Ma221 Lecture 10

Symmetric Eigenproblems + SVD

Goals: Perturbation Theory

Algorithms

Perturbation Theory:

Theorems needed to explain algorithms

Real symmetric $A = A^T$

(Complex Hermitian $A = A^*$ analogous)

$A = Q \Lambda Q^T$

$\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n) \quad \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$

$Q = [q_1, \ldots, q_n]$

Complex symmetric different

$[1 \quad i] \quad i = \sqrt{-1}$, double eval at 0

$[i \quad -1]$, not diagonalizable
Most results hold for SVD of general matrix \textit{(Thm 3.3 part 4)}

\[ B = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} = B^T \]

Eigendecomposition of \( B \) closely related to SVD of \( A \), evals of \( B = \pm \) singular vals of \( A \) plus some zeros if \( A \) rectangular.

Small change of \( A \) to \( A + E \)

is a small symmetric change in \( B \).

Perturbation theory for \( B \) applies to \( A \).

\textbf{Def: Rayleigh Quotient} \( p(u, A) = \frac{u^T A u}{u^T u} \)

**Properties:**

- If \( A u = \lambda u \) \( \Rightarrow p(u, A) = \lambda \)

\[ u = \sum \frac{b_i}{\sigma_i} q_i = Q b, \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} \]

\[ p(u, A) = \frac{(Q b)^T A (Q b)}{(Q b)^T (Q b)} = \frac{b^T Q^T A Q b}{b^T Q Q b} = \frac{b^T \Lambda b}{b^T b} \leq \lambda_{\text{max}} b^T b \leq \| b \|^2 \]
= convex combination of $\{d_j, \ldots, d_n\}$

\[ \lambda_1 \geq \rho(v_J A) \geq \lambda_n \]

\[ \lambda_1 = \max_{u \neq 0} \rho(v_J A) \]

\[ \lambda_n = \min_{u \neq 0} \rho(v_J A) \]

In fact all evals expressible using $\rho(v_J A)$

Courant-Fischer Minimax Theorem

\[ R^j = j\text{-dimensional subspace of } R^n \]

\[ S^{n-j+1} = n-j+1\text{-dimensional subspace} \]

\[ \max_{R^j} \min_{0 \neq r \in R^j} \rho(r, A) = \lambda_j \]

\[ = \min_{S^{n-j+1}} \max_{0 \neq s \in S^{n-j+1}} \rho(s, A) \]

\[ \max_{R^j} \text{ attained by } \operatorname{span}(g_j, \ldots, g_j) \]

\[ \min_{S^{n-j+1}} \text{ attained by } \operatorname{span}(g_i, g_{i+1}, \ldots, g_n) \]

\[ \min_{R^j} \text{ attained by } r = g_i \]

\[ \max_{S \in S^{n-j+1}} \text{ attained by } s = g_i \]
Proof: Given $R^j$ and $S^{n-j+1}$
their dimensions add up to $j + (n-j+1) = n+1$
\[\Rightarrow\text{most intersect in some nonzero } x_{RS}\]
\[
\min_{0 \neq r \in R^j} p(r, A) \leq p(x_{RS}, A) \leq \max_{0 \neq s \in S^{n-j+1}} p(s, A)
\]
Let $R'$ maximize $\min_{0 \neq r \in R^j} p(r, A)$

Let $S'$ minimize $\max_{0 \neq s \in S^{n-j+1}} p(s, A)$

\[
\sum_i j_i = \max_{0 \neq r \in R^j} \min_{0 \neq r' \in R'} p(r, A) = \min_{0 \neq r' \in R'} \max_{0 \neq r \in R^j} p(r, A) \\
\leq p(x_{RS}, s_{RS'}) \leq \max_{0 \neq s \in S^{n-j+1}} p(s, A) = \min_{0 \neq s \in S^{n-j+1}} \max_{0 \neq s' \in S^{n-j+1}} p(s', A)
\]

If we choose $R^j = \text{span}(q_1, \ldots, q_j)$
r = q_j \Rightarrow
\[
\min_{0 \neq r' \in R^j} p(r, A) = p(q_j, A) = d_j
\]

If we choose $S^{n-j+1} = \text{span}(q_{j+1}, q_{j+2}, \ldots, q_n)$
and $s = q_j \Rightarrow$
\[
\max_{0 \neq s \in \mathbb{S}^{n-1}} \rho(s, A) = \rho(q_i, A) = \delta_i
\]

\[\Rightarrow \text{ all equalities in } (*)\]

\[\text{Weyl's Thm: } A = A^T \text{ with evals } \lambda_1, \ldots, \lambda_n \text{ and } (A+E) = (A+E)^T \]

\[\text{with evals } \mu_1, \ldots, \mu_n \text{ then } \]

\[|\lambda_i - \mu_i| \leq \|E\|_2 \text{ for all } i\]

\[\text{Corollary: if } A \text{ general, with singular vals } 0, \lambda_2, \ldots, \lambda_n \]

\[\text{and } A+E \text{ has sing vals. } \tau_1, \ldots, \tau_n \text{ then } |\tau_i - \lambda_i| \leq \|E\|_2 \text{ for all } i\]

