4.4. Problem Transformations

Shifts

$$\begin{aligned} Ax &= \lambda x \\ -\sigma x &= -\sigma x \end{aligned}$$

$$(A - \sigma I)x = (\lambda - \sigma)x$$

eigenvalues shifted
eigenvectors unaffected.

Inversion.

$Ax = \lambda x$ $\quad$ A nonsingular.

$$\Rightarrow \quad x = \lambda A^{-1} x$$

$$\Rightarrow \quad A^{-1}x = \frac{1}{\lambda} x$$

eigenvalues reciprocal
eigenvectors unaffected

Powers

$$Ax = \lambda x$$

$$A^2 x = A(Ax) = A\lambda x = \lambda Ax = \lambda^2 x$$

eigenvalues squared
eigenvectors unaffected

Polynomials.

$$p(t) = c_0 + c_1 t + c_2 t^2 + \cdots + c_n t^n$$
$$p(A) = c_0 + c_1 A + c_2 A^2 + \cdots + c_n A^n$$

$$Ax = \lambda x$$

$$p(A)x = p(\lambda)x$$

eigenvalues $p(\lambda)$
eigenvectors unaffected.

Similarity.

A B similar, $\exists\, T$ nonsingular

$$B = T^{-1} A T$$

$$By = \lambda y \Rightarrow T^{-1}ATy = \lambda y$$
$$\Rightarrow A(Ty) = \lambda(Ty)$$

eigenvalues unaffected
eigenvectors transformed by T

4.5 Computing Eigenvalues & Eigenvectors.

4.5.1 POWER ITERATION

Algorithm: Power Iteration

x_0 = arbitrary initial vec

for $k = 1, 2, \cdots$

$x_k = A x_{k-1}$ [generate next vector]

end

Algorithm: Normalized Power Iteration

x_0 = arbitrary nonzero vector

for $k = 1, 2, \cdots$

$y_k = A x_{k-1}$ [generate next vector]

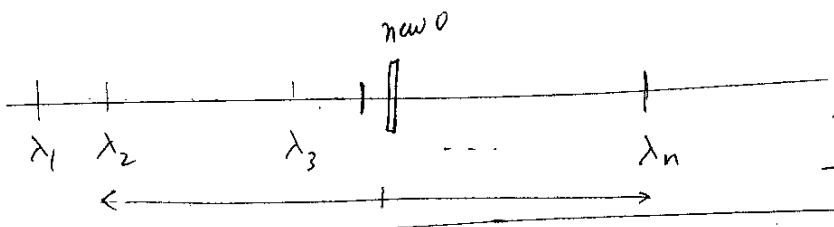
$x_k = y_k / \|y_k\|_\infty$ [normalize]

- convergence rate of power iteration depends on ratio $|\lambda_2 / \lambda_1|$
- possible to choose a shift σ, s.t.

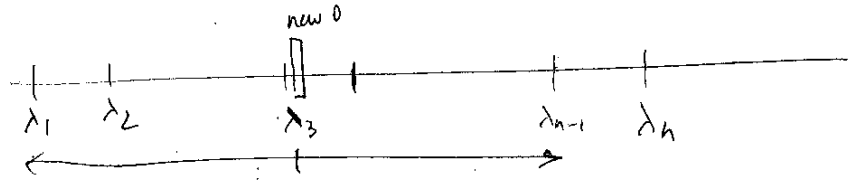
$$\left| \frac{\lambda_2 - \sigma}{\lambda_1 - \sigma} \right| < \left| \frac{\lambda_2}{\lambda_1} \right|$$

hence speed up convergence

can compute either of the extreme eigenvalues of A



- shift by $\frac{\lambda_2 + \lambda_n}{2}$
- λ_1 largest in mag
- minimize $|\lambda_2 / \lambda_1|$



- shift by $\frac{\lambda_1 + \lambda_{n-1}}{2}$
- λ_n largest in mag
- minimize $|\lambda_n / \lambda_1|$

4.5. Computing Eigenvalues & Eigenvectors.

4.5.1 Power Iteration

$$x_k = A x_{k-1}$$

$x_k \to$ converges to eigenvector corresponding to largest eigenvalue.

$$x_0 = \alpha_1 v_1 + \alpha_2 v_2 + \dots + \alpha_n v_n$$

$$A^k x_0 = \alpha_1 \lambda_1^k v_1 + \alpha_2 \lambda_2^k v_2 + \dots + \alpha_n \lambda_n^k v_n$$

$$= \lambda_1^k \left(\alpha_1 v_1 + \sum_{i=2}^{n} \alpha_i \left(\frac{\lambda_i}{\lambda_1}\right)^k v_i \right)$$

$\left(\frac{\lambda_i}{\lambda_1} < 1\right) \iff \to 0$ as $k \to \infty$.

$$\to \lambda_1^k \alpha_1 v_1$$

May fail:

- $\alpha_1 = 0$.

unlikely; in practice, rounding error will introduce such a component.

