Numerical Methods I

MATH-GA 2010.001/CSCI-GA 2420.001

Benjamin Peherstorfer
Courant Institute, NYU

Based on slides by G. Stadler and A. Donev

Today

Last time

- QR factorization
- Condition of finding eigenvalues

Today

- Computing eigenvalues
- Singular value decomposition (SVD)

Announcements

- Homework 3 is due Mon, Oct 24 before class

Recap: Numerically finding eigenvalues

For a matrix $A \in \mathbb{C}^{n \times n}$ (potentially real), we want to find $\lambda \in \mathbb{C}$ and $\boldsymbol{x} \neq 0$ such that

$$A\boldsymbol{x} = \lambda \boldsymbol{x}.$$

Most relevant problems:

- A symmetric (and large)
- A spd (and large)
- A stochastic matrix, i.e., all entries $0 \leq a_{ij} \leq 1$ are probabilities, and thus $\sum_j a_{ij} = 1$.

Recap: How hard are they to find numerically?

- This is a nonlinear problem.
- How difficult is this? Eigenvalues are the roots of the characteristic polynomial. Also, any polynomial is the characteristic polynomial of a matrix $\rightsquigarrow$ For matrices larger than 4×4, eigenvalues cannot be computed in closed form (Abel's theorem).
- Must use an iterative algorithm $\rightsquigarrow$ this is fundamentally different from what we have seen previously when solving systems of *linear* equations! These algorithms (LU, QR) give the *exact* solution in *exact* arithmetic in finite number of steps. We *cannot* expect something similar for computing eigenvalues!

Recap: Condition of finding eigenvalues of a matrix

The absolute condition number of determining a simple eigenvalue λ_0 of a matrix $\boldsymbol{A} \in \mathbb{C}^{n \times n}$ with respect to the $\|\cdot\|_2$ is

$$\kappa_{\text{abs}} = \frac{1}{|\cos(\angle(\boldsymbol{x}, \boldsymbol{y}))|}$$

and the relative condition number is

$$\kappa_{\text{rel}} = \frac{\|\boldsymbol{A}\|}{|\lambda_0 \cos(\angle(\boldsymbol{x}, \boldsymbol{y}))|},$$

where $\boldsymbol{x}$ is an eigenvector of $\boldsymbol{A}$ for the eigenvalue λ_0 ($\boldsymbol{A}\boldsymbol{x} = \lambda_0 \boldsymbol{x}$) and $\boldsymbol{y}$ an adjoint eigenvector ($\boldsymbol{A}^H \boldsymbol{y} = \bar{\lambda}_0 \boldsymbol{y}$).

Recap: Bounding error in eigenvalue computation

Let $\boldsymbol{A} \in \mathbb{C}^{n \times n}$ be a Hermitian matrix and let $(\hat{\lambda}, \hat{\boldsymbol{x}})$ be a computed approximation of an eigenvalue/eigenvector pair $(\lambda, \boldsymbol{x})$ of $\boldsymbol{A}$. Defining the residual

$$\hat{\boldsymbol{r}} = \boldsymbol{A}\hat{\boldsymbol{x}} - \hat{\lambda}\hat{\boldsymbol{x}}\,, \qquad \hat{\boldsymbol{x}} \neq \boldsymbol{0}\,,$$

it then follows that

$$\min_{\lambda_i \in \sigma(\boldsymbol{A})} |\hat{\lambda} - \lambda_i| \leq \frac{\|\hat{\boldsymbol{r}}\|_2}{\|\hat{\boldsymbol{x}}\|_2}\,,$$

where $\sigma(\boldsymbol{A}) = \{\lambda | \lambda \text{ is an eigenvalue of } \boldsymbol{A}\}$ is the spectrum of $\boldsymbol{A}$.

Proof $\rightsquigarrow$ board

What is special about this bound?

