Eigenvalue algorithms: The QR algorithm

MATH 6610 Lecture 20

October 23, 2020

Trefethen & Bau: Lecture 28

Simultaneous power iteration, I

(A Hermitian)

Let $(\lambda_j, v_j)_{j=1}^n$ be the ordered eigenpairs of A, with $|\lambda_j| > |\lambda_{j+1}|$.

As relatively ineffective as power iteration is, consider applying it to 2 vectors v, w, which have expansions

$$v = \sum_{j=1}^{n} c_j v_j, \qquad w = \sum_{j=1}^{\cancel{N}\, n} d_j v_j.$$

c_j, d_j : scalars

$$A^k [v, w] = \left[\sum_{j=1}^{n} c_j \lambda_j^k v_j \quad \sum_{j=1}^{n} d_j \lambda_j^k v_j \right]$$

$$\approx \left[\lambda_1^k \left[c_1 v_1 + \sum_{j \geq 2} \left(\lambda_j / \lambda_1\right)^k c_j v_j \right], \right.$$

$$\left. \lambda_1^k \left[d_1 v_1 + \sum_{j \geq 2} d_j \left(\lambda_j / \lambda_1\right)^k v_j \right] \right]$$

$$\approx \left[c_1 \lambda_1^k v_1 \qquad \lambda_1^k d_1 v_1 + \lambda_2^k d_2 v_2 \right]$$

next highest order term.

Simultaneous power iteration, I

Let $(\lambda_j, v_j)_{j=1}^n$ be the ordered eigenpairs of A, with $|\lambda_j| > |\lambda_{j+1}|$.

As relatively ineffective as power iteration is, consider applying it to 2 vectors v, w, which have expansions

$$v = \sum_{j=1}^{n} c_j v_j, \qquad w = \sum_{j=1}^{N} d_j v_j.$$

For large k, then:

$$A^k \left[v \;\; w\right] \approx \begin{pmatrix} \lambda_1^k c_1 v_1 & \lambda_1^k \left[d_1 v_1 + \overset{d_2}{} \left(\frac{\lambda_2}{\lambda_1}\right)^k v_2\right] \end{pmatrix}$$

$$= (v_1 \;\; v_2) \begin{pmatrix} c_1 \lambda_1^k & d_1 \lambda_1^k \\ 0 & d_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k \end{pmatrix} =: QR$$

Simultaneous power iteration, II

Extending this argument, if W is some square, full-rank matrix, then

$$A^k W = QR \approx VR,$$

large k

where V is the eigenvector matrix for A.

Simultaneous power iteration, II

Extending this argument, if W is some square, full-rank matrix, then

$$A^k W = QR \approx VR,$$

where V is the eigenvector matrix for A.

We'll slightly modify simultaneous power iteration: perform orthogonalization at every step:

W above

Initialize $Q_0^{PI} = I$. For $k = 0, 1, \ldots,$

1. $A_{k+1}^{PI} := AQ_k^{PI}$
2. $Q_{k+1}^{PI} R_{k+1}^{PI} := A_{k+1}^{PI}$

For large k, we expect $Q_k^{PI} \to V$.

Original simultaneous PI idea: QR decomposition of A^k.

How does this relate to algorithm above?

$$A^k = Q_k R_k \quad (\text{QR decomposition of } A^k)$$

$$A^k = \underbrace{AAA\cdots AA}_{k \text{ times}} = \underbrace{AA\cdots}_{k-1 \text{ times}} \underbrace{\boxed{A Q_1^{PI}}}_{Q_2^{PI} R_2^{PI}} R_1^{PI} \quad (A = A_1^{PI})$$

$$= \underbrace{A\cdots}_{k-2 \text{ times}} \underbrace{\boxed{A Q_2^{PI}}}_{A_3^{PI} = Q_3^{PI} R_3^{PI}} R_2^{PI} R_1^{PI}$$

$$= \cdots = \underline{Q_k^{PI}}\, \underline{R_k^{PI} R_{k-1}^{PI} \cdots R_1^{PI}} \quad (= A^k = \underline{Q_k}\, \underline{R_k})$$

$$\Longrightarrow Q_k^{PI} = Q_k$$

(but computing Q_k^{PI} is more stable than Q_k).

Simultaneous power iteration, II

Extending this argument, if W is some square, full-rank matrix, then

$$A^k W = QR \approx VR,$$

where V is the eigenvector matrix for A.

We'll slightly modify simultaneous power iteration: perform orthogonalization at every step:

Initialize $Q_0^{PI} = I$. For $k = 0, 1, \ldots,$

1. $A_{k+1}^{PI} := AQ_k^{PI}$
2. $Q_{k+1}^{PI} R_{k+1}^{PI} := A_{k+1}^{PI}$

For large k, we expect $Q_k^{PI} \to V$. In fact, we can show that if we have the QR decomposition $A^k = Q_k R_k$, then

$$Q_k^{PI} R_k^{PI} R_{k-1}^{PI} \cdots R_1^{PI} = Q_k R_k$$

So simultaneous power iteration compute Q_k implicitly. $Q_k = Q_k^{PI} \xrightarrow[k\uparrow\infty]{} V$

The QR algorithm

The QR algorithm is a procedure for computing eigenvalues.