(follows from forming \[\begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix}\])
Proof of Weyl:

\[ N_j = \min_{S^{n-j+1}} \max_{0 \neq s \in S^{n-j+1}} \rho (s, A + E) \]

\[ = \min \max \frac{s^T (A + E) s}{s^T s} \]

\[ = \min \max \frac{s^T A s}{s^T s} + \frac{s^T E s}{s^T s} \]

\[ \leq \| E \|_2 \]

\[ \leq \min \max \frac{s^T A s}{s^T s} + \| E \|_2 \]

\[ = \lambda_j + \| E \|_2 \]

\[ \nu_i = \lambda_i + \| E \|_2 \quad \text{for all } i \]

swap roles of \( \nu_i \) and \( \lambda_j \), get

\[ \lambda_j \leq \nu_i + \| E \|_2 \Rightarrow |\lambda_j - \nu_i| \leq \| E \|_2 \]

Def: Inertia (A) = (# negative evals of A, # zero evals of A, # positive evals of A)
Sylvester's Thm: $A = A^T$, nonsingular

$\text{Inertia}(A) = \text{Inertia}(X^TAX)$

Fact: Suppose we factor $A = LDL^T$
(Gaussian Elim with symmetric or no pivoting)

$\text{Inertia}(A) = \text{Inertia}(D)$

$= (\# D_{ii} < 0, \# D_{ii} = 0, \# D_{ii} > 0)$

Factor $A - xI = L' \cdot D' \cdot L'^T$

$\# D'_{ii} < 0 = \# \text{eigenvals of } A - xI < 0
= \# \text{eigvals of } A < x$

Factor $A - yI$, compute

$\# \text{eigvals of } A \leq y, x \leq y$

$\# \text{eigvals of } A \in [x, y] \Rightarrow$

$\# \text{eigvals} \leq y - \# \text{eigvals} < x
\Rightarrow \text{can count \# eigvals in any interval}$

$\text{Inertia} (A - \frac{x+y}{2} I)$
can keep bisecting intervals that contain at least one eval to get upper and lower bounds as tight as desired, and can only compute ones in "interval of interest"
More details on implementation later

Proof of Sylvester's Thm:

Suppose \( \# \) evals of \( A < 0 \) is \( m \)
and \( \# \) evals of \( X^T A X < 0 \) is \( m' \) but \( m' < m \)

Seek contradiction:
\( N = m \)-dimensional subspace for
\( m \) negative evals of \( A \)
\( \exists \mathbf{x} \in N \Rightarrow x^T A x < 0 \)

\( P = n - m' \)-dimensional subspace
for \( n - m' \) nonnegative evals
of \( X^T A X \), so
\( x \in P \Rightarrow x^TAXx = (Xx)^TAXx > 0 \)
\[ \text{or } y \notin \mathcal{X}P \Rightarrow y^TAy > 0 \]
\[
\dim(\mathcal{X}P) + \dim(\mathcal{N}) = n - m' + m > n
\]
\( \Rightarrow \mathcal{X}P \) and \( \mathcal{N} \) must intersect in \( z \neq 0 \)
\[
\Rightarrow z^TAz < 0 \text{ and } z^TAx > 0
\]
\( \Rightarrow \) a contradiction

\underline{Ma 221 Lecture 10 Segment 3}

\textbf{Perturbation Theory for Evecs:}

\textbf{Thm:} \( A = Q\Lambda Q^T \quad \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n) \)
\[
Q = [q_1, \ldots, q_n]
\]
\( A + E = Q^T\Lambda'Q \quad \Lambda' = \text{diag}(\lambda'_1, \ldots, \lambda'_n) \)
\[
Q' = [q'_1, \ldots, q'_n]
\]
(\( q_i \) : angle between \( q_i \) and \( q'_i \)

\[
\text{gap}(i, A) = \min_{j \neq i} |\lambda_i - \lambda_j| . \text{ Then}
\]
\[
1.5 \sin(2\Theta_i) \leq \frac{\|E\|_2}{\text{gap}(i, A)}
\]
\[
\varphi_c < \varphi \Rightarrow \frac{1}{2} \sin(2\theta_c) \approx \theta_c
\]
\[
\Rightarrow \| \mathbf{E} \|_2 < \text{gap}(\varepsilon, \Delta) \Rightarrow \theta_c \text{ small}
\]

Why \( \text{gap}(\varepsilon, \Delta) \) in denominator?