Recap: Bounding error in eigenvalue computation

Let $\boldsymbol{A} \in \mathbb{C}^{n \times n}$ be a Hermitian matrix and let $(\hat{\lambda}, \hat{\boldsymbol{x}})$ be a computed approximation of an eigenvalue/eigenvector pair $(\lambda, \boldsymbol{x})$ of $\boldsymbol{A}$. Defining the residual

$$\hat{\boldsymbol{r}} = \boldsymbol{A}\hat{\boldsymbol{x}} - \hat{\lambda}\hat{\boldsymbol{x}}\,, \qquad \hat{\boldsymbol{x}} \neq \boldsymbol{0}\,,$$

it then follows that

$$\min_{\lambda_i \in \sigma(\boldsymbol{A})} |\hat{\lambda} - \lambda_i| \leq \frac{\|\hat{\boldsymbol{r}}\|_2}{\|\hat{\boldsymbol{x}}\|_2}\,,$$

where $\sigma(\boldsymbol{A}) = \{\lambda | \lambda \text{ is an eigenvalue of } \boldsymbol{A}\}$ is the spectrum of $\boldsymbol{A}$.

Proof $\rightsquigarrow$ board

What is special about this bound?

- This is an *a posteriori* bound that bounds the error *after* we have computed the result
- We will see many more residual-based *a posteriori* bounds (broadly speaking: the residual is something we can compute, and if the problem is "well-behaved" then the norm of the residual is a reasonable bound of the norm of the error.)

Recap: Condition of computing eigenvectors

- The condition of computing eigenvector $\boldsymbol{x}_i$ for an eigenvalue λ_i depends on the separation between the eigenvalues

$$\kappa = \frac{1}{\min_{i \neq j} |\lambda_i - \lambda_j|}$$

(Quarteroni et al., Section 5)

- Computing $\boldsymbol{x}_i$ can be ill-conditioned if some eigenvalue λ_j is "very close" to the eigenvalue λ_i

- This indicates that multiple eigenvalues require care. Even for Hermitian matrices eigen*vectors* can be hard to compute

The Power Method

Let $\boldsymbol{A} \in \mathbb{C}^{n \times n}$ be diagonalizable matrix and λ_1 be a simple eigenvalue with

$$|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$$

Let $\boldsymbol{x}_0$ be an initial guess that is not orthogonal to the eigenspace of λ_1, then $\boldsymbol{x}_k$ obtained via the iterations

$$\boldsymbol{z}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k \tag{1}$$
$$\boldsymbol{x}_{k+1} = \boldsymbol{z}_{k+1} / \|\boldsymbol{z}_{k+1}\|_2 \tag{2}$$

will converge to the normalized eigenvector of $\boldsymbol{A}$ corresponding to λ_1 for $k \to \infty$.

This process is called the power method.

Proof $\rightsquigarrow$ board

$$x_k = \frac{A^k x_0}{\| A^k x_0 \|_2}$$

A is diagonalizable : vectors $\{v_i\}_{i=1}^n$

$$x_0 = \sum_{i=1}^n \alpha_i v_i \qquad \alpha_1 \neq 0$$

$$A^k x_0 = V \Lambda^k V^{-1} x_0 = V \Lambda^k \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \sum_{i=1}^n \alpha_i \lambda_i^k v_i$$

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

$$\boxed{\left| \frac{\lambda_i}{\lambda_1} \right| < 1}$$

$$x_k = \frac{\alpha_1 \lambda_1^k \left(v_1 + \gamma^{(k)} \right)}{\| \alpha_1 \lambda_1^k (v_1 + \gamma^{(k)}) \|_2} = \eta_k \frac{v_1 + \gamma^{(k)}}{\| v_1 + \gamma^{(k)} \|_2}$$

$\| \gamma^{(k)} \| \to 0$ for $k \to \infty$

$$x_k \longrightarrow \frac{v_1}{\| v_1 \|} \eta_k$$

Convergence speed

$$\|x_k - (\pm v_1)\|_2 \in \mathcal{O}\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right)$$

Eigenvalue

$$\frac{y^T A y}{y^T y} = \frac{y^T \lambda y}{y^T y} = \lambda$$

$$\lambda_1^{(k)} = \frac{x_k^T A x_k}{x_k^T x_k}$$

Power method (cont'd)

Start with initial guess $\boldsymbol{x}_0$ and then iterate

- Compute matrix-vector product and normalize it

$$\boldsymbol{x}_k = \frac{\boldsymbol{A}\boldsymbol{x}_{k-1}}{\|\boldsymbol{A}\boldsymbol{x}_{k-1}\|}$$

- Obtain eigenvalue estimate (note that $\|\boldsymbol{x}_k\| = 1$)

$$\lambda_1^{(k)} = \boldsymbol{x}_k^H \boldsymbol{A}\boldsymbol{x}_k$$

- Test for convergence? How?