(It is distinct from the QR decomposition, but does use QR decompositions.)

The algorithm is so striking that we'll introduce it first without explanation.

As usual we assume A is Hermitian, so that it has a unitary diagonalization: $A = V\Lambda V^*$.

The QR algorithm

The QR algorithm is a procedure for computing eigenvalues.

(It is distinct from the QR decomposition, but does use QR decompositions.)

The algorithm is so striking that we'll introduce it first without explanation.

As usual we assume A is Hermitian, so that it has a unitary diagonalization: $A = V\Lambda V^*$.

1. Compute $A = QR$, the QR decomposition of A
2. Replace A by the procedure $A \leftarrow RQ$
3. Return to step 1

In the limit of an infinite number of iterations, A converges to Λ.

The QR algorithm

The QR algorithm is a procedure for computing eigenvalues.

(It is distinct from the QR decomposition, but does use QR decompositions.)

The algorithm is so striking that we'll introduce it first without explanation.

As usual we assume A is Hermitian, so that it has a unitary diagonalization: $A = V\Lambda V^*$.

"Unshifted/Pure" QR algorithm.

1. Compute $A = QR$, the QR decomposition of A
2. Replace A by the procedure $A \leftarrow RQ$
3. Return to step 1

In the limit of an infinite number of iterations, A converges to Λ. (!!!)

At face value, it's remarkable that this algorithm does *anything* useful.

In fact, this actually is performing simultaneous power iteration in disguise.

Since the unshifted QR algorithm performs power iteration, it's actually not that great.

QR algorithm pseudocode

Set $A_0^{QR} = A$

For $k = 1, 2 \ldots$

$$Q_k^{QR} R_k^{QR} = A_{k-1}^{QR}$$

$$A_k^{QR} = R_k^{QR} Q_k^{QR}$$

Terminate when A_k^{QR} is "sufficiently close" to diagonal.

Goal: see why this is power iteration.

The QR algorithm and simultaneous power iteration

We can now understand why the QR algorithm and simultaneous power iteration are performing similar operations.

The QR algorithm and simultaneous power iteration

We can now understand why the QR algorithm and simultaneous power iteration are performing similar operations.

A useful fact from the QR algorithm is the following relationship:

$$R_j^{QR} Q_j^{QR} = \cancel{Q^Q_{j+1} R R^Q_{j+1} R}, \qquad j \geqslant 1.$$

$$Q^{QR}_{j+1} R^{QR}_{j+1}$$

From algorithm:

$$\left.\begin{aligned} Q_k^{QR} R_k^{QR} &= A_{k-1}^{QR} \\ A_k^{QR} &= R_k^{QR} Q_k^{QR} \end{aligned}\right\} \text{Step } k$$

$$\left.\begin{aligned} Q_{k+1}^{QR} R_{k+1}^{QR} &= A_k^{QR} \\ A_{k+1}^{QR} &= R_{k+1}^{QR} Q_{k+1}^{QR} \end{aligned}\right\} \text{Step } k+1$$

The QR algorithm and simultaneous power iteration

We can now understand why the QR algorithm and simultaneous power iteration are performing similar operations.

A useful fact from the QR algorithm is the following relationship:

$$R_j^{QR} Q_j^{QR} = Q_{j+1}^{QR} R_{j+1}^{QR}, \qquad\qquad j \geqslant 1.$$

This last relation yields the following result via induction:

$$A^k = \left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right) \left(R_k^{QR} R_{k-1}^{QR} \cdots R_1^{QR}\right)$$

Proof (induction)

step 1 of algorithm.

$k=1$: $\quad A^1 = A_0^{QR} = Q_1^{QR} R_1^{QR}$ ✓

Assume true for k, consider $k+1$:

$$A^{k+1} = AA^k = A\left(Q_1^{QR} \cdots Q_k^{QR}\right)\left(R_k^{QR} R_{k-1}^{QR} \cdots R_1^{QR}\right)$$

$$= Q_1^{QR} \underbrace{R_1^{QR}\left(Q_1^{QR}\right.}_{Q_2^{QR} R_2^{QR}} \cdots Q_k^{QR}\left.\right)\left(R_k^{QR} \cdots R_1^{QR}\right)$$

$$= Q_1^{QR} Q_2^{QR} \underbrace{R_2^{QR} Q_2^{QR}}_{Q_3^{QR} R_3^{QR}} Q_3^{QR} \cdots Q_k^{QR}\left(R_k^{QR} \cdots R_1^{QR}\right)$$

$$= Q_1^{QR} Q_2^{QR} Q_3^{QR} R_3^{QR} Q_3^{QR} \cdots Q_k^{QR}\left(R_k^{QR} \cdots R_1^{QR}\right)$$

$$\vdots$$

$$= Q_1^{QR} Q_2^{QR} \cdots Q_{k+1}^{QR} R_{k+1}^{QR}\left(R_k^{QR} \cdots R_1^{QR}\right) \quad \checkmark \quad \blacksquare$$