Worst case: \( A = I \Rightarrow \text{gap} = 0 \)

showed that we can move evecs arbitrarily with arbitrarily small \( \| \mathbf{E} \|_2 \)

Proof of a weaker result (see text for full result)

Write evec of \( A + \mathbf{E} \) as \( \hat{\varphi}_i + d \),

where \( d^T \hat{\varphi}_i = 0 \)

\[
\hat{\varphi}_i = \frac{\varphi_i + d}{\| \varphi_i + d \|} : \text{goal: bound } d = \tan \theta_c
\]

\[
(A + \mathbf{E}) (\varphi_i + d) = \lambda_c \varphi_i + \lambda_c d
\]

\[
A \cdot \varphi_i + \mathbf{E} \varphi_i + A \cdot d = \lambda_c \varphi_i + \lambda_c d \quad \text{ignore } \mathbf{E} d
\]

\[
(A + \mathbf{E} - \lambda_c I) \varphi_i = (\lambda_c I - A) d
\]

\[
(\lambda_c I + \mathbf{E} - \lambda_c I) \varphi_i = (\lambda_c I - A) d
\]

\[
d = \sum_{i \neq c} d_{ij} \varphi_j
\]
\[
\text{LHS} = (\lambda_i I + E - \lambda'_j I) g_i = \sum_{k \neq i} (\lambda_k' - \lambda_j) d_k \cdot g_i = \text{RHS}
\]

\[\|\text{LHS}\|_2 \leq 2\|E\|_2 \text{ because } |\lambda_i - \lambda_j| \leq \|E\|_2 \text{ by Weyl's Theorem.}
\]

\[\|\text{RHS}\|_2 \geq (\text{gap}(i, A) - \|E\|_2) \|d_i\|_2
\]

\[\lambda'_j - \lambda_j = \lambda'_j - \lambda_j + \lambda_j - \lambda_i \leq \|E\|_2 \geq \text{gap}
\]

\[\frac{2\|E\|_2}{\text{gap}(i, A)} \geq \frac{2\|E\|_2}{\text{gap}(i, A) - \|E\|_2} \geq \|d_i\|_2 = (\tan \theta_i) |\lambda_i| \quad \text{if } |\theta_i| \leq 1
\]

If \(\|E\|_2 \gg \text{gap}(i, A)\)

---

**Ma221 Lecture 10 Segment 4**

More results on Rayleigh Quotients

\[\rho(u, A) = \frac{u^T A u}{u^T u}
\]

Rayleigh Quotient is a "best approximation" to an eval

**Thm:** \(\|x\|_2 = 1\) and \(B\) given.

Then \(A\) has an eigenvalue \(\alpha\)

\[|\alpha - \beta| \leq \|A x - B x\|_2
\]
Given only \( x \), \( \beta = p(x, A) \) minimizes \( \| A x - \beta \|_2 \).

Given any unit vector \( x \), there is always an eigenvalue within distance \( \| A x - p(x, A) \cdot x \|_2 \) of \( p(x, A) \), and \( p(x, A) \) minimizes distance.

Now let \( \lambda_i \) be eval of \( A \) closest to \( p(x, A) \), and \( \text{gap} = \min \| \lambda_j - p(x, A) \| \)

Then

\[
| \lambda_i - p(x, A) | \leq \frac{\| A x - p(x, A) \cdot x \|_2}{\text{gap}}
\]

i.e. error in \( p(x, A) \) as an approximate eval of \( A \) \( \| A x - p(x, A) \cdot x \|_2 \)

Proof of Part 1: \( \| x \|_2 = 1 \Rightarrow \]

\[
1 = \| x \|_2 = \| (A - \beta I) (A - \beta I) x \|_2
\]

\[
\leq \| (A - \beta I) \|_2 \cdot \| A x - \beta x \|_2
\]

\[
\leq \frac{\| A x - \beta x \|_2}{\min \| \lambda \|_2 - \beta} \cdot \| A x - \beta x \|_2
\]

Assume \( A = Q \Lambda Q^T \)

\[
= \| (A - \beta I) \|_2 \cdot \| A x - \beta x \|_2
\]
Part 2: to show that $k = p(x, A)$ minimizes $||Ax - Bx||_2$:

$$Ax - Bx = \left( \frac{Ax - p(A)x + p(x, A)x - Bx}{\gamma} \right)$$

if $y^Tz \geq 0$ then

$$||Ax - Bx||_2 = (||Ax - p(A)x ||_2^2 + ||p(x, A)x - Bx||_2^2)$$

by Pythagorean Theorem

$$\Rightarrow ||Ax - p(x, A)x||_2 \leq ||Ax - Bx||_2 \forall \beta$$

$$y^T = (p(x, A) - \beta)x^T (Ax - p(x, A)x)$$

$$= (p(x, A) - \beta) (Ax - p(x, A)x^T A - p(x, A)x^T A)$$

$$= 0 \text{ by def of } p(x, A)$$

Last Part: Special case of

A $2 \times 2$ diagonal matrix, captures all ideas of general proof.