Power method (cont'd)

Start with initial guess $\boldsymbol{x}_0$ and then iterate

- Compute matrix-vector product and normalize it

$$\boldsymbol{x}_k = \frac{\boldsymbol{A}\boldsymbol{x}_{k-1}}{\|\boldsymbol{A}\boldsymbol{x}_{k-1}\|}$$

- Obtain eigenvalue estimate (note that $\|\boldsymbol{x}_k\| = 1$)

$$\lambda_1^{(k)} = \boldsymbol{x}_k^H \boldsymbol{A}\boldsymbol{x}_k$$

- Test for convergence? How? Compute residual

$$\boldsymbol{r}_k = \boldsymbol{A}\boldsymbol{x}_k - \lambda_1^{(k)}\boldsymbol{x}_k$$

and terminate if the error estimate is small enough (bound if $\boldsymbol{A}$ Hermitian, heuristic otherwise)

$$\min_i |\lambda_i - \lambda_1^{(k)}| \leq \frac{\|\boldsymbol{r}_k\|}{\|\boldsymbol{x}_k\|} < \epsilon$$

- The power method converges linearly

$$\|\boldsymbol{x}_k - (\pm \boldsymbol{v}_1)\| \in \mathcal{O}((|\lambda_2|/|\lambda_1|)^k)$$

- If $\boldsymbol{A}$ is normal, then the eigenvalue estimate converges a bit faster but still linearly

$$|\lambda_1^{(k)} - \lambda_1| \in \mathcal{O}((|\lambda_2|/|\lambda_1|)^{2k})$$

- The power method is fast when the dominant eigenvalue is well separated from the rest

- This conclusion is rather general for all iterative methods, convergence is often good if eigenvalues are well separated and bad otherwise

- The power method is typically too slow to be used in practice

The inverse power method

For any μ not an eigenvalue of $\boldsymbol{A}$:

- The eigenvectors of $(\boldsymbol{A} - \mu \boldsymbol{I})^{-1}$ are the same as the eigenvectors of $\boldsymbol{A}$
- The eigenvalues of $(\boldsymbol{A} - \mu \boldsymbol{I})^{-1}$ are $\{(\lambda_j - \mu)^{-1}\}$

Thus, if we have a good estimate μ of an eigenvalue λ_J of matrix $\boldsymbol{A}$, then

$$\frac{|\lambda_j - \mu|^{-1}}{|\lambda_J - \mu|^{-1}} \ll 1\,, \qquad j \neq J$$

and thus the power method applied to $(\boldsymbol{A} - \mu \boldsymbol{I})^{-1}$ converges rapidly to $\boldsymbol{v}_J$:

$$(\boldsymbol{A} - \mu \boldsymbol{I})\boldsymbol{y}_{k+1} = \boldsymbol{x}_k \tag{3}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1} / \|\boldsymbol{y}_{k+1}\| \tag{4}$$

- Costs?

The inverse power method

For any μ not an eigenvalue of $\boldsymbol{A}$:

- The eigenvectors of $(\boldsymbol{A} - \mu \boldsymbol{I})^{-1}$ are the same as the eigenvectors of $\boldsymbol{A}$
- The eigenvalues of $(\boldsymbol{A} - \mu \boldsymbol{I})^{-1}$ are $\{(\lambda_j - \mu)^{-1}\}$

Thus, if we have a good estimate μ of an eigenvalue λ_J of matrix $\boldsymbol{A}$, then

$$\frac{|\lambda_j - \mu|^{-1}}{|\lambda_J - \mu|^{-1}} \ll 1\,, \qquad j \neq J$$

and thus the power method applied to $(\boldsymbol{A} - \mu \boldsymbol{I})^{-1}$ converges rapidly to $\boldsymbol{v}_J$:

$$(\boldsymbol{A} - \mu \boldsymbol{I})\boldsymbol{y}_{k+1} = \boldsymbol{x}_k \tag{3}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1} / \|\boldsymbol{y}_{k+1}\| \tag{4}$$

