Lecture 13

The QR Algorithm

Part 2:

Again, in this lecture: $A = A^T$ (real symmetric)

$$A \underbrace{\begin{bmatrix} | & | & & | \\ v_1 & v_2 & \cdots & v_n \\ | & | & & | \end{bmatrix}}_{V} = \underbrace{\begin{bmatrix} | & | & & | \\ v_1 & v_2 & \cdots & v_n \\ | & | & & | \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix}}_{\Lambda}$$

ONB of eigenvectors — real eigenvalues

Recap

"Pure" QR algorithm

$A^{(0)} = A$

for $k = 1, 2, 3, \ldots$

$Q^{(k)} R^{(k)} = A^{(k-1)}$ [QR factors of $A^{(k-1)}$]

$A^{(k)} = R^{(k)} Q^{(k)}$ [Combine factors backward]

If eigenvalues are distinct,

$$A^{(k)} \to \Lambda \quad \text{and} \quad Q^{(1)} Q^{(2)} \cdots Q^{(k)} \xrightarrow{\pm} V \quad \text{as } k \to \infty.$$

(up to signs of columns)

Convergence established by connection to Power Iterations.

Today

To make QR algorithm computationally efficient:

(1) Each iteration should be "fast."

(2) Iterates should converge to Λ, V quickly.

The cost of a "Pure" QR iteration

At each iteration of "Pure" QR, we must compute

- QR factors of $A^{(k-1)}$ $\sim \frac{4}{3}m^3$ flops [Householder Triangularization Lecture 10]
- Reverse product $R^{(k)}Q^{(k)}$ $\sim 2m^3$ flops [m^2 inner product]

$\Rightarrow$ $O(m^3)$ flops/iteration

Optional argument for total cost of "Pure" QR to motivate 2-Phase Algorithm.

Recall (Lecture 12) that columns of $Q^{(k)}$ converge sequentially to ($\pm$) columns of V. Once first column $q_1^{(k)} \approx \pm v_1$, then $v_1^T q_2^{(k)} \approx 0$ and $q_2^{(k)}$ begins converging to $\pm v_2$, and so on for $q_3^{(k)}, q_4^{(k)}, \ldots, q_m^{(k)}$. In the best case, each column converges rapidly in a few iterations - the total # iterations to resolve m eigenvectors and eigenvalues is then $O(m)$.

$\Rightarrow$ Computing m eigenpairs $\sim O(m^4)$ flops

"Pure" QR iterations are inefficient: even if only a small # iterations are required for <u>each</u> eigenvector/value, Computing $A = V \Lambda V^*$ may be an order of magnitude slower than $A = LU$ or $A = QR$.

2-Phase Algorithm

More efficient to split algorithm into two phases:

$$\underset{A}{\begin{bmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{bmatrix}} \xRightarrow{\text{Phase 1}} \underset{T}{\begin{bmatrix} x & x & & \\ x & x & x & \\ & x & x & x \\ & & x & x \end{bmatrix}} \xRightarrow{\text{Phase 2}} \underset{\Lambda}{\begin{bmatrix} x & & & \\ & x & & \\ & & x & \\ & & & x \end{bmatrix}}$$

Phase 1: Reduction to tridiagonal form.

Similarity transform preserves eigenvalues of A.

$$T = \underbrace{O^T A O}_{\text{orthogonal matrix } O} = O^T V \Lambda V^T O = \underbrace{[O^T V]}_{\substack{\text{orthogonal} \\ \text{eigenvectors} \\ \text{of } T}} \Lambda [O^T V]^T$$

T has eigenvalues Λ and eigenvectors $O^T V$.

Idea Construct O^T from Householder reflections that introduce zeros below 1^{st} subdiagonal in columns of A. Because A is symmetric, then O (multiplied from the right) will introduce zeros to the right of 1^{st} superdiagonal in rows of A.

$\Rightarrow$ $T = O^T A O$ is tridiagonal.

E.g.)

