EECS 275 Matrix Computation

Ming-Hsuan Yang

Electrical Engineering and Computer Science
University of California at Merced
Merced, CA 95344
http://faculty.ucmerced.edu/mhyang

Lecture 14

Overview

- Eigenvalue algorithms
- Schur decomposition
- Power iteration

Reading

- Chapter 25-27 of *Numerical Linear Algebra* by Llyod Trefethen and David Bau
- Chapter 7 of *Matrix Computations* by Gene Golub and Charles Van Loan

Eigenvalue algorithm

- Shortcomings of obvious algorithms
  - characteristic polynomial: Compute the coefficients of the characteristic polynomial and find the roots (an ill-conditioned problems in general)
  - it is well known that no formula exists for expressing the roots of an arbitrary polynomial, given its coefficients
  - Abel in 1824 proved that non analog of the quadratic formula can exist for polynomials of degree 5 or more
- Power iteration: The sequence

$$\frac{\mathbf{x}}{\|\mathbf{x}\|}, \frac{A\mathbf{x}}{\|A\mathbf{x}\|}, \frac{A^2\mathbf{x}}{\|A^2\mathbf{x}\|}, \frac{A^3\mathbf{x}}{\|A^3\mathbf{x}\|}, \ldots$$

slowly converges, under certain assumptions, to an eigenvector corresponding to the largest eigenvalue of A

Eigenvalue solvers

- Best general purpose eigenvalue algorithms are based on a different principle: the computation of an eigenvalue revealing factorization of $A \in \mathbb{C}^{m \times m}$ where the eigenvalues appear as entries of one of the factors
  - diagonalization: $A = X \Lambda X^{-1}$ (if and only if A is nondefective)
  - unitary diagonalization: $A = Q \Lambda Q^H$ (if and only if A is normal)
  - unitary triangularization (Schur factorization): $A = QTQ^H$ (no matter whether A is defective or not)
- Most of these direct algorithms proceed in two phases:
  - a preliminary reduction from full to structured form
  - an iterative process for the final convergence
- Any eigenvalue solver must be iterative
- The goal is to produce sequences of numbers that converge rapidly toward eigenvalues

Schur factorization and diagonalization

- Most of the general purpose eigenvalue algorithms in use today proceed by computing the Schur factorization

$$A = QTQ^H \quad Q^H AQ = T$$

- Compute Schur factorization $A = QTQ^H$ by transforming A using a sequence of elementary unitary similarity transformation $X \mapsto Q_j^H X Q_j$, so the product

$$\underbrace{Q_j^H \cdots Q_2^H Q_1^H}_{Q^H} A \underbrace{Q_1 Q_2 \cdots Q_j}_{Q}$$

converges to an upper triangular matrix T as $j \to \infty$

Eigenvalue solvers (cont'd)

- If A is real but not symmetric, then in general it may have complex eigenvalues in conjugate pairs
- An algorithm that computes the Schur factorization will have to be capable of generating complex outputs from real inputs
- If A is Hermitian, then $Q_j^H \cdots Q_2^H Q_1^H A Q_1 Q_2 \cdots Q_j$ is also Hermitian, and thus the limit of the converging sequence is both triangular and Hermitian, hence diagonal
- This implies that the same algorithms that compute a unitary triangularization of a general matrix also compute a unitary diagonalization of a Hermitian matrix

Two phases of eigenvalue computation

- In the first phase, a direct method is applied to produce an upper Hessenberg matrix, i.e., a matrix with zeros below the first subdiagonal
- In the second phase, an iteration is applied to generate a formally infinite sequences of Hessenberg matrices that converge to a triangular form

$$\underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}}_{A \neq A^H} \to \underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}}_{H} \to \underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & & \times \end{bmatrix}}_{T}$$

Two phases of eigenvalue computation (cont'd)

- If A is Hermitian, the two phase approach becomes faster
- The intermediate matrix is a Hermitian Hessenberg matrix, i.e., tridiagonal
- The final result is a Hermitian triangular matrix, i.e., diagonal

$$\underset{A = A^H}{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}} \rightarrow \underset{T}{\begin{bmatrix} \times & \times & & & \\ \times & \times & \times & & \\ & \times & \times & \times & \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}} \rightarrow \underset{D}{\begin{bmatrix} \times & & & & \\ & \times & & & \\ & & \times & & \\ & & & \times & \\ & & & & \times \end{bmatrix}}$$