- Costs? Requires matrix solve in every iteration; same matrix, different right-hand sides ($\rightsquigarrow$ LU and Cholesky decompositions)
- This algorithm is used in practice to find eigen*vectors* if the eigen*values* are already known

Rayleigh quotient iterations

- The convergence speed of the inverse power method increases with a better eigenvalue estimate
- Combine estimating eigenvalue and eigenvector $\rightsquigarrow$ Rayleigh quotient iteration

Accelerated version of the inverse power method using changing shifts:

- Choose starting vector $\boldsymbol{x}^0$ with $\|\boldsymbol{x}^0\| = 1$. Compute $\lambda^{(0)} = (\boldsymbol{x}^0)^T A \boldsymbol{x}^0$.
- For $i = 0, 1, \ldots$ do

$$(A - \lambda^{(k)} I)\boldsymbol{x}^{k+1} = \boldsymbol{x}^k, \quad \boldsymbol{y}^{k+1} = \boldsymbol{x}^{k+1}/\|\boldsymbol{x}^{k+1}\|.$$

- Compute $\lambda^{(k+1)} = (\boldsymbol{y}^{k+1})^T A \boldsymbol{y}^{k+1}$, and go back.

If it converges (depends on starting point), then it converges *cubically* $\rightsquigarrow$ details in Trefethen & Bau
(This is the only method we will see that converges so quickly!)

The QR algorithm

The QR algorithm

The power method is not well suited for finding all eigenvalues of a matrix $\boldsymbol{A}$

Idea of the QR method: Build a matrix A' that shares the eigenvalues of $\boldsymbol{A}$ via similarity transformations

$$\boldsymbol{A}' = \boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{P}\,, \qquad \boldsymbol{A}, \boldsymbol{A}' \text{ have the same eigenvalues}$$

and for which we know the eigenvalues

The QR algorithm for finding eigenvalues is as follows ($A_0 := A$), and for $k = 0, 1, \ldots$:

- Compute QR decomposition of A_k, i.e., $A_k = Q_k R_k$.
- $A_{k+1} := R_k Q_k$, $k := k + 1$ and go back.

All iterates $A_1, A_2, \ldots$ have the same eigenvalues because

$$Q_k A_{k+1} Q_k^T = Q_k R_k Q_k Q_k^T = Q_k R_k = A_k$$

and A_k converges to a *diagonal* matrix if A is Hermitian and eigenvalues well separated

Intuition why QR method converges

Think of it as the power method applied to many linearly independent vectors $z_1^{(0)}, \ldots, z_n^{(0)}$ at once

Define

$$Z^{(0)} = \begin{bmatrix} | & & | \\ z_1^{(0)} & \cdots & z_n^{(0)} \\ | & & | \end{bmatrix}$$

and define

$$Z^{(k)} = A^k Z^{(0)} = \begin{bmatrix} | & & | \\ z_1^{(k)} & \cdots & z_n^{(k)} \\ | & & | \end{bmatrix}$$

Recall that in the power method we had to re-normalize after each step $\rightsquigarrow$ now we have multiple vectors and therefore also need to orthogonalize $\rightsquigarrow$ QR

With orthogonalization after each iteration, we obtain the algorithm

1. $Z^{(k)} = A\bar{Q}^{(k-1)}$
2. $\bar{Q}^{(k)}R^{(k)} = Z^{(k)}$
3. $A^{(k)} = (\bar{Q}^{(k)})^T A\bar{Q}^{(k)}$

$\rightsquigarrow$ equivalent to the QR method

Summary: Let the QR algorithm be applied to a symmetric real matrix A with well separated eigenvalues

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$$

and eigenvectors matrix V that has nonsingular leading principal submatrices (all upper-left 1×1, 2×2, ... submatrices). Then, for $k \to \infty$, the iterates $A^{(k)}$ converge linearly in

$$\mathcal{O}\left(\max_j \frac{|\lambda_{j+1}|^k}{|\lambda_j|^k}\right)$$

to $\mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ and $\bar{Q}^{(k)}$ to V (up to $\pm$)