1st row untouched — 1st column untouched — new zeros

$$\underset{\substack{A \\ \text{(real symmetric)}}}{\begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix}} \xRightarrow{Q_1^T \cdot} \underset{Q_1^T A}{\begin{bmatrix} x & x & x & x & x \\ \boxed{x} & \boxed{x} & \boxed{x} & \boxed{x} & \boxed{x} \\ 0 & \boxed{x} & \boxed{x} & \boxed{x} & \boxed{x} \\ 0 & \boxed{x} & \boxed{x} & \boxed{x} & \boxed{x} \\ 0 & \boxed{x} & \boxed{x} & \boxed{x} & \boxed{x} \end{bmatrix}} \xRightarrow{\cdot Q_1} \underset{Q_1^T A Q_1}{\begin{bmatrix} x & \boxed{x} & 0 & 0 & 0 \\ x & \boxed{x} & \boxed{x} & \boxed{x} & \boxed{x} \\ 0 & \boxed{x} & \boxed{x} & \boxed{x} & \boxed{x} \\ 0 & \boxed{x} & \boxed{x} & \boxed{x} & \boxed{x} \\ 0 & \boxed{x} & \boxed{x} & \boxed{x} & \boxed{x} \end{bmatrix}}$$

new zeros

$$Q_2^T \cdot \vdots \cdot Q_2 \Rightarrow \begin{bmatrix} \times & \times & 0 & 0 & 0 \\ \times & \times & \boxtimes & 0 & 0 \\ 0 & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\ 0 & 0 & \boxtimes & \boxtimes & \boxtimes \\ 0 & 0 & \boxtimes & \boxtimes & \boxtimes \end{bmatrix} \Rightarrow \cdots \Rightarrow \begin{bmatrix} \times & \times & & & \\ \times & \times & \times & & \\ & \times & \times & \boxtimes & \\ & & \boxtimes & \boxtimes & \boxtimes \\ & & & \boxtimes & \boxtimes \end{bmatrix}$$

$$Q_2^T Q_1^T A Q_1 Q_2 \qquad T = \underbrace{Q_{m-2}^T \cdots Q_2^T Q_1^T}_{Q^T} A \underbrace{Q_1 Q_2 \cdots Q_{m-2}}_{Q}$$

B/c symmetry is maintained at each stage, we can work with just the lower triangular part of $A, Q_1^T A Q_1, \ldots, T$ - similar to Cholesky! (Lecture 8)

$$\text{Phase 1 cost} \sim \tfrac{4}{3} m^3$$

<u>Phase 2:</u> Tridiagonal QR iterations.

QR iterations with T are significantly faster than A!

<u>Compute QR</u>

one new nonzero $\qquad$ one new nonzero

$$T = \begin{bmatrix} \times & \times & & & \\ \times & \times & \times & & \\ & \times & \times & \times & \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix} \overset{Q_1^T}{\Rightarrow} \begin{bmatrix} \boxtimes & \boxtimes & \boxtimes & & \\ 0 & \boxtimes & \boxtimes & & \\ & \times & \times & \times & \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix} \overset{Q_2^T}{\Rightarrow} \begin{bmatrix} \times & \times & \times & & \\ 0 & \boxtimes & \boxtimes & \boxtimes & \\ & 0 & \boxtimes & \boxtimes & \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}$$

one new zero $\qquad$ one new zero

To compute $T = QR$, we only need to "zero out" one subdiagonal entry in each column. We use 2×2 Householder

$$Q_1^T = \begin{bmatrix} F_1 & \\ & \underbrace{\begin{matrix} 1 & & \\ & \ddots & \\ & & 1 \end{matrix}}_{m-2 \text{ identity}} \end{bmatrix}, \quad Q_2^T = \begin{bmatrix} \overset{1\times 1 \text{ identity}}{1} & & \\ & F_2 & \\ & & \underbrace{\begin{matrix} 1 & & \\ & \ddots & \\ & & 1 \end{matrix}}_{m-3 \text{ identity}} \end{bmatrix}, \ldots, Q_k = \begin{bmatrix} \overset{(k-1)\times(k-1)}{\begin{matrix} 1 & & \\ & \ddots & \\ & & 1 \end{matrix}} & & \\ & F_k & \\ & & \underbrace{\begin{matrix} 1 & & \\ & \ddots & \\ & & 1 \end{matrix}}_{m-k-2 \text{ identity}} \end{bmatrix}$$

where $F_1, F_2, \ldots, F_k$ are 2×2 Householder matrices.

