \qquad (v \neq 0)$$

play fundamental roles in linear algebra. How are eigenvalues computed?

Perhaps the conceptually easiest strategy is to compute roots of a polynomial:

$$Av = \lambda v \qquad \Longleftrightarrow \qquad p_A(\lambda) := \det\left(A - \lambda I\right) = 0.$$

~~characteric~~ polynomial

characteristic

Eigenvalues

We've seen that eigenvalues, i.e., numbers λ such that

$$Av = \lambda v, \qquad (v \neq 0)$$

play fundamental roles in linear algebra. How are eigenvalues computed?

Perhaps the conceptually easiest strategy is to compute roots of a polynomial:

$$Av = \lambda v \qquad \Longleftrightarrow \qquad p_A(\lambda) := \det\left(A - \lambda I\right) = 0.$$

While conceptually easy, this turns out to be <u>very</u> difficult practically.

Polynomial roots

We are trying to compute λ satisfying

$$p_A(\lambda) = 0.$$

$A \in \mathbb{C}^{n \times n}$

$$p_A(\lambda) = \lambda^n + \sum_{j=0}^{n-1} a_j \lambda^j$$

coefficients

$a_j = a_j(A)$

Polynomial roots

We are trying to compute λ satisfying

$$p_A(\lambda) = 0.$$

The first, sobering reality: there is no explicit way to do this in general.

Theorem (Abel)

Let $n \geqslant 5$. There is an $n \times n$ matrix A, with rational entries, having an eigenvalue λ that cannot be expressed via elementary arithmetic operations on rational numbers (additions/subtraction, multiplication/division, rational exponentiation).

Polynomial roots

We are trying to compute λ satisfying

$$p_A(\lambda) = 0.$$

The first, sobering reality: there is no explicit way to do this in general.

Theorem (Abel)

Let $n \geqslant 5$. There is an $n \times n$ matrix A, with rational entries, having an eigenvalue λ that cannot be expressed via elementary arithmetic operations on rational numbers (additions/subtraction, multiplication/division, rational exponentiation).

Note that this contrasts with other operations like LU factorizations, QR factorizations, determinants, ...

(Ax=b)

(orthogonalizing vectors)

The result: we cannot obtain *explicit* expressions for eigenvalues in general. We must build approximations from iterative applications of elementary arithmetic operations.

Polynomial rootfinding

We are trying to compute λ satisfying

$$p_A(\lambda) = 0.$$

So our algorithm must be iterative.

There is an even bigger problem: this operation is unstable.

Polynomial rootfinding

We are trying to compute λ satisfying

$$p_A(\lambda) = 0.$$

So our algorithm must be iterative.

There is an even bigger problem: this operation is unstable.

1.) - $x^2 - 2x + 1$: The (relative) condition number of the roots is unbounded
2.) - $\prod_{j=1}^{n}(x - x_j)$: tiny perturbations in monomial coefficients can cause wild changes in roots

1.) $p_A(x) = x^2 - 2x + 1 = (x-1)^2$

$= x^2 + ax + b \quad (a = -2,\ b = 1)$

consider $f: \mathbb{R}^2 \longrightarrow \mathbb{R}$ that maps monic poly. coefficients to a root of p_A. [Say, largest root.]

Is this map well-conditioned?

$$\kappa = \frac{\|\Delta f\| / \|\Delta \binom{a}{b}\|}{\|f\binom{a}{b}\| / \|\binom{a}{b}\|} \qquad @ \binom{a}{b} = \binom{-2}{1}$$

$$\binom{a}{b} + \Delta\binom{a}{b} = \binom{-2}{1-\varepsilon^2} \qquad (\varepsilon \ll 1)$$

$$f\left(\binom{a}{b} + \Delta\binom{a}{b}\right) \longrightarrow \text{roots of } x^2 - 2x + 1 - \varepsilon^2$$

$$\| \qquad\qquad x^2 - 2x + 1 = \varepsilon^2$$

$$1+\varepsilon \qquad\qquad (x-1)^2 = \varepsilon^2$$

$$x = 1 \pm \varepsilon$$

$$\to \text{largest} \to x = 1 + \varepsilon$$

$$f\binom{a}{b} = 1 \Rightarrow \Delta f = \varepsilon$$

$$\|\Delta\binom{a}{b}\| = \varepsilon^2$$

$$\kappa = \frac{\varepsilon / \varepsilon^2}{f\binom{a}{b} / \|\binom{a}{b}\|} = O(1/\varepsilon) \nearrow \infty \text{ as } \varepsilon \downarrow 0.$$

1 $\qquad$ $\sqrt{5}$

This operation is ill-conditioned.