- The convergence of the QR algorithm is closely related to that of the power method: It is only fast if all eigenvalues are well separated

- For more general (e.g., non-symmetric) matrices in complex arithmetic, the algorithm converges to the Schur decomposition $A = UTU^H$ with triangular matrix T and unitary matrix U $\rightsquigarrow$ read eigenvalues from diagonal of T

- The work *per iteration* of the basic QR algorithm that we discussed is in $\mathcal{O}(n^3)$ because of the QR factorization in each step; the power method has cost $\mathcal{O}(n^2)$ (mat-vec) per iteration

- There are several key improvements to the basic QR algorithm that bring down the cost per iteration to $\mathcal{O}(n^2)$ (Hessenberg matrices)

- There also can be shifts (compare power method) to accelerate the convergence

- As always with linear algebra routines, the "best" are implemented in LAPACK and can be called via Matlab, numpy, etc

Eigenvalues in Matlab

- In MATLAB, sophisticated variants of the **QR algorithm** (LAPACK library) are implemented in the function *eig*:

$$\Lambda = eig(A)$$

$$[X, \Lambda] = eig(A)$$

- For large or sparse matrices, iterative methods based on the **Arnoldi iteration** (ARPACK library), can be used to obtain a few of the largest/smallest/closest-to-μ eigenvalues:

$$\Lambda = eigs(A, n_{eigs})$$

$$[X, \Lambda] = eigs(A, n_{eigs})$$

- The **Schur decomposition** is provided by $[U, T] = schur(A)$.

Conclusions/summary

- Eigenvalues are **well-conditioned** for **unitarily diagonalizable matrices** (includes Hermitian matrices), but ill-conditioned for nearly non-diagonalizable matrices.
- Eigenvectors are **well-conditioned** only when e**igenvalues are well-separated**.
- Eigenvalue algorithms are **always iterative**.
- The **power method** and its variants can be used to find the **largest or smallest eigenvalue**, and they converge fast if there is a **large separation** between the target eigenvalue and nearby ones.
- Estimating **all eigenvalues and/or eigenvectors** can be done by combining the power method with QR factorizations.
- MATLAB has high-quality implementations of sophisticated variants of these algorithms.

Singular Value Decomposition (SVD)

Let $A \in \mathbb{C}^{m \times n}$. A singular value decomposition of A is a factorization