Note that Q_k^* only changes 3 entries in row k and 3 entries in row $k+1$, so that

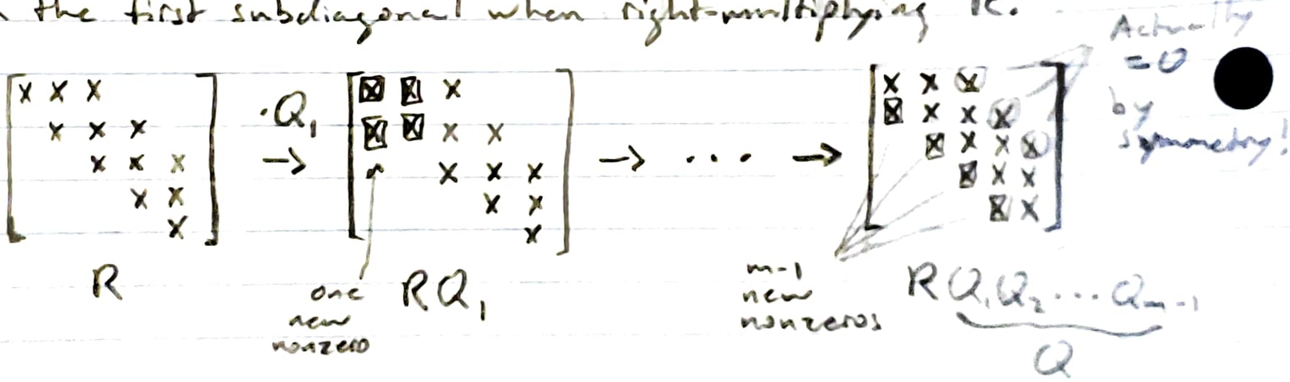
$$\text{cost of } T = QR \text{ is } O(m) \text{ flops} \quad (\text{as } m \to \infty)$$

Compute RQ

Critically, RQ is again symmetric tridiagonal.

$$RQ = (Q^* T) Q = Q^* T Q \qquad (\text{real symmetric})$$

Note that $Q = Q_1 Q_2 \cdots Q_{m-1}$ only adds new nonzeros on the first subdiagonal when right-multiplying R.



Since $RQ = Q^*TQ$ is symmetric, the "nonzero" entries above the 1^{st} superdiagonal must actually be zero!
$\Rightarrow RQ = Q^*TQ$ is again tridiagonal.

$$\text{cost of } RQ \text{ is } O(m) \text{ flops}$$

So each iteration of QR on T produces another symmetric tridiagonal matrix in $O(m)$ flops.

$$\text{Phase 2 cost} \approx \#\text{iterations} \cdot O(m) \text{ flops}$$

2-Phase Summary

$\Rightarrow$ Phase 1 is dominant cost (comparable to single QR factorization!) after which Tridiagonal QR iterations (Phase 2) are fast.

Convergence Rate for "Pure" QR Iterations

In Lecture 12, we linked "Pure" QR iterations to Simultaneous Power Iterations to establish convergences

(*) $\quad A^k = \underline{Q}^{(k)}\underline{R}^{(k)}$ and $A^{(k)} = \underline{Q}^{(k)T} A \underline{Q}^{(k)}$ [Lecture 12 Thm 2]

where $\underline{Q}^{(k)} = Q^{(1)}Q^{(2)}\cdots Q^{(k)}$ and $\underline{R}^{(k)} = R^{(k)}R^{(k-1)}\cdots R^{(1)}$.