$A = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$ \quad $x = \begin{bmatrix} c \\ s \end{bmatrix}$ \quad $c^2 + s^2 = 1$

$p(x, A) = c^2 \lambda_1 + s^2 \lambda_2$

Assume $c > s \Rightarrow p(x, A)$ closer to $d_1$ and hence

$$|\lambda_1 - p(x, A)| = |\lambda_1 - c^2 \lambda_1 - s^2 \lambda_2|$$

$$= |s^2 \lambda_1 - s^2 \lambda_2| = s^2 |\lambda_1 - \lambda_2|$$
\[ \text{gap} = |d_2 - \rho(x,A)| = |d_2 - c^2 d_1 - s^2 d_2| = c^2 |d_1 - d_2| \]

\[ r = A_k \cdot \rho(x,A) \cdot x = \begin{bmatrix} d_1 & c \\ d_2 & s \end{bmatrix} - (c^2 d_1 + s^2 d_2) \begin{bmatrix} c \\ s \end{bmatrix} \]

\[ = \begin{bmatrix} c s^2 (d_1 - d_2) \\ s c^2 (d_2 - d_1) \end{bmatrix} \]

\[ \|r\|_2^2 = s^2 c^2 (d_1 - d_2)^2 (s^2 + c^2) \]

\[ \frac{\|r\|_2^2}{\text{gap}} = s^2 |d_1 - d_2| = |d_1 - \rho(x,A)| \]

i.e. bound in Thm is exact.

Result important later to show that QR iteration from Chap 4 applied to \( A = A^T \) converges cubically, not "merely" quadratically.
Ma221 Lecture10 Segment5

Overview of Algorithms

(1) "Usual Accuracy"

backward stable: exact
evals + evecs for $A + E$
$\|E\|_1 = O(\epsilon); \|A\|_1$

(1.1) All evals (with or without evecs)

(1.2) Just get evals in $[x,y]$ (w or w/o evecs)

(1.3) Just get $\lambda_i$ through $\lambda_j$ (w or w/o evecs)

Eg: $\lambda_1$ through $\lambda_{10}$: 10 largest evals

(1.2) and (1.3) cheaper than (1.1) when only few desired
(2) "High Accuracy": get tiny
    evals and avcs more accurately
    than "usual accuracy"

Ex: If A well-conditioned
     i.e., all singular values same magnitude
     error bound O(e). ||A|| ⇔ all
     singular values computed with small
     relative errors
     B = D·A where D diagonal
     with some large and tiny entries
     ⇔ B has some tiny, some large
    singular values, error bound
    O(e). ||D||·A|| ⇔ tiny sing vals.
    do not have (many) correct
     leading digits. But there is
     a "tighter" perturbation theory
     and more accurate algorithm
     that computes tiny sing vals.
     as accurately as large ones,
     as many correct leading digits
See links to papers in typed notes and class web page

(3) **Updating:** given orals and eves of $A$, compute them for $A \pm xx^T$ more cheaply than starting from scratch

All these options apply to $A = A^T$ and SVD

**Algorithms and costs:**

(1) **Reduce** $A = QTQ^T$

$Q$ orthogonal $T = T^T$ tridiagonal

$[\begin{bmatrix} \cdots & 0 \\ 0 & \cdots \end{bmatrix}]$. Same idea as in Hessenberg reduction in Chap 4

$[\begin{bmatrix} \cdots \\ 0 \end{bmatrix}]$: Hessenberg + symmetry $\implies$ tridiagonal

(LAPACK: ssyr2d). All subsequent stages of algorithms operate on $T$
There is an algorithm for tridiagonal reduction that does $O\left( \frac{n^3}{\text{fast mn size}} \right)$ moving words.
(See typed notes for reference)

When $A$ banded

tridiagonal reductions cost $O(n^2 \cdot bw)$, also communicates less (LAPACK sssbtd)

- Communication avoiding tridiagonal reduction routines not yet in LAPACK

- SVD analogous: Reduce $A$ to bi-diagonal form $A = UV^T$
  $U$, $V$ orthogonal
  
  $B = \begin{bmatrix} & \ast \\
  \ast & \ast \end{bmatrix}$,

  subsequent algorithms operate on $B$
Given $T$, find evals $\lambda$ possibly $\lambda$vecds; many algs, all cost $O(n^2)$ for evals but from $O(n^2)$ to $O(n^3)$ for $\lambda$vecds, varying numerical stability properties

Oldest is QR as in Chap 4 Thm (Wilkinson) With right shift, tridiagonal QR is globally convergent, and usually cubically (correct digits triples at each iteration)

Cost: $O(n^2)$ for evals, but $O(n^3)$ for $\lambda$vecds, more expensive than some later algorithms (LAPACK ssyev)

LAPACK sgesvd uses a variant of QR with additional property of guaranteeing high relative accuracy of SVD for bidiagonal $B$
(see typed notes)

(1.1.2) Improve cost of \( O(n^3) \) for
evecs to \( O(n^2) \), but not
guarantee evecs orthogonal