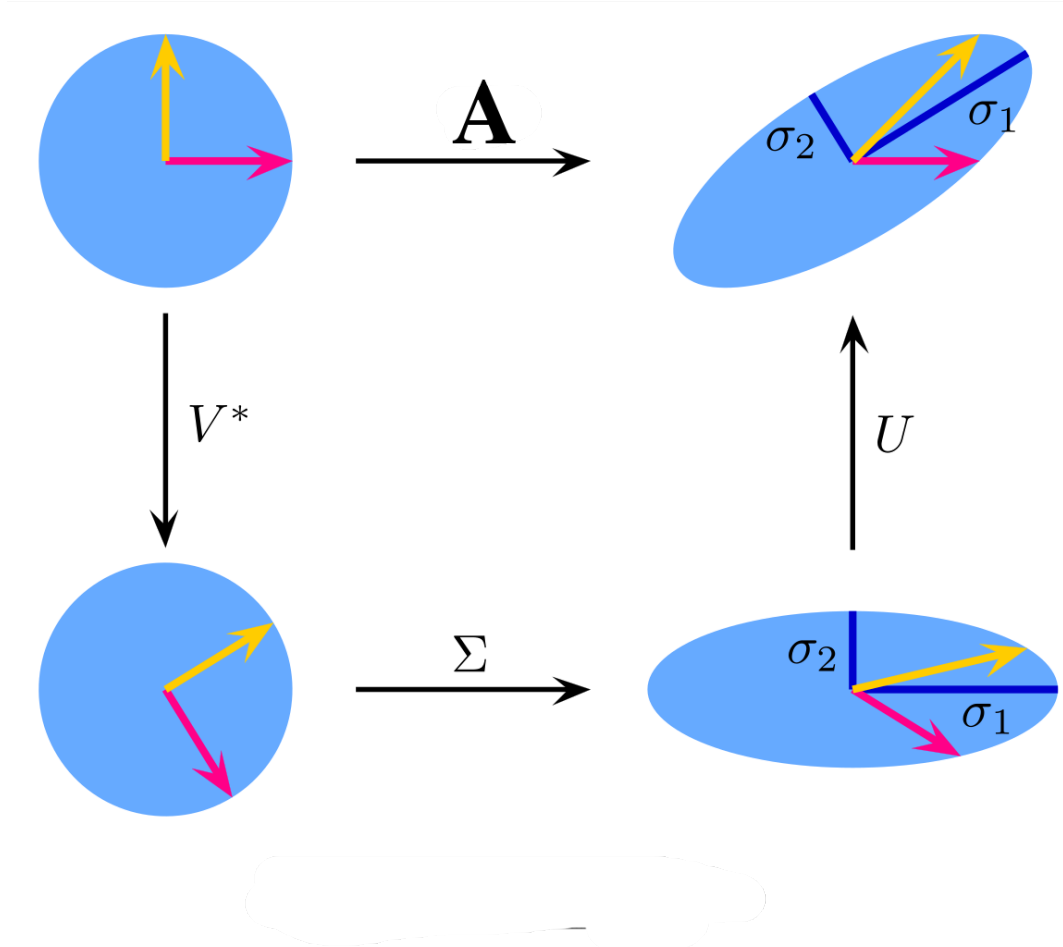
$$A = U \Sigma V^H ,$$

where

$$U \in \mathbb{C}^{m \times m} \text{ is unitary} \tag{5}$$
$$V \in \mathbb{C}^{n \times n} \text{ is unitary} \tag{6}$$
$$\Sigma \in \mathbb{R}^{m \times n} \text{ is diagonal.} \tag{7}$$

Additionally, the diagonal entries σ_j of Σ are non-negative and in non-decreasing order so that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$ where $p \in \min(m, n)$.

- The diagonal matrix Σ is real and has the same shape as A even when A is not square
- The matrices U and V are always square



[Edited from figure by Georg-Johann, Wikipedia]

The image of the unit sphere under a map A is a hyperellipse (in $\mathbb{R}^m$). Thus, with $A = U\Sigma V^H$, have

- The unitary map V^H preserves the sphere (rotating a sphere is a sphere)
- The diagonal matrix Σ stretches the sphere into a hyperellipse aligned with the canonical basis
- The unitary map U rotates or reflects the hyperellipse without changing shape

Existence and uniqueness of SVD

Theorem: Every matrix $A \in \mathbb{C}^{m \times n}$ has a singular value decomposition. Furthermore, the singular values $\{\sigma_j\}$ are uniquely determined, and, if A is square and the σ_j are distinct, the left and right singular vectors $\{u_j\}$ and $\{v_j\}$ are uniquely determined up to complex signs (i.e., complex scalar factors of absolute value 1.) Proof in Trefethen & Bau.

Reduced SVD

$$\underbrace{\begin{bmatrix} \\ A \\ \\ \end{bmatrix}}_{m \times n} = \underbrace{\begin{bmatrix} \\ \hat{U} \\ \\ \end{bmatrix}}_{m \times n} \underbrace{\begin{bmatrix} \\ \hat{\Sigma} \\ \\ \end{bmatrix}}_{n \times n} \underbrace{\begin{bmatrix} \\ \hat{V}^H \\ \\ \end{bmatrix}}_{n \times n}$$

SVD vs. eigenvalue decomposition

SVD expresses a matrix in proper bases for domain and range space to represent it as a diagonal matrix

$$A = U\Sigma V^H$$

We have seen something similar with eigenvectors: A non-defective square matrix A can be expressed as a diagonal matrix of eigenvalues Λ if the range and domain are presented in a basis of eigenvectors

$$A = X\Lambda X^{-1}$$

There are fundamental differences between the SVD and eigenvalue decomposition

- SVD uses two different bases (left and right singular vectors); eigenvalue decomposition uses just one (eigenvectors)
- SVD uses orthonormal bases, whereas eigenvalue basis generally is not orthogonal
- Not all matrices (even square ones) have an eigendecomposition, but all matrices (even rectangular ones) have a singular value decomposition