Reduction to Hessenberg or tridiagonal form

- To compute Schur decomposition $A = QTQ^H$, we would like to apply unitary similarity transformation to A and introduce zeros below the diagonal
- One may apply Householder transformations on the left of A

$$\underset{A}{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}} \xrightarrow{Q_1^H \cdot} \underset{Q_1^H A}{\begin{bmatrix} \boxtimes & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\ \mathbf{0} & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\ \mathbf{0} & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\ \mathbf{0} & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\ \mathbf{0} & \boxtimes & \boxtimes & \boxtimes & \boxtimes \end{bmatrix}} \xrightarrow{\cdot Q_1} \underset{Q_1^H A Q_1}{\begin{bmatrix} \boxtimes & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\ \boxtimes & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\ \boxtimes & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\ \boxtimes & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\ \boxtimes & \boxtimes & \boxtimes & \boxtimes & \boxtimes \end{bmatrix}}$$

- Unfortunately, to complete the similarity transformation, we need to multiply by Q_1 on the right of A
- The zeros that were introduced by Householder transformation Q_1^H are destroyed by rotation of Q_1 in similarity transformation

Reduction by Householder transformations

- The right strategy is to introduce zeros selectively
- Select a Householder reflector Q_1^H that leaves the first row unchanged
- When multiplied on the left of A, it forms linear combinations of only rows $2, \ldots, m$ to introduce zeros into rows $3, \ldots, m$ of the first column
- When multiplied on the right of $Q_1^H Q$, it leaves the first column unchanged, and forms linear combinations of columns $2, \ldots, m$ so it does not alter the zeros that have been introduced

$$\underset{A}{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}} \underset{Q_1^H \cdot}{\rightarrow} \underset{Q_1^H A}{\begin{bmatrix} \times & \times & \times & \times & \times \\ \boxtimes & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\ \mathbf{0} & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\ \mathbf{0} & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\ \mathbf{0} & \boxtimes & \boxtimes & \boxtimes & \boxtimes \end{bmatrix}} \underset{\cdot Q_1}{\rightarrow} \underset{Q_1^H A Q_1}{\begin{bmatrix} \times & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\ \times & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\ & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\ & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\ & \boxtimes & \boxtimes & \boxtimes & \boxtimes \end{bmatrix}}$$

Reduction by Householder transformations

- Same idea is repeated to introduce zeros into subsequent columns

$$
\underset{Q_1^H A Q_1}{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \end{bmatrix}}
\rightarrow
\underset{Q_2^H Q_1^H A Q_1}{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\ & \mathbf{0} & \boxtimes & \boxtimes & \boxtimes \\ & \mathbf{0} & \boxtimes & \boxtimes & \boxtimes \end{bmatrix}}
\rightarrow
\underset{Q_2^H Q_1^H A Q_1 Q_2}{\begin{bmatrix} \times & \times & \boxtimes & \boxtimes & \boxtimes \\ \times & \times & \boxtimes & \boxtimes & \boxtimes \\ & \times & \boxtimes & \boxtimes & \boxtimes \\ & & \boxtimes & \boxtimes & \boxtimes \\ & & \boxtimes & \boxtimes & \boxtimes \end{bmatrix}}
$$

- Repeating this process $m-2$ times, we have a product in Hessenberg form

$$
\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}
$$

$$
\underbrace{Q_{m-1}^H \cdots Q_2^H Q_1^H A}_{Q^H} A \underbrace{Q_1 Q_2 \cdots Q_{m-2}}_{Q} = H
$$

Householder reduction to Hessenberg form

- Algorithm:

  for $k = 1$ to $m - 2$ **do**
  $\mathbf{x} = A_{k+1:m,k}$
  $\mathbf{v}_k = \text{sign}(\mathbf{x}_1)\|\mathbf{x}\|_2\mathbf{e}_1 + \mathbf{x}$
  $\mathbf{v}_k = \frac{\mathbf{v}_k}{\|\mathbf{v}_k\|_2}$
  $A_{k+1:m,k:m} = A_{k+1:m,k:m} - 2\mathbf{v}_k(\mathbf{v}_k^H A_{k+1:m,k:m})$
  $A_{1:m,k+1:m} = A_{1:m,k+1:m} - 2(A_{1:m,k+1:m}\mathbf{v}_k)\mathbf{v}_k^H$
  end for
- Work for Hessenberg reduction: $\sim \frac{10}{3}m^3$ or $O(m^3)$ flops
- If A is Hermitian, the algorithm will reduce A to tridiagonal form
- Since A is Hermitian, $Q^H AQ$ is also Hermitian, and any Hermitian Hessenberg matrix is tridiagonal
- Work for tridiagonal reduction: $\sim \frac{4}{3}m^3$ or $O(m^3)$ flops