2.) $\prod_{j=1}^{n} (x - x_j)$. $\underbrace{\text{Choose } n=20,\ x_j = j}_{\|}$

$\|$

$$\begin{array}{ccc} \parallel & & \Downarrow \\ x^n + \sum_{j=0}^{n-1} q_j x^j & & \text{"Wilkinson's polynomial"} \end{array}$$

Perturbing q_j by $O(10^{-13})$ results in $O(1)$ changes to roots.

This is an unstable operation (computing roots from monic coefficients)

Polynomial rootfinding

We are trying to compute λ satisfying

$$p_A(\lambda) = 0.$$

So our algorithm must be iterative.

There is an even bigger problem: this operation is unstable.

- $x^2 - 2x + 1$: The (relative) condition number of the roots is unbounded
- $\prod_{j=1}^{n}(x - x_j)$: tiny perturbations in monomial coefficients can cause wild changes in roots

Conclusion: rootfinding is (typically) a bad idea.

Power iteration, I

We'll specialize discussion to Hermitian matrices, with simple eigenvalues.

(why? Formulas and ideas simplify a lot.)

$A \in \mathbb{C}^{n \times n} \rightarrow A = V \Lambda V^*$, $\quad VV^* = I$

diag(Λ) are real-valued. & distinct.

Power iteration, I

We'll specialize discussion to Hermitian matrices, with simple eigenvalues.

Let $\cancel{(\lambda_1, v_1)_{n=1}^N}$ be the ordered eigenpairs of an $n \times n$ matrix A.
The ordering is such that $|\lambda_1| > |\lambda_2| > \cdots$

$(\lambda_n, v_n)_{n=1}^N$

Let $v \in \mathbb{C}^n$, $\quad v = \sum_{j=1}^n c_j v_j$ (mostly we care about case when $c_1 \neq 0$)

$$Av = V \Lambda V^* \left(\sum_{j=1}^n c_j v_j \right) = \sum_{j=1}^n \lambda_j c_j v_j$$

$$A^2 v = V \Lambda V^* \left(\sum_{j=1}^n \lambda_j c_j v_j \right) = \sum_{j=1}^n \lambda_j^2 c_j v_j$$

$$\Rightarrow A^k v = \sum_{j=1}^{n} c_j \lambda_j^k v_j$$

$$= \lambda_1^k \Big[c_1 v_1 + \underbrace{\left(\frac{\lambda_2}{\lambda_1}\right)^k}_{\downarrow \atop 0 \text{ as } k \uparrow \infty} c_2 v_2 + \cdots + \underbrace{\left(\frac{\lambda_n}{\lambda_1}\right)^k}_{\downarrow \atop 0 \text{ as } k \uparrow \infty} c_n v_n \Big]$$

$$A^k v \xrightarrow{k \uparrow \infty} \lambda_1^k c_1 v_1 \longrightarrow \text{this is an eigenvector of } A.$$

How to compute λ_1? $R_A(v) := \dfrac{v^* A v}{\|v\|_2^2}$

Recall: $R_A(v_1) = \lambda_1$

If I compute a good approximation to v_1

$$\Rightarrow \lambda_1 = R_A(v_1)$$

Power iteration, I

We'll specialize discussion to Hermitian matrices, with simple eigenvalues.

Let $(\lambda_1, v_1)_{n=1}^{N}$ be the ordered eigenpairs of an $n \times n$ matrix A.
The ordering is such that $|\lambda_1| > |\lambda_2| > \cdots$

The first algorithm we consider, power iteration, exercises two properties:

- For large k, $\frac{A^k x}{\|A^k x\|_2}$ converges to v_1.
- If $v \approx v_1$, then $R_A(v) := \frac{v^* A v}{\|v\|_2^2}$ is approximately λ_1.

Power iteration, II

Here is a simplistic algorithm to compute the dominant eigenvalue of A:
Initialize with a vector v (e.g., randomly):

1. $v \leftarrow \frac{Av}{\|Av\|_2}$
2. $\lambda \leftarrow R_A(v)$
3. Return to step 1

Power iteration, II

Here is a simplistic algorithm to compute the dominant eigenvalue of A: Initialize with a vector v (e.g., randomly):

1. $v \leftarrow \frac{Av}{\|Av\|_2}$
2. $\lambda \leftarrow R_A(v)$
3. Return to step 1

When to terminate iteration? E.g., at step 3 keep track of $\|Av - \lambda v\|_2$.

This is *not* a particularly useful algorithm, but it does converge.

Power iteration, II

Here is a simplistic algorithm to compute the dominant eigenvalue of A:
Initialize with a vector v (e.g., randomly):

1. $v \leftarrow \frac{Av}{\|Av\|_2}$
2. $\lambda \leftarrow R_A(v)$
3. Return to step 1

When to terminate iteration? E.g., at step 3 keep track of $\|Av - \lambda v\|_2$.