(1) compute evals \( \text{ssleba in LAPACK} \)

(2) compute evecs using
inverse iteration \( \text{ssstein in LAPACK} \)
\[
X_{n+1} = (T - \lambda I)^{-1} X_n
\]

tri-diagonal \( \Rightarrow \) solve cost \( O(n) \)

\( \lambda_i \) accurate \( \Rightarrow \) few steps

If \( \lambda_i \) and \( \lambda_{i+1} \) very close,
no guarantee of orthogonality

(Ex: suppose \( \lambda_i \) and \( \lambda_{i+1} \) so close
they round to same floating
point number)

Long goal: find orthogonal evecs
for \( O(n^2) \) – solved by MRRR
(1.1.3) Divide + Conquer:

- faster than QR, not as fast as bisection + inverse iteration
- (LAPACK routine ssyevd)
- cost: $O(n^3)$, $n \leq 3$

Same idea used for competing
evals, evecs of $A \pm xx^T$ given
evals, evecs of $A$

(1.1.4) MRRR= Multiple Relatively
Robust Representations

- like inverse iteration, but with
- guaranteed orthogonal evecs
- (Parlett, Dhillon, see typed notes,
web page/ link)

- (LAPACK ssyevr)

Extension to SVD by Paul Willems,
but examples exist with
poor orthogonality in sing. evecs,
so not yet in LAPACK
(open problem)
Beating $O(n^2)$: based on divide-and-conquer, represents even matrix implicitly, $T = ZΛZ^T$

Thm 6.1: One can compute $Z$ in $O(n \cdot \log^* n)$ (p small integer) if represented implicitly (so that we can multiply $Zx$ quickly).

But all evecs cost $O(n^3)$ to get: evecs of $A = QTQ^T = (ΩZ)Λ^2(ΩZ)^T$ get $O(n^3)$

(1.2) or (1.3): few evals and evecs
- Reduce $A = QT AQ = T$ as above
- Use bisection (based on Sylvester’s Thm.) to get a few desired evals, then inverse iteration for evecs
- Cost = $O(n)$ per evec;
  for orthogonality can use MRRR
(2) **High Accuracy:**
Based on Jacobi's Method (historically oldest)
LAPACK sgesvj for SVD only (Drmac + Veselic)

(3) Updating $A = Q \Lambda Q^T$ to $A + \kappa x x^T$ possible to use divide and conquer to get evals of $A + \kappa x x^T$ in $O(n^2)$, not $O(n^3)$

Ma221 Lecture 10 Segment 6

**QR Iteration**

*matlab* demo - cubic convergence
(see typed notes for code)

Cubic convergence follows from analysis of Rayleigh Quotient Iteration
\[ i = 0 \]

choose unit vector \( \mathbf{x}_0 \)

repeat

\[ S_i = p(x_i, A) = x_i^T A x_i \]

\[ y = (A - S_i I)^T x_i \]

\[ x_{i+1} = y / \| y \|_2 \]

\[ i = i + 1 \]

until convergence

(\( S_i \) and/or \( x_i \) stop changing)

Inverse Iteration using Rayleigh Quotient as shift, best available approx eval

given approx eigenvector \( x_i \):

Suppose \( A \mathbf{q} = \lambda \mathbf{q} \), \( \| \mathbf{q} \|_2 = 1 \), \( |(\mathbf{x}_i - \mathbf{q})^T\mathbf{u}| = \epsilon < 1 \)

Want to show \( \| (\mathbf{x}_i - \mathbf{q}) \|_2 = O(\epsilon^2) \)

Bound \( |S_i - \lambda| : \)

\[ S_i = x_i^T A x_i = (x_i - \mathbf{q} + \mathbf{q})^T A (x_i - \mathbf{q} + \mathbf{q}) \]

\[ = (x_i - \mathbf{q})^T A (x_i - \mathbf{q}) + 2 \mathbf{q}^T A (x_i - \mathbf{q}) \]

\[ + (x_i - \mathbf{q})^T A \mathbf{q} + \mathbf{q}^T A \mathbf{q} \]

\[ \leq \| x_i - \mathbf{q} \|_2^2 + \mathbf{q}^T A \mathbf{q} \]
One step of inverse iteration:

(analysis from Chap 4)

\[ \| x_{i+1} - g \| \leq \frac{\| x_i - g \| \cdot | s_i - \lambda |}{\text{gap}} \]

where gap = distance from \( s_i \) to next closest eval

\[ | s_i - \lambda | \leq \frac{1}{\| A x_i - s_i x_i \|_2^2} / \text{gap} \]

\[ = O \left( \frac{1}{\| A x_i - s_i x_i \|_2^2} \right) \cdot | s_i - \lambda | \leq O \left( \frac{1}{\| A x_i - s_i x_i \|_2^2} \right) / \text{gap} \]

\[ \leq \frac{1}{\| A(s_i - I) \|_2} \cdot \frac{| s_i - \lambda |}{\text{gap}} \]

\[ \leq O \left( \frac{1}{\| A(s_i - I) \|_2} \right) \cdot \frac{| s_i - \lambda |}{\text{gap}} \]

\[ = O \left( \frac{1}{\| A(s_i - I) \|_2} \right) \cdot \frac{| s_i - \lambda |}{\text{gap}} \]

\[ = O \left( \frac{1}{\| A(s_i - I) \|_2} \right) \cdot \frac{| s_i - \lambda |}{\text{gap}} \]

\[ = O \left( \frac{1}{\| A(s_i - I) \|_2} \right) \cdot \frac{| s_i - \lambda |}{\text{gap}} \]
Show that QR iteration doing Rayleigh Quotient Iteration implicitly

\[ T - s_i I = QR \]

\[ (T - s_i I)^{-1} = R^{-1} Q^{-1} \]

\[ \text{sym} \]

\[ = R^{-1} Q^T \]

\[ = (R^{-1} Q^T)^T \]

\[ = Q \cdot R^{-T} \]

\[ (T - s_i I)^{-1} R^T = Q \]

last col: \[ (T - s_i I)^n e_n \cdot R_{nn} = g_n = \text{last col of } Q \]

\[ s_c = T_{nn} = e_n^T T e_n = \rho(e_n, T) \]

\[ \Rightarrow g_n \text{ result of one step of Rayleigh Quotient iteration} \]

\[ \text{Updated } T = RQ + s_i I = Q^T T Q \]

\[ \text{(updated } T)(n, n) = g_n^T T g_n = \rho(g_n, T) \]

In practice: QR iteration analogous to Chap 4, by "bulge chasing", cost \( \approx O(n) \) per iteration
\( \Rightarrow \) cost = \( O(n^2) \) to find all evals
Accumulate all Givens rotations to compute evecs costs \( O(n^3) \) so something faster is desired.

Ma221 Lecture 10 Segment 7

Divide and Conquer alg for tridiagonal eigen problem used when want all evals and evecs

Main Ingredient: how to cheaply update \( A = Q \Lambda Q^T \) for \( A + \alpha uu^T : \)

\[
A + \alpha uu^T = Q \Lambda Q^T + \alpha uu^T \\
= Q (\Lambda + \alpha(Q^T u)(u^T Q)) Q^T \\
= Q (\Lambda + \alpha vv^T )Q^T
\]

Need to update eigendecomp of \( \Lambda + \alpha vv^T \)
Compute characteristic polynomial
Lemma: $\det(I + xy^T) = 1 + y^Tx$

\[
\det (A + \alpha v v^T - \lambda I) = \det ((A - \lambda I) (I + \alpha (A - \lambda I)^{-1} v v^T))
\]
\[
= \prod (\lambda_i - \lambda) \left( 1 + \alpha \sum \frac{v_i}{\lambda_i - \lambda} \right)
\]
\[
= \prod (\lambda_i - \lambda) \cdot f(\lambda)
\]

Solve $f(\lambda) = 0$: secular equation

Fig 5.2 in text
\[
f(\lambda) = 1 + \frac{.5}{1 - \lambda} + \frac{.5}{2 - \lambda} + \frac{.5}{3 - \lambda} + \frac{.5}{4 - \lambda}
\]

Fig 5.3 in text
\[
f(\lambda) = 1 + \frac{.001}{1 - \lambda} + \frac{.001}{2 - \lambda} + \frac{.001}{3 - \lambda} + \frac{.001}{4 - \lambda}
\]
can’t use plain Newton

Instead use variation on Newton
approximate $f(\lambda)$ at point by a function
with poles at boundaries $\lambda_i$ and $\lambda_i^-$
\[
g(\lambda) = c_1 + \frac{c_2}{\lambda_i - \lambda} + \frac{c_3}{\lambda_i^- - \lambda}
\]
choose \( c_1, c_2, c_3 \) as in regular Newton to match \( f(x) \) and \( f'(x) \) at each iteration

\[
\begin{array}{c}
\text{Choose } c_1, c_2, c_3 \\
\text{as in regular Newton to match } f(x) \text{ and } f'(x) \text{ at each iteration.}
\end{array}
\]

Solve \( p(t) = 0 \): quadratic

Computing evecs

Lemma: If \( \tilde{A} + \alpha v v^T \) then its evvec is \( (\tilde{A} - \lambda I)^{-1} v \)

cost \( O(n) \)

Proof:

\[
(\tilde{A} + \alpha v v^T) (\tilde{A} - \lambda I)^{-1} v
\]

\[
= (\tilde{A} - \lambda I + \mu I + \alpha vv^T) (\tilde{A} - \lambda I)^{-1} v
\]

\[
= v + \lambda (\tilde{A} - \lambda I)^{-1} v + \alpha v (v^T (\tilde{A} - \lambda I)^{-1} v)
\]

\[
= v + \lambda (\tilde{A} - \lambda I)^{-1} v - v = \lambda (\tilde{A} - \lambda I)^{-1} v
\]
Unfortunately not numerically stable because of cancellation in \( A - \beta I \)

