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:

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

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:

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

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

Polynomial roots

We are trying to compute λ satisfying

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

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 <u>*cannot*</u> *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, ...

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.

- $x^2 - 2x + 1$: 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

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.

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 $|\lambda_1| > |\lambda_2| > \cdots$

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,

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

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

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.

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.