Typically, eigenvalues tell us something about the behavior of iterative processes that involve the matrix A such as A^k and e^{tA}

Singular values tend to tell us something about A itself

The SVD and matrix properties

In the following:

- The matrix A is of dimension $m \times n$
- $p = \min(m, n)$
- $r \leq p$ is the number of non-zero singular values of A

We now list how the SVD is related to fundamental properties of the matrix A

- The rank of A is r, the number of non-zero singular values.
- The range (column span) of A is $\text{span}(u_1, \ldots, u_r)$, the kernel is $\text{span}(v_{r+1}, \ldots, v_n)$
- $\|A\|_2 = \sigma_1$ and $\|A\|_F = \sqrt{\sigma_1^2 + \cdots + \sigma_r^2}$
- For square A, $|\det(A)| = \prod_{i=1}^{m} \sigma_i$
- The non-zero singular values of A are the square roots of the non-zero eigenvalues of $A^H A$ and AA^H $\rightsquigarrow$ proof

$$A^H A = (U \Sigma V^H)^H (U \Sigma V^H)$$
$$= V \Sigma^H U^H U \Sigma V^H$$
$$= V \Sigma^H \Sigma V^H$$

Full SVD

$$\underbrace{\boldsymbol{A}}_{m\times n} = \underbrace{\boldsymbol{U}}_{m\times m}\underbrace{\boldsymbol{\Sigma}}_{m\times n}\underbrace{\boldsymbol{V}^H}_{n\times n}$$

Reduced SVD

$$\underbrace{\boldsymbol{A}}_{m\times n} = \underbrace{\boldsymbol{U}}_{m\times n}\underbrace{\boldsymbol{\Sigma}}_{n\times n}\underbrace{\boldsymbol{V}^H}_{n\times n}$$

Rank-revealing SVD of a rank r matrix with dimension $m \times n$

$$\underbrace{\boldsymbol{A}}_{m\times n} = \underbrace{\boldsymbol{U}}_{m\times r}\underbrace{\boldsymbol{\Sigma}}_{r\times r}\underbrace{\boldsymbol{V}^H}_{r\times n}$$

Representing matrices via sums of rank-one matrices

Represent A as a sum of M rank-one matrices

$$A = \sum_{j=1}^{M} A^{(j)}$$

There are many possibilities of choosing $A^{(j)}$

- Let $A^{(j)}$ contain the j-th of the m rows of A
- Let $A^{(j)}$ contain the j-th of the n columns of A
- Let $A^{(j)}$ contain one of the mn entries of A
- ...

What is a property of a "sum representation" that we like to see in numerical analysis?

Representing matrices via sums of rank-one matrices

Represent A as a sum of M rank-one matrices

$$A = \sum_{j=1}^{M} A^{(j)}$$

There are many possibilities of choosing $A^{(j)}$

- Let $A^{(j)}$ contain the j-th of the m rows of A
- Let $A^{(j)}$ contain the j-th of the n columns of A
- Let $A^{(j)}$ contain one of the mn entries of A
- ...

What is a property of a "sum representation" that we like to see in numerical analysis?
$\rightsquigarrow$ we can truncate it after a few terms and get a good approximation

$$A \approx \sum_{j=1}^{M'} A^{(j)}$$

with $M' \ll M$

SVD for low-rank approximation

Consider the SVD $A = U\Sigma V^H$, then

$$A = \sum_{j=1}^{r} \sigma_j u_j v_j^H$$

Let us truncate the sum after $1 \leq q \leq r$ terms and define

$$A_q = \sum_{j=1}^{q} \sigma_j u_j v_j^H.$$

Then, A_q is a best rank q approximation of A in the $\|\cdot\|_2$ norm

$$\|A - A_q\|_2 = \inf_{\substack{B \in \mathbb{C}^{m\times n} \\ \text{rank}(B) \leq q}} \|A - B\|_2$$