The QR algorithm's QR decomposition

Finally, we can uncover what the QR algorithm is doing since we have uncovered two QR decompositions of A:

$$A^k = \left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right) \left(R_k^{QR} R_{k-1}^{QR} \cdots R_1^{QR}\right)$$

$$A^k = Q_k^{PI} \left(R_k^{PI} R_{k-1}^{PI} \cdots R_1^{PI}\right)$$

$\Longrightarrow$ (up to uniqueness signs) $Q_k^{PI} = Q_1^{QR} \cdots Q_k^{QR}$

The QR algorithm's QR decomposition

Finally, we can uncover what the QR algorithm is doing since we have uncovered two QR decompositions of A:

$$A^k = \left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right) \left(R_k^{QR} R_{k-1}^{QR} \cdots R_1^{QR}\right)$$
$$A^k = Q_k^{PI} \left(R_k^{PI} R_{k-1}^{PI} \cdots R_1^{PI}\right)$$

Therefore: with $A = V \Lambda V^*$ the eigenvalue decomposition of A, then:

$$\left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right) = Q_k^{PI} \xrightarrow{k \uparrow \infty} V.$$

The QR algorithm's QR decomposition

Finally, we can uncover what the QR algorithm is doing since we have uncovered two QR decompositions of A:

$$A^k = \left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right) \left(R_k^{QR} R_{k-1}^{QR} \cdots R_1^{QR}\right)$$
$$A^k = Q_k^{PI} \left(R_k^{PI} R_{k-1}^{PI} \cdots R_1^{PI}\right)$$

Therefore: with $A = V \Lambda V^*$ the eigenvalue decomposition of A, then:

$$\left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right) = Q_k^{PI} \xrightarrow{k \uparrow \infty} V.$$

Great, but the QR algorithm doesn't compute the entire matrix,

$$\left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right),$$

it just computes Q_k^{QR}. Actually, I just have A_k^{QR}.

The QR algorithm computes eigenvalues

What *does* the QR algorithm compute? Basically just A_k^{QR}.

The QR algorithm computes eigenvalues

What *does* the QR algorithm compute? Basically just A_k^{QR}.

The final property to note is that A_k^{QR} is unitarily equivalent to A:

$$A_k^{QR} = \left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right)^* \; A \; \left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right).$$

Proof:

$$\left.\begin{aligned} Q_k^{QR} R_k^{QR} &= A_{k-1}^{QR} \\ A_k^{QR} &= R_k^{QR} Q_k^{QR} \end{aligned}\right\} \text{ step } k \text{ of QR algorithm.}$$

$$= \left(Q_k^{QR}\right)^* Q_k^{QR} R_k^{QR} Q_k^{QR}$$

$$= \boxed{\left(Q_k^{QR}\right)^* A_{k-1}^{QR} Q_k^{QR} = A_k^{QR}} \quad \blacksquare$$

The QR algorithm computes eigenvalues

What *does* the QR algorithm compute? Basically just A_k^{QR}.

The final property to note is that A_k^{QR} is unitarily equivalent to A:

$$A_k^{QR} = \left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right)^* \; A \; \left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right).$$

But we know that $\left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right) \to V$. Therefore:

$$\begin{aligned} A_k^{QR} &= \left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right)^* \; A \; \left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right) \\ &\overset{k\uparrow\infty}{\longrightarrow} V^* A V = \Lambda. \end{aligned}$$

The QR algorithm and convergence

The QR algorithm therefore performs an eigenvalue decomposition. For real, symmetric matrices, we have convergence

$$A_k^{QR} \to \Lambda,$$

with error c^k, where c depends on the ratio of magnitude of consecutive (ordered) eigenvalues.

error estimate from power iteration $\left(c = \left| \lambda_{j+1} / \lambda_j \right| \right)$

This algorithm actually works for non-Hermitian matrices as well.

For any matrix A, then A_k^{QR} is unitarily equivalent to A (stability!)

The QR algorithm and convergence

The QR algorithm therefore performs an eigenvalue decomposition. For real, symmetric matrices, we have convergence

$$A_k^{QR} \to \Lambda,$$

with error c^k, where c depends on the ratio of magnitude of consecutive (ordered) eigenvalues.

In fact: the real symmetric assumption is not necessary. For (fairly) general matrices A, the QR algorithm computes A_k^{QR} that converges to the Schur factor T in the Schur decomposition

$$A = QTQ^*.$$