Rayleigh quotient and inverse iteration

- First examine classical eigenvalue algorithms with real matrices
- Rayleigh quotient of a vector $\mathbf{x} \in \mathbb{R}^m$ is the scalar

$$r(\mathbf{x}) = \frac{\mathbf{x}^\top A\mathbf{x}}{\mathbf{x}^\top \mathbf{x}}$$

- If $\mathbf{x}$ is an eigenvector, then $r(\mathbf{x}) = \lambda$ is the corresponding eigenvalue
- Given $\mathbf{x}$, what scalar α acts most like an eigenvalue for $\mathbf{x}$ in the sense of minimizing $\|A\mathbf{x} - \alpha\mathbf{x}\|_2$?
- An $m \times 1$ least squares of the form $\mathbf{x}\alpha \approx A\mathbf{x}$ ($\mathbf{x}$ is the matrix, α is the unknown, $A\mathbf{x}$ is the right hand side), and the solution is

$$\alpha = (\mathbf{x}^\top \mathbf{x})^{-1}\mathbf{x}^\top (A\mathbf{x}) = r(\mathbf{x})$$

- $r(\mathbf{x})$ is a natural eigenvalue estimate to consider if $\mathbf{x}$ is close to, but not necessarily equal to, an eigenvector

Gradient of Rayleigh quotient and eigenvector

- Consider $\mathbf{x} \in \mathbb{R}^m$ as a variable so that r is a function: $\mathbb{R}^m \to \mathbb{R}$
- Interested in the local behavior of $r(\mathbf{x})$ when $\mathbf{x}$ is near an eigenvector
- One way to quantitatively approach this is to compute the partial derivatives of $r(\mathbf{x})$ with respect to the coordinate x_j

$$\begin{aligned} \frac{\partial r(\mathbf{x})}{\partial x_j} &= \frac{\frac{\partial}{\partial x_j}(\mathbf{x}^\top A\mathbf{x})}{\mathbf{x}^\top\mathbf{x}} - \frac{(\mathbf{x}^\top A\mathbf{x})\frac{\partial}{\partial x_j}(\mathbf{x}^\top\mathbf{x})}{(\mathbf{x}^\top\mathbf{x})^2} \\ &= \frac{2(A\mathbf{x})_j}{\mathbf{x}^\top\mathbf{x}} - \frac{(\mathbf{x}^\top A\mathbf{x})2x_j}{(\mathbf{x}^\top\mathbf{x})^2} = \frac{2}{\mathbf{x}^\top\mathbf{x}}(A\mathbf{x} - r(\mathbf{x})\mathbf{x})_j \end{aligned}$$

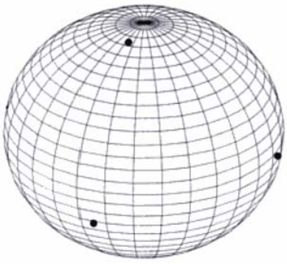
- Collect these partial derivatives into an m-vector, we get the gradient of $r(\mathbf{x})$,

$$\nabla r(\mathbf{x}) = \frac{2}{\mathbf{x}^\top\mathbf{x}}(A\mathbf{x} - r(\mathbf{x})\mathbf{x})$$

- At an eigenvector $\mathbf{x}$ of A, the gradient of $r(\mathbf{x})$ is the zero vector
- Conversely, if $\nabla r(\mathbf{x}) = 0$ with $\mathbf{x} \neq 0$, then $\mathbf{x}$ is an eigenvector and $r(\mathbf{x})$ is the corresponding eigenvalue

Geometric perspective

- The eigenvectors of A are the stationary points of the function $r(\mathbf{x})$
- The eigenvalues of A are the values of $r(\mathbf{x})$ at these stationary points
- Since $r(\mathbf{x})$ is independent of the scale of $\mathbf{x}$, these stationary points lie along lines through the origin in $\mathbb{R}^m$
- If we normalize $\mathbf{x}$ to unit sphere $\|\mathbf{x}\| = 1$, they become isolated points
- For $\mathbb{R}^3$, there are 3 orthogonal stationary points