The first column $q_1^{(k)}$ of $\underline{Q}^{(k)}$ converges to $\pm v_1$ at rate $\left|\frac{\lambda_2}{\lambda_1}\right|$:

$$\Rightarrow q_1^{(k)} r_{11}^{(k)} = \underline{Q}^{(k)}\underline{R}^{(k)} e_1 = A^k e_1$$

$$\Rightarrow q_1^{(k)} = \frac{1}{r_{11}^{(k)}} A^k \overbrace{\left[(v_1^T e_1)v_1 + (v_2^T e_1)v_2 + \dots + (v_m^T e_1)v_m\right]}^{e_1}$$

$$= \frac{\lambda_1^k}{r_{11}^{(k)}}\left[(v_1^T e_1)v_1 + \underbrace{\left[\frac{\lambda_2}{\lambda_1}\right]^k (v_2^T e_1)v_2 + \dots + \left[\frac{\lambda_m}{\lambda_1}\right]^k (v_m^T e_1)v_m}_{\text{component of } q_1^{(k)} \text{ in directions other than } v_1 \text{ decays like } \left|\frac{\lambda_2}{\lambda_1}\right|^k}\right]$$

Since $q_1^{(k)}$ and v_1 are both normalized, $|q_1^{(k)} - \pm v_1| = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right)$ as $k\to\infty$. Similarly, $|q_j^{(k)} - \pm v_j| = O\left(\left|\frac{\lambda_{j+1}}{\lambda_j}\right|^k\right)$ as $k\to\infty$.

$\Rightarrow$ Slow convergence when $\lambda_j \approx \lambda_{j+1}$ for some $1\le j\le m-1$

Shifted QR Iterations

QR iterations can be accelerated by introducing shifts.

$$A^{(k-1)} - \mu^{(k)} I = Q^{(k)}R^{(k)}, \quad A^{(k)} = R^{(k)}Q^{(k)} + \mu^{(k)} I$$

shift at k^{th} iteration

In place of (*) above, we have

$$(**) \quad (A-\mu^{(k)}I)(A-\mu^{(k-1)}I)\cdots(A-\mu^{(1)}I) = \underline{Q}^{(k)}\underline{R}^{(k)},$$

with $A^{(k)}$, $\underline{Q}$, and $\underline{R}$ the same as in (*).

"Shift-and-invert" Power Iterations

The power method is often accelerated with a "shift-and-invert" transformation: Note that if $Av_j = \lambda_j v_j$ then $(A-\mu I)^{-1}v_j = (\lambda_j-\mu)^{-1}v_j$ so that

$$(A-\mu I)^{-1}x = \frac{v_1^Tx}{\lambda_1-\mu}v_1 + \frac{v_2^Tx}{\lambda_2-\mu}v_2 + \cdots + \frac{v_m^Tx}{(\lambda_m-\mu)}v_m.$$

If μ is closest to λ_J, the power method with $(A-\mu I)^{-1}$ converges with rate $\min_{j\neq J}\left|\frac{\lambda_j-\mu}{\lambda_J-\mu}\right|$. If we have a good estimate of an eigenvalue λ_J, this can be a huge improvement over power method for A.

Rayleigh Quotient Iteration (RQI)

We can do even better if we update the shift at each iteration, using the approximation to v_J to get a better approximation to λ_J.

Given $x^{(0)}$ (start vector)

for $k=1,2,3,\dots$

$\hat{x}^{(k)} = (A-\mu^{(k)}I)\,x^{(k-1)}$	shift-and-invert
$x^{(k)} = \hat{x}^{(k)}/\lVert\hat{x}^{(k)}\rVert$	normalize
$\mu^{(k)} = x^{(k)T}Ax^{(k)}$	Rayleigh Quotient

All but a measure zero set

RQI converges for "almost every" starting vector and convergence is cubic when $x^{(k)}$ gets close to eigvec v_J:

$$\|x^{(k+1)} - (\pm v_J)\| = O(\|x^{(k)} - (\pm v_J)\|^3)$$

as $k \to \infty$.

$$|\mu^{(k+1)} - \lambda_J| = O(|\mu^{(k)} - \lambda_J|^3)$$

E.g. if $\|x^{(k)} - (\pm v_J)\| \approx 10^{-2}$, then $\|x^{(k+1)} - (\pm v_J)\| \approx 10^{-6}$

and $\|x^{(k+2)} - (\pm v_J)\| \approx 10^{-18}$. Cubic convergence is fast!