$\rightsquigarrow$ proof

SVD for low-rank approximation

Consider the SVD $A = U\Sigma V^H$, then

$$A = \sum_{j=1}^{r} \sigma_j u_j v_j^H$$

Let us truncate the sum after $1 \leq q \leq r$ terms and define

$$A_q = \sum_{j=1}^{q} \sigma_j u_j v_j^H.$$

Then, A_q is a best rank q approximation of A in the $\|\cdot\|_2$ norm

$$\|A - A_q\|_2 = \inf_{\substack{B \in \mathbb{C}^{m\times n} \\ \text{rank}(B) \leq q}} \|A - B\|_2 = \sigma_{q+1}$$

The error is the first left out singular value σ_{q+1}! $\rightsquigarrow$ proof

$$A = U \Sigma V^H$$

$$A_r = \sum_{j=1}^{r} \sigma_j u_j v_j^H$$

(1) $\| A - A_q \|_2 = \sigma_{q+1}$

(2) $\| A - A_q \|_2 = \inf_{\substack{B \in \mathbb{C}^{m \times n} \\ \operatorname{rank}(B) \le q}} \| A - B \|_2$

(1): $A - A_q = \sum_{i=q+1}^{r} \sigma_i u_i v_i^H = \begin{bmatrix} | & & | \\ u_{q+1} & \cdots & \end{bmatrix} \begin{bmatrix} \sigma_{q+1} & & \\ & \ddots & \end{bmatrix} \begin{bmatrix} - & v_{q+1}^H & - \\ & \vdots & \end{bmatrix}$

$\| A - A_q \|_2 = \sigma_{q+1}$

(2) Suppose there is B with $\operatorname{rank}(B) \le q$

$\| A - B \|_2 < \| A - A_q \|_2 = \sigma_{q+1}$ (*)

B has rank $\le q$

$n - q$ dimensional null space W

$w \in W \rightarrow Bw = 0$

$w \in W$

$Aw = (A - B)w$

$\|Aw\|_2 = \|(A-B)w\|_2 \leq \|A-B\|_2 \|w\|_2$

$< \sigma_{q+1} \|w\|_2$

Now take v_i

$$Av_i = U \Sigma V^H v_i = U \Sigma \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \sigma_i u$$

$v_1, \dots, v_{q+1}$ span $q+1$-dim subspace Z

$$z \in Z : \quad z = \sum_{i=1}^{q+1} a_i v_i$$

$$\|z\|_2 = \sqrt{\sum_{i=1}^{q+1} a_i^2}$$

$$Az = \sum_{i=1}^{q+1} \sigma_i u_i a_i$$

$$\|Az\|_2 = \sqrt{\sum \sigma_i^2 a_i^2 \underbrace{\|u_i\|_2^2}_{=1}} \geq \sigma_{q+1} \sqrt{\sum a_i^2} = \sigma_{q+1} \|z\|_2$$

Furthermore, in the Frobenius norm $\|\cdot\|_F$, for any $0 \leq q \leq r$, the matrix A_q from the previous slide also satisfies

$$\|A - A_q\|_F = \inf_{\substack{B \in \mathbb{C}^{m \times n} \\ \text{rank}(B) \leq q}} \|A - B\|_F = \sqrt{\sigma_{q+1}^2 + \cdots + \sigma_r^2}$$

SVD and pseudo inverse

The (Moore-Penrose) pseudo inverse of an $m \times n$ matrix that is regular and square is $A^+ = A^{-1}$.

Otherwise, the pseudo inverse is given by

$$A^+ = V \Sigma^+ U^H ,$$

where

$$\Sigma^+ = \text{diag}(\sigma_1^{-1}, \sigma_2^{-1}, \ldots, \sigma_r^{-1}, 0, \ldots, 0)$$

Thus, we can solve least-squares problems $\min_x \|b - Ax\|_2$ by taking the SVD of A, computing A^+, and setting $x = A^+ b$ for the minimal norm solution w.r.t. $\|\cdot\|_2$

The approach we discussed via the QR decomposition is cheaper but the approach via the SVD is sometimes preferred because it allows to easily regularize the problem by truncating small singular values.