Clever fix in text (Gu, Eisenstat)

Cheaper ways to solve \( f(u) = 0 \)

when \( d_i \) and \( d_{i+1} \) very close, there is an \( f(u) = 0 \), \( d_i \approx d_{i+1} \), available with little work

when \( v \) tiny \( \Rightarrow d \) nearly an eval, requiring no work

called 'deflation' yields smaller secular equation, happens surprisingly often

Write \([Q', \Lambda'] = \text{Eig-update}\([Q, \Lambda, \nu]\) as function we just defined

Use \text{Eig-update} as building block for recursive eigensolver using Divide and Conquer
Need to divide tridiagonal $T$ into 2 problems of half the size, plus rank 1

\[
T = \begin{bmatrix}
a & b & 0 & 0 & \cdots & 0 \\
b & a & b & 0 & \cdots & 0 \\
0 & b & a & b & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & 0 & \cdots & a \\
0 & 0 & 0 & 0 & \cdots & b \\
\end{bmatrix}
\]

\[
T = \begin{bmatrix}
a_{i-1} & \sqrt{c} & 0 & 0 & \cdots & 0 \\
\sqrt{c} & a_i & \sqrt{c} & 0 & \cdots & 0 \\
0 & \sqrt{c} & a_{i+1} & \sqrt{c} & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & 0 & \cdots & a \\
0 & 0 & 0 & 0 & \cdots & b \\
\end{bmatrix}
\]

\[
= \text{diag}(T_1, T_2) + \begin{bmatrix} b_i \end{bmatrix}
\]

\[
U = \begin{bmatrix} 1 & 0 & \cdots & 0 \\
0 & 1 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 1 \\
\end{bmatrix}
\]

\[
\bar{c} = \left\lfloor \frac{n}{2} \right\rfloor
\]
Overall Divide & Conquer Alg

function \([Q, \Lambda] = DC\_eig(T)\)

... return \(T = Q \Lambda Q^T\)

\(n = \text{dim} (T)\)

if \(n \text{ small enough}\)

call QR

else

\(c = n^{\frac{1}{2}}\)

write \(T = \text{diag}(T_1, T_2) + b e u^T\)

... just notation, no work

\([Q_1, \Lambda_1] = DC\_eig(T_1)\)

\([Q_2, \Lambda_2] = DC\_eig(T_2)\)

... note that \(\text{diag}(Q_1, Q_2)\)

... and \(\text{diag}(\Lambda_1, \Lambda_2)\) are

... eigen decompositions of \(\text{diag}(T_1, T_2)\)

\([Q, \Lambda] = \text{Eig\_update(diag}(Q_1, Q_2), \\text{diag}(\Lambda_1, \Lambda_2), b, c, u)\)

end if

return
Algorithms when only a few evals and evecs desired

Recall Sylvester's Thm

\[ A = A^T, \ X \text{ nonsingular} \Rightarrow \]

\[ A \text{ and } XAX^T \text{ have same} \]

\[ \text{Inertia} = (\# \text{evals} < 0, \# \text{evals} = 0, \# \text{evals} > 0) \]

So \( A - \sigma I \) and \( X(A - \sigma I)X^T \) have

same inertia

\[ = (\# \text{evals of } A < \sigma, \# \text{evals of } A = \sigma, \# \text{evals of } A > \sigma) \]

So if \( X(A - \sigma I)X^T \) were diagonal

It would be easy to compute Inertia

By doing this for \( \sigma_1, \sigma_2 \) count \# evals in \([\sigma_1, \sigma_2]\). Repeatedly bisecting interval

Inertia \( (A - \frac{\sigma_1 + \sigma_2}{2} I) \), compute intervals containing evals as narrow as desired.
How to choose $X$ so $X(A-\sigma I)X^\top$ is diagonal?

Start with $A = T$ tridiagonal, do Gaussian Elimination without pivoting

$$T-\sigma I = LDL^T$$

so $\text{Inertia}(T-\sigma I) = \text{Inertia}(D)$

But no pivoting seems numerically dangerous but it is not if done correctly

function $c = \text{Neg count}(T,s)$

... count $c$ = #evals of $T<s$

... $\text{diag}(T) = (a_1, a_2, \ldots, a_n)$

... $\text{offdiag}(T) = (b_1, b_2, \ldots, b_{n-1})$

... only compute diagonal entries $d_i$

... of $D$ in $T-\sigma I = LDL^T$

$c=0$

$b_0 = 0$, $d_0 = 1$

for $i = 1$ to $n$

$$d_i = (a_i-s) - b_{i-1}^2/d_{i-1} \ldots \text{obey parentheses!}$$

if $d_i < 0$, $c = c + 1$

end

We don't need $b_i = i^{th}$ offdiagonal of

$$(T-\sigma I)(i,i+1) = b_i = (LDL^T)(i,i+1) = d_i \cdot l_i$$
Replace usual inner loop
\[ d_i = a_c - s - b_i \cdot d_{i-1} \]
by \[ d_i = a_c - s - b_i \cdot /d_{i-1} \]