Shifted QR and RQI

Just as the power method applied to $m \times m$ identity helped us to understand "Pure" QR, RQI applied to a special matrix will help us understand "Shifted" QR.

We start by inverting (**)

$$(A - \mu^{(1)} I)(A - \mu^{(2)} I) \cdots (A - \mu^{(k)} I) = (\underline{R}^{(k)})^{-1} \underline{Q}^{(k)T}$$

$$(\text{LHS is symmetric so RHS must be} \Rightarrow) = \underline{Q}^{(k)} (\underline{R}^{(k)})^{-T}$$

Let $P = \begin{bmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{bmatrix}$ and note $P^2 = I$. We have that

$$\underbrace{(A - \mu^{(1)} I)(A - \mu^{(2)} I) \cdots (A - \mu^{(k)} I)}_{A(\mu^{(1)}, \mu^{(2)}, \ldots, \mu^{(k)})} P = \underbrace{[\underline{Q}^{(k)} P]}_{\text{orthogonal mat.}} \underbrace{[P (\underline{R}^{(k)})^{-T} P]}_{\text{upper tri. mat.}}$$

This is a QR factorization of $A(\mu^{(1)}, \mu^{(2)}, \ldots, \mu^{(k)})$, and first column of $\underline{Q}^{(k)} P$ – that is last column of $\underline{Q}^{(k)}$ – is the result of apply k steps of RQI to the first column of P, ... e_m.

Consequently, the last column, $q_m^{(k)}$, of $\underline{Q}^{(k)}$ converges to an eigenvector of A rapidly if the shifts $\mu^{(k)}$ are chosen to be the Rayleigh-Quotient shifts

$$\mu^{(k)} = q_m^{(k)T} A q_m^{(k)}.$$

Notice from $(*)$-$(**)$, that since $A^{(k)} = \underline{Q}^{(k)T} A \underline{Q}^{(k)}$,

$$\Rightarrow \quad \mu^{(k)} = q_m^{(k)T} A q_m^{(k)} = e_m^T A^{(k)} e_m = A^{(k)}_{m,m}$$

so our shifts are readily available!

Deflation

When $q_m^{(k)} \approx v_j$ (convergence of $q_m^{(k)}$ to an eigvec of A)

we have
$$A^{(k)} = \underline{Q}^{(k)T} A \underline{Q}^{(k)} = \begin{bmatrix} x & x & & & \\ x & x & x & & \\ & x & x & x & \\ & & x & x & 0 \\ & & & 0 & x \end{bmatrix} \leftarrow \text{(symmetry)}$$

b/c $A^{(k)}_{m,m-1} = q_{m-1}^{(k)T} A q_m^{(k)} \approx \lambda_j q_{m-1}^{(k)T} q_m^{(k)} = 0$

We can continue diagonalizing just the top left $(m-1)\times(m-1)$ submatrix w/further shifted QR iterations applied to this smaller submatrix. This is called <u>deflation</u>.

Practical Notes

- "Wilkinson Shifts" (see LNT Lecture 29) avoids rare stalled conv.
- "Aggressive Early deflation" breaks $A^{(k)}$ into subproblems whenever <u>any</u> subdiag entry is close to zero.
- "Implicit Shifts" Combine the steps QR and RQ steps by applying Householder from left & then right to compute $A^{(k)} = Q^{(k)T} A^{(k-1)} Q^{(k)T}$ directly. Google "<u>Bulge-Chasing QR</u>."