This is *not* a particularly useful algorithm, but it does converge.

Theorem

If the initial vector is not orthogonal to v_1, then at iteration k, power iteration produces vector v and eigenvalue estimate λ satisfying,

$$\|v - v_1\|_2 = \mathcal{O}(r^k), \qquad \cancel{\|\lambda - \lambda_1\|} = \mathcal{O}(r^{2k}),$$

$|\lambda - \lambda_1|$

where $r = |\lambda_2/\lambda_1|. < 1$

"Proof" of Theorem: Assume v at iteration k satisfies $\|v - v_1\|_2 = O(r^k)$.

λ @ iteration k is $\lambda = R_A(v)$

Taylor series around $v = v_1$:

$$R_A(v) = R_A(v_1) + \nabla R_A(v_1) \cdot (v - v_1) + O(\|v - v_1\|^2)$$

dot product

$$|\underbrace{R_A(v)}_{\lambda} - \underbrace{R_A(v_1)}_{\lambda_1}| \lesssim \nabla R_A(v_1) \cdot (v - v_1) + O(\|v - v_1\|^2)$$

Show $\nabla R_A(v_1) = 0$. Assume v_1 is real-valued, A is real-symmetric.

$v = (w_1 \dots w_n)^*$, $w_j \in \mathbb{R}$

$$R_A(v) = \frac{\sum_{i,j=1}^{n} w_i A_{ij} w_j}{\sum_{j=1}^{n} w_j^2}$$

$$\frac{\partial}{\partial w_\ell} R_A(w) = \frac{\left(\sum_{i=1}^{n} A_{i,\ell} w_i + \sum_{j=1}^{n} A_{\ell,j} w_j\right) \|\cancel{w}\, v\|^2 - 2 w_\ell \langle Aw, w \rangle}{\|\cancel{w}\, v\|^4}$$

$$= \frac{2\,(Aw)_\ell \cdot \|\cancel{w}\, v\|^2 - 2\, v_\ell \langle Av, v \rangle}{\|v\|^4}$$

$$= \frac{2}{\|v\|^2}\left((Av)_\ell - v_\ell R_A(v) \right)$$

$$\Rightarrow \nabla R_A(v) = \frac{2}{\|v\|^2}\left(Av - v \cdot R_A(v) \right)$$

$$\nabla R_A(v_1) = \frac{2}{\|v_1\|^2}\left(Av_1 - v_1 \underbrace{R_A(v_1)}_{\lambda_1} \right)$$

$$= \frac{2}{\|v\|^2}\left(Av_1 - \lambda_1 v_1 \right) = 0.$$

$$\Rightarrow |\lambda - \lambda_1| \lesssim \nabla R_A(v_1) \circ (v - v_1) + O(\|v - v_1\|^2)$$

$$= O(\|v - v_1\|^2)$$

$$= O(r^{2k}). \quad \blacksquare$$

Deflation

We have only discussed how to compute a single eigenvalue.

To compute the rest, *deflation* techniques, which reduce A to an $(n-1) \times (n-1)$ matrix are used.

Deflation

We have only discussed how to compute a single eigenvalue.

To compute the rest, *deflation* techniques, which reduce A to an $(n-1) \times (n-1)$ matrix are used.

The simplest (and neither efficient nor stable) deflation technique is Hotelling deflation:
If (λ, v) are a "converged" eigenpair for some eigenpair of A, then define

$$A_2 := A - \lambda v v^*.$$

Then the dominant eigenpair of A_2 is (λ_2, v_2)
Subsequently, power iteration can be used to compute λ_2.

Recall A symmetric $\Rightarrow A = \sum_{j=1}^{n} \lambda_j v_j v_j^*$

$$\rightarrow \quad A - \lambda_1 v_1 v_1^* = \underbrace{\sum_{j=2}^{n} \lambda_j v_j v_j^*}_{\text{rank is } (n-1)} := A_2$$

Now: run power iteration on $A_2 \rightarrow$ results in (λ_2, v_2)

$$A_3 = A_2 - \lambda_2 v_2 v_2^* \quad \ldots \text{ continue.}$$

Deflation

We have only discussed how to compute a single eigenvalue.

To compute the rest, *deflation* techniques, which reduce A to an $(n-1) \times (n-1)$ matrix are used.

The simplest (and neither efficient nor stable) deflation technique is Hotelling deflation:
If (λ, v) are a "converged" eigenpair for some eigenpair of A, then define

$$A_2 := A - \lambda v v^*.$$

Then the dominant eigenpair of A_2 is (λ_2, v_2)
Subsequently, power iteration can be used to compute λ_2.

A full power iteration algorithm employs the basic iterative scheme, along with a termination criterion, and some deflation technique.