Thm: Assuming we don't divide by zero, overflow or underflow,
\( \text{Neg count}(T,s) \) is exactly correct for \( T+E \) where \( E = E^\top + \text{diagonal} \)
\[ E(i,i) = 0 \text{ (diagonal not perturbed)} \]
\[ |E(i,i+1)| \leq 2.5 \varepsilon |T(i,i+1)| \]

Proof: usual approach: replace
\( a \circ \oplus b \) by \( (a \circ \oplus b)(1 + \varepsilon) \)
\[ |\varepsilon| \leq \varepsilon \]. Need to obey parentheses
do \( a_c - s \) first \( \Rightarrow \) we can divide
by all \( 1+\varepsilon \) factors multiplying \( a_c - s \)
to get:
\[ d_i \cdot F_i = a_c - s - b_i \cdot G_i / d_{i-1} \]
\( \uparrow \)
a few \( 1+\varepsilon \) factors
\[ d'_i \cdot F_i = a_c - s - b_{i-1} / G_{i-1} F_{i-1} / (d_{i-1} F_{i-1}) \]
\[ d'_i = a_{i-5} - b'_{i-1}/d_{i-1} \]

= exact recurrence for T+E

since \( d'_i = d_i \). Some factors have same sign, get same
Neg count for both

(see Lemma 5.3 in the text for details)

More is true: This works even if some \( d_{i-1} = 0 \), so we divide by 0
\[ \Rightarrow d_i \text{ infinite} \Rightarrow d_{i+1} = a_{i+1} - 5 \]

i.e. "infinity" disappears.

To get Neg count correctly need to count \( d_{i-1} = 0 \) as <0 and increment <, versus \( d_{i-1} = 0 \) as >0
(i.e. depends on this detail of IEEE arithmetic)

Also can prove \( g(\sigma) \) = Zevs < \( \sigma \) is monotonically increasing function of \( \sigma \) as long as arithmetic correctly rounded
If \( g(\sigma) \) were not monotonic
\[ g(\sigma_1) > g(\sigma_2) \text{ for } \sigma_1 < \sigma_2 \Rightarrow \]

# evals in \([\sigma_1, \sigma_2]\) would be negative

Cost of Negcount is \(O(n)\).

To find one eval via bisection, Negcount needs to be called at most \#bits_in_word_representing \(g\) times

- \(= 64\) in double precision
- \(= 32\) in single precision

because each bisection step provides one more bit in answer

(usually fewer steps) \(\Rightarrow\) cost still \(O(n)\) per eval

\(\Rightarrow\) cost \(O(k \cdot n)\) to compute \(k\) evals

---

Ma221 Lecture 10 Segment 9

Given accurate eval \(dj\) from bisection: use inverse iteration for even
choose \( x_0 \), \( i = 0 \)

repeat
\( i = i + 1 \)

solve \( (T - d_i E^\top) y = x_i \) for \( y \)
\( x_i = y / \| y \|_2 \)

until \( x_i \) converges

Cost = \( O(n) \) per evc, should converge in \( O(1) \) steps because \( d_j \) accurate

Seems like \( O(n^2) \) algorithm for all evcs, but no guarantee of orthogonality when \( d_j \) and \( d_i \) very close. Could try different random \( x_0 \) for different \( d_j \) but no guarantee of orthogonality

Possible fix: take evcs for each cluster of nearby evals (these may lose orthogonality), do QR decomp. use columns of \( Q \) instead. Guarantees orthogonality but 2 problems:

(1) if computed evcs not linearly
independent, then columns $q_i$ of $Q$ may not satisfy $A q_i = q_i$

(2) if cluster is large, $S$ even, cost of QR is $O(nS^2)$, so if $S$ large, cost could rise to $O(n^3)$

\[
\text{MRRR} = \text{Multiple Relatively Robust Representations}
\]

computes all $n$ eigs in $O(n^2)$ ops guarantees orthogonality. Fastest alg for large problems (defaults to other alg for smaller problems)

Complicated to explain; see paper by Dhillon and Parlett on class web page, or Ma221 lectures from 2004.

Jacobi's method: oldest method, slow, but most accurate. uses a "relative version" of Weyl's theorem to bound relative change in eigs
so tiny evals can be as accurate as big evals. Drmac and Veselic paper for details.

All algorithms discussed so far extend to SVD via \[
\begin{bmatrix}
O \cdot A \\
A^+ \cdot O
\end{bmatrix}
\]

One exception is MRRR, because some matrices known where accuracy less than expected:
open problem