- repeated λ_1

  ⇝ converge to vector in subspace associated with λ_1

- for real A, real x_0 can't get complex value.

→ rescale x_k to avoid underflow & overflow.

→ convergence rate depends on $\left|\frac{\lambda_2}{\lambda_1}\right|$ $\left(> \left|\frac{\lambda_2 - \sigma}{\lambda_1 - \sigma}\right| \right)$ shift:

- can find extreme eigenvalues λ_1, λ_n

4.5.2 Inverse Iteration

$$A^{-1}v = \frac{1}{\lambda}v$$

- find smallest (in mag.) eigenvalue of A
  largest (") " A^{-1}

Algorithm: inverse iteration

x_0 = arbitrary nonzero vec.
for $k = 1, 2, \ldots$
  Solve $Ay_k = x_{k-1}$ [next vector]
  $x_k = y_k / \|y_k\|_\infty$ [normalize]

- can solve by, e.g., LU or Cholesky
- with appropriate choice of shift σ, inverse iteration can compute any eigenvalue of $A - \sigma I$
  - smallest eigval. of $A - \sigma I$ is λ closest to σ
  - rapid convergence if shift is close to λ
  - particularly useful if estimate to λ is available

4.5.3 Rayleigh Quotient Iteration

- if x is an approx. eigenvec. of A, then

$$\lambda x \approx Ax$$

can be interpreted as L.S. problem, with normal equations

$$x^T x \lambda = x^T A x \Rightarrow \boxed{\lambda = \frac{x^T A x}{x^T x}}$$

- can accelerate power method

Rayleigh Quotient

Matlab 'eig'

$Av = \lambda Bv$

QZ or generalized Schur decomposition
- ignores symmetry of A or B

Cholesky - symm A + spd B

Algorithm: Rayleigh Quotient Iteration

x_0 = arbitrary non-zero vector

for $k = 1, 2, \ldots$

$\sigma_k = x_{k-1}^T A x_{k-1} / x_{k-1}^T x_{k-1}$ [compute shift]

Solve $(A - \sigma_k I) y_k = x_{k-1}$ [next vector]

$x_k = y_k / \|y_k\|_\infty$ [normalize]

end

- quadratic convergence rate (for non-defective eigenval).
  - cubic for normal matrices (including symm) $AA^T = A^TA$
- but must refactor the matrix each iteration - high cost

4.5.4 Deflation

- Assume found $(\lambda_1, \vec{x}_1)$ s.t. $A\vec{x}_1 = \lambda_1 \vec{x}_1$
- Find H (e.g., Householder) s.t. $H\vec{x}_1 = \alpha \vec{e}_1$

Then $\frac{1}{\alpha}\vec{x}_1 = H^{-1}\vec{e}_1$

$HAH^{-1}\vec{e}_1 = HA(\frac{1}{\alpha}\vec{x}_1) = \frac{1}{\alpha} HA\vec{x}_1 = \frac{\lambda_1}{\alpha} H\vec{x}_1 = \lambda_1 \vec{e}_1$

i.e. first column of HAH^{-1} is $\lambda_1 \vec{e}_1$

4.5.5 Simultaneous Iteration

Algorithm: Simultaneous Iteration

X_0 = arbitrary $n \times p$ matrix (rank p) $\quad p < n$.

for $k = 1, 2, \ldots$

$$X_k = A X_{k-1} \qquad [\text{next matrix}]$$

end

Let $S_0 = \text{span}(X_0)$

$S = \text{span}(\{\vec{v}_1, \ldots, \vec{v}_p\})$ first p eigenvectors (largest in $|\lambda|$)

- If no non-zero vec in S is $\perp S_0$, then

$$S_k = A^k S_0$$

has as basis

$$X_k = A^k X_0$$

- and if $|\lambda_p| > |\lambda_{p+1}|$

$$S_k \to S$$

→ "subspace iteration"

Problems:

- need to rescale cols. of X_k*
- each column converging to mult. of v_1
- X_k becoming increasingly ill-conditioned

Therefore ..., orthogonalize →

Algorithm: Orthogonal Iteration

X_0 = arbitrary $n \times p$ matrix of rank p

for $k = 1, 2, \ldots$

$\hat{Q}_k R_k = X_{k-1}$ [reduce QR of X_{k-1}]

$X_k = A \hat{Q}_k$ [next matrix]

end

- converges to same subspace as Simultaneous Iter, but X_k stays well-conditioned
- $\hat{Q}_k \to \hat{Q}$ orthonormal basis for S.