Convergence rate

- Let $\mathbf{q}_J$ be one of the eigenvectors of A, it can be shown

$$r(\mathbf{x}) - r(\mathbf{q}_J) = O(\|\mathbf{x} - \mathbf{q}_J\|^2) \quad \text{as } \mathbf{x} \to \mathbf{q}_J \tag{1}$$

- Expand $\mathbf{x}$ as a line combination of the eigenvectors $\mathbf{q}_1, \ldots, \mathbf{q}_m$ of A, $\mathbf{x} = \sum_{j=1}^{m} a_j \mathbf{q}_j$, then

$$r(\mathbf{x}) = \frac{\sum_{j=1}^{m} a_j^2 \lambda_j}{\sum_{j=1}^{m} a_j^2}$$

- Thus, $r(\mathbf{x})$ is a weighted mean of the eigenvalues of A, with the weights equal to the squares of the coordinates of $\mathbf{x}$ in the eigenvector basis
- Due to this squaring of the coordinates, if $|a_j/a_J| < \varepsilon$ for $j \neq J$, then $r(\mathbf{x}) - r(\mathbf{q}_J) = O(\varepsilon^2)$
- Rayleigh quotient is a quadratically accurate estimate of an eigenvalue

Power iteration

- Produce a sequence $\mathbf{v}^{(i)}$ that converges to an eigenvector corresponding to the largest eigenvalue of A
- Algorithm

  Initialize $\mathbf{v}^{(0)}$ randomly with $\|\mathbf{v}^{(0)}\| = 1$
  for $k = 1, 2, \ldots$ **do**
  $\mathbf{w} = A\mathbf{v}^{(k-1)}$ // apply A
  $\mathbf{v}^{(k)} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$ //normalize
  $\lambda^{(k)} = (\mathbf{v}^{(k)})^\top A\mathbf{v}^{(k)}$ // Rayleigh quotient
  end for

- Let $\mathbf{v}^0$ denote the linear combination of the orthonormal eigenvectors $\mathbf{q}_i$, we can analyze power iteration

$$\mathbf{v}^{(0)} = a_1\mathbf{q}_1 + a_2\mathbf{q}_2 + \cdots + a_m\mathbf{q}_m$$

- Since $\mathbf{v}^{(k)}$ is a multiple of $A^k\mathbf{v}^{(0)}$, we have some constant c_k

$$\begin{aligned}
\mathbf{v}^{(k)} &= c_k A^k \mathbf{v}^{(0)} \\
&= c_k(a_1\lambda_1^k\mathbf{q}_1 + a_2\lambda_2^k\mathbf{q}_2 + \cdots + a_m\lambda_m^k\mathbf{q}_m) \qquad (2) \\
&= c_k\lambda_1^k(a_1\mathbf{q}_1 + a_2(\lambda_2/\lambda_1)^k\mathbf{q}_2 + \cdots + a_m(\lambda_m/\lambda_1)^k\mathbf{q}_m)
\end{aligned}$$

Power iteration (cont'd)

Theorem

Suppose $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_m| \geq 0$*, and* $\mathbf{q}_1^\top \mathbf{v}^{(0)} \neq 0$*, then after* k *iterations*

$$\|\mathbf{v}^{(k)} - (\pm \mathbf{q}_1)\| = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right), \quad |\lambda^{(k)} - \lambda_1| = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^{2k}\right)$$

as $k \to \infty$*. The* $\pm$ *sign means that at each step* k*, one or the other choice of sign is to be taken, and then the indicated bound holds.*

Proof.

The first equation follows from the power iteration (2) since $a_1 = \mathbf{q}_1^\top \mathbf{v}^{(0)} \neq 0$ by assumption. The second one follows from this and quadratical error (1). If $\lambda_1 > 0$, then the $\pm$ signs are all $+$ or all $-$, whereas if $\lambda_1 < 0$, they alternate. □

Power iteration (cont'd)

- Can be used to compute the spectral radius (supremum among the absolute values of the spectrum) of a matrix

$$\rho(A) = \sup\{|\lambda_i|\}$$

  where λ_i is an eigenvalue of A
- However, it has limited use
  - it can find only the eigenvector corresponding to the largest eigenvalue
  - the convergence is linear, reducing the error only by a constant factor $\approx |\lambda_2/\lambda_1|$ at each iteration
  - the quality of this factor depends on having a largest eigenvalue that is significantly larger than the others
- Google uses it to compute the PageRank of documents for search
- More efficient than other methods of finding the dominant eigenvector for matrices that are well-conditioned and